

USER GUIDE

Essential Studio for Blazor

Version - v19.4.0.38 | Release Date - December 17, 2021

Menu Bar	55
Getting Started with Blazor Menu Bar Component.....	55
Importing Syncfusion Blazor component in the application.....	55
Adding component package to the application.....	55
Add SyncfusionBlazor service in Startup.cs	55
Adding Menu Bar component to the application	56
Run the application	57
See Also	57
Accessibility in Blazor Menu Bar Component	57
ARIA attributes.....	57
User interaction with keyboard	58
DataSource Binding and Custom Items in Blazor Menu Bar Component.....	58
Self-referential data	58
Custom Menu Bar Items	60
Icons and submenu Items in Blazor Menu Bar Component	63
Icons	63
Navigation	65
Multilevel nesting	66
Use Case Scenarios in Blazor Menu Bar Component.....	67
Scrollable Menu Bar	67
Hamburger Menu.....	69
Mobile View	70
Styles and Appearances in Blazor Menu Bar Component	73
How To	73
Change Orientation in Blazor Menu Bar Component	73
Customize Menu Bar Items in Blazor Menu Bar Component	74
Customize Menu Bar Using Events in Blazor Menu Bar Component.....	78
Menu Bar with Rounded Corner in Blazor Menu Bar Component	79
Open Sub Menu on Menu Item Click in Blazor Menu Bar Component	81
Right to Left in Blazor Menu Bar Component.....	82
MultiSelect Dropdown	83
Getting Started with Blazor MultiSelect Dropdown Component	83
Importing Syncfusion Blazor component in the application.....	83
Adding component package to the application.....	84
Add SyncfusionBlazor service in Startup.cs	84

Adding MultiSelect component to the application.....	84
Binding data source	85
Configure the popup list	86
See Also	87
Data Binding in Blazor MultiSelect Dropdown Component.....	87
Data Source in Blazor MultiSelect Dropdown Component.....	87
Binding local data	88
Binding remote data	90
Binding ExpandoObject.....	95
Binding DynamicObject.....	96
Binding ObservableCollection.....	97
Entity Framework.....	98
Templates in Blazor MultiSelect Dropdown Component	100
Item template	101
Value template.....	102
Header template	103
Footer template	104
No records template	105
Action failure template	106
Grouping in Blazor MultiSelect Dropdown Component	106
Filtering in Blazor MultiSelect Dropdown Component.....	108
Custom Filtering	109
Custom Value in Blazor MultiSelect Dropdown Component.....	110
Value as non-string type	111
Checkbox Grouping in Blazor MultiSelect Component.....	112
CheckBox in Blazor MultiSelect Dropdown Component	113
Select All.....	114
Selection Limit.....	115
Selection Reordering.....	116
Style and appearance in Blazor MultiSelect Dropdown Component	117
Customizing the background color of wrapper element	118
Customizing the appearance of the delimiter wrapper element	118
Customizing the appearance of chips	118
Customizing the dropdown icon's color	118
Customizing the focus color	118

Customizing the disabled component's text color	119
Customizing the color of the placeholder text	119
Customizing the placeholder to add mandatory indicator(*).....	119
Customizing the float label element's focusing color	119
Customizing the outline theme's focus color	119
Customizing the background color of focus, hover, and active item's	120
Customizing the appearance of pop-up element	120
Customizing the color of the checkbox.....	120
Virtualization in Blazor MultiSelect Dropdown Component	120
Localization in Blazor MultiSelect Dropdown Component	121
Blazor server side	121
Blazor WebAssembly	124
Accessibility in Blazor MultiSelect Dropdown Component.....	126
ARIA attributes.....	126
Keyboard interaction	127
Events in Blazor MultiSelect Component.....	127
Blur	127
ValueChange	128
Closed.....	128
Created.....	129
Destroyed.....	129
Focus	130
OnOpen	130
OnClose	131
DataBound	131
Filtering	132
OnActionBegin	133
OnActionFailure	133
OnValueSelect.....	134
Opened.....	135
ChipSelected	135
Cleared	136
OnChipTag.....	136
OnValueRemove	137
ValueRemoved	137

CustomValueSpecifier	138
SelectedAll.....	138
How To	139
Blazor MultiSelect Dropdown List Options with Tooltip.....	139
Numeric TextBox.....	141
Getting Started with Blazor Numeric TextBox Component	141
Importing Syncfusion Blazor component in the application.....	141
Adding component package to the application	142
Add SyncfusionBlazor service in Program.cs	142
Adding NumericTextBox component to the application	143
Run the application	143
Range validation.....	143
Formatting the value.....	143
Precision of numbers	143
See Also	144
Data Binding in Blazor Numeric TextBox Component	144
One-way binding	144
Two-way data binding.....	145
Dynamic value binding.....	145
Number Formats in Blazor Numeric TextBox Component	145
Standard formats	146
Custom formats	146
Globalization in Blazor Numeric TextBox Component.....	146
Blazor server side	146
Blazor WebAssembly	149
Customize the localized text	150
Accessibility in Blazor Numeric TextBox Component	151
Keyboard interaction	151
Native Events in Blazor Numeric TextBox Component.....	152
Bind native events to NumericTextBox.....	152
Pass event data to event handler	152
List of Native events supported	153
Events in Blazor Numeric TextBox Component	153
Blur	153
Created.....	153

Destroyed.....	154
Focus	154
ValueChanged	154
How To	155
Customize the UI appearance of Blazor Numeric TextBox Component	155
Customize the up and down arrow in Blazor Numeric TextBox Component	156
Customize step value and hide spin buttons in Blazor NumericTextBox.....	156
Model Binding in Blazor Numeric TextBox Component	156
PDF Viewer.....	157
Getting Started.....	157
Features in Blazor PDF Viewer Component	157
Server-side application in Blazor PDF Viewer Component	158
WebAssembly application in Blazor PDF Viewer Component	165
Open PDF files in PDF Viewer for Blazor from various storage location	174
Opening a PDF from URL.....	174
Opening a PDF from Cloud.....	175
Opening a PDF from database	176
Opening a PDF from file system.....	177
Opening a PDF from base64 data	177
Opening a PDF from stream.....	177
Saving PDF file in Blazor PDF Viewer Component	178
Save PDF file to Server	178
Save PDF file to Database	178
Download PDF file as a copy	179
Toolbar Customization in Blazor PDF Viewer Component	180
Show or hide toolbar	180
Show or hide navigation toolbar.....	181
Show or hide the toolbar item.....	182
Navigation in Blazor PDF Viewer Component.....	182
Page navigation.....	183
Bookmark navigation	184
Page thumbnail navigation	185
Hyperlink navigation	186
Table of content navigation	187
Magnification in Blazor PDF Viewer Component.....	187

Text Search in Blazor PDF Viewer Component	189
Interaction mode in Blazor PDF Viewer Component	191
Selection mode	191
Panning mode	191
Annotations.....	192
Text markup annotations in Blazor PDF Viewer Component	192
Shape annotations in Blazor PDF Viewer Component.....	198
Stamp annotations in Blazor PDF Viewer Component	202
Sticky notes annotations in Blazor PDF Viewer Component	204
Measurement annotations in Blazor PDF Viewer Component.....	208
Free text annotations in Blazor PDF Viewer Component	214
Comments in Blazor PDF Viewer Component.....	219
Import and Export annotations in Blazor PDF Viewer Component	222
Ink Annotation in the Blazor PDF Viewer component	226
Form filling in Blazor PDF Viewer Component.....	229
Disabling form fields	230
How to draw handwritten signature in the signature field	230
Delete the signature inside the signature field.....	232
Import and export FormFields	232
Importing FormFields using PDF Viewer API.....	232
Exporting FormFields from the PDF document using PDF Viewer API	233
Handwritten Signature in Blazor PDF Viewer Component	233
Adding a handwritten signature to the PDF document.....	233
Editing the properties of handwritten signature	235
Print in Blazor PDF Viewer Component	235
Download in Blazor PDF Viewer Component.....	236
Events in Blazor PDF Viewer Component	237
Adding PDF Viewer events to Blazor component	237
Globalization and RTL in Blazor PDF Viewer Component	238
How To	242
Create PDF Viewer in a popup window in Blazor PDF Viewer Component	242
Load PDF documents dynamically in Blazor PDF Viewer Component	243
View DOCX file in Blazor application in Blazor PDF Viewer Component	244
Unload the PDF document from Viewer in Blazor PDF Viewer Component	244
Import annotations as objects in Blazor PDF Viewer Component.....	245

Suppress the error dialog in Blazor PDF Viewer Component	245
Include the Authorization token in Blazor PDF Viewer Component.....	246
Move the scrollbar to the exact location of annotations	246
View the created PDF document	247
Resolve "Unexpected token T in JSON at position 0" error	248
Pivot Table	249
Getting Started with Blazor Pivot Table Component.....	249
Importing Syncfusion Blazor Pivot Table component in the application.....	249
Adding component package to the application	250
Add SyncfusionBlazor service in Startup.cs	250
Initializing pivot table component in an application	251
Assigning sample data to the pivot table.....	251
Adding fields to row, column, value and filter axes.....	255
Applying formatting to a value field	256
Enable Field List.....	257
Enable Grouping Bar	258
Exploring Filter Axis.....	259
Calculated Field.....	261
See Also	263
Data Binding in Blazor Pivot Table Component	264
JSON	264
CSV	268
Remote Data Binding	306
List binding	308
Mapping	310
Entity Framework.....	312
Values in row axis.....	316
Show 'no data' items.....	317
Show value headers always	318
Customize empty value cells.....	319
Event	320
OLAP in Blazor Pivot Table Component	321
Getting Started.....	321
Data Binding.....	350
OLAP Cube: Elements.....	355

Pivot Chart in Blazor Pivot Table Component.....	357
Data Binding.....	358
Chart Types	358
Accumulation Charts.....	360
Field List	370
Grouping Bar	371
Single Axis	374
Multi Axis	375
Series Customization.....	378
Data Label Customization	379
Axis Customization	381
Legend Customization.....	382
User Interaction	384
Export.....	388
Print.....	390
Drill Down in Blazor Pivot Table Component.....	391
Drill down and drill up.....	391
Drill position	391
Expand all	392
Expand all except specific member(s).....	393
Expand specific member(s)	394
Aggregation in Blazor Pivot Table Component	395
Assigning aggregation type for value fields through API	397
Show desired aggregation types in its dropdown menu	398
Modifying aggregation type for value fields at runtime	399
Hiding aggregation type from button text.....	400
Hiding aggregation type icon from UI.....	401
Number Formatting in Blazor Pivot Table Component.....	402
Custom format	403
Toolbar	405
Calculated Field in Blazor Pivot Table Component	406
Editing through the field list and the grouping bar	409
Renaming the existing calculated field	411
Editing the existing calculated field formula.....	413
Reusing the existing formula in a new calculate field.....	415

Apply the format to the calculated field values.....	418
Supported operators and functions for the calculated field formula.....	419
Field List in Blazor Pivot Table Component.....	420
Grouping Bar in Blazor Pivot Table Component	420
Filtering in Blazor Pivot Table Component	420
Member filtering.....	420
Sorting in Blazor Pivot Table Component	420
Member Sorting	420
Grouping in Blazor Pivot Table Component.....	420
Virtual Scrolling in Blazor Pivot Table Component	421
Defer Layout Update in Blazor Pivot Table Component	421
Row and Column in Blazor Pivot Table Component	421
Width and Height.....	421
State Persistence in Blazor Pivot Table Component.....	421
Hide Totals in Blazor Pivot Table Component.....	421
Show or hide grand totals	421
Conditional Formatting in Blazor Pivot Table Component	421
Drill Through in Blazor Pivot Table Component.....	421
Editing in Blazor Pivot Table Component.....	421
Hyperlink in Blazor Pivot Table Component	421
Toolbar in Blazor Pivot Table Component	421
Tooltip in Blazor Pivot Table Component	421
CSS Customization in Blazor Pivot Table Component.....	421
Hiding Axis.....	421
Globalization in Blazor Pivot Table Component	423
Localization	423
Internationalization.....	436
Right-to-left (RTL).....	437
PDF Export in Blazor Pivot Table Component	438
Changing the pivot table style while exporting	439
Changing the file name while exporting	441
Changing page size while exporting.....	442
Changing page orientation while exporting.....	443
Excel Export in Blazor Pivot Table Component	444
Changing the pivot table style while exporting	445

Changing the file name while exporting	447
CSV Export	448
Accessibility in Blazor Pivot Table Component	449
WAI-ARIA	449
Keyboard Navigation	450
Events in Blazor Pivot Table Component	453
AggregateMenuOpen	453
BeforeExport	453
BeginDrillThrough	453
CalculatedFieldCreate	454
CellClick	454
CellSelected	455
CellSelecting	455
ChartSeriesCreated	456
ConditionalFormatting	456
Drill	457
DrillThrough	458
EnginePopulating	459
EnginePopulated	460
FetchReport	460
FieldListRefreshed	460
FieldDragStart	460
FieldDrop	460
FieldDropped	460
FieldRemove	460
HyperlinkCellClicked	460
LoadReport	460
MemberEditorOpen	460
NewReport	460
OnLoad	460
RenameReport	460
RemoveReport	461
SaveReport	461
ToolbarRender	461
How To	463

Switching to older themes style in Blazor Pivot Table Component	463
Customize number and date format in Blazor Pivot Table Component	465
Customizing loading indicator in Blazor Pivot Table Component	467
Hide empty headers in Blazor Pivot Table Component	468
ProgressBar	469
Blazor ProgressBar Component in Server Side App using Visual Studio	469
Importing Syncfusion Blazor Progress Bar component in the application	469
Adding component package to the application	469
Adding SyncfusionBlazor service in the Startup.cs	470
Adding Progress Bar component	470
Progress Type	470
Types in Blazor ProgressBar Component	471
Linear	471
Circular	471
States in Blazor ProgressBar Component	472
Determinate	472
Indeterminate	472
Buffer	473
Active	473
Striped	473
Customization in Blazor ProgressBar Component	474
Segments	474
Thickness	474
Radius	475
Inner Radius	475
Progress Color and Track Color	475
Range Colors	476
RTL	476
Annotations and Label in Blazor ProgressBar Component	477
Annotations	477
Label	478
Animation in Blazor ProgressBar Component	478
Range in Blazor ProgressBar Component	478
Events in Blazor ProgressBar Component	479
ValueChanged	479

ProgressCompleted	479
AnimationComplete	479
AnnotationRender	480
TextRender	480
Loaded.....	481
ProgressBar	481
Getting Started with Blazor ProgressBar Component.....	481
Importing Syncfusion Blazor component in the application.....	481
Adding component package to the application	482
Add SyncfusionBlazor service in Startup.cs	482
Adding Progress Button component to the application	482
Run the application	483
See Also	483
Native Events in Blazor ProgressBar Component	483
List of Native events supported	483
How to bind click event to Progress Button	483
Types and Icons in Blazor ProgressBar Component.....	484
Types	484
Icons	485
Spinner and Progress in Blazor ProgressBar Component	485
Spinner	486
Progress.....	487
Accessibility in Blazor ProgressBar Component.....	490
ARIA attributes.....	490
Keyboard interaction	490
Styles and Appearances in Blazor ProgressBar Component	491
How To	491
Change text content and styles of the Blazor ProgressBar Component	491
Customize progress using cssClass in Blazor ProgressBar Component.....	492
Hide spinner in Blazor ProgressBar Component	492
Stop Progress Indicator in Blazor ProgressBar Component	493
Trace Events in Blazor ProgressBar Component	493
QueryBuilder	494
Getting Started with Blazor QueryBuilder Component	494
Importing Syncfusion Blazor component in the application.....	494

Adding component package to the application	495
Add SyncfusionBlazor service in Startup.cs	495
Adding Query Builder component to the application.....	496
Run the application	496
See Also	497
Columns in Blazor QueryBuilder Component	497
Auto generation	497
Labels	498
Operators	498
Step	499
Format.....	499
Validations	500
Data Binding in Blazor QueryBuilder Component	501
List Binding.....	502
Remote Data	503
Filtering in Blazor QueryBuilder Component.....	505
Templates in Blazor QueryBuilder Component	507
Value Template	507
Column Template.....	509
Header Template	512
Importing and Exporting in Blazor QueryBuilder Component.....	515
Initial rendering.....	515
Post rendering.....	516
Exporting	517
Exporting to SQL.....	517
Localization in Blazor QueryBuilder Component	518
Styles and Appearances in Blazor QueryBuilder Component.....	521
How To	521
Blazor QueryBuilder Component in WebAssembly App using Visual Studio.....	521
Change Display Mode in Blazor QueryBuilder Component	524
Show Not Operator in Blazor QueryBuilder Component.....	525
Maintain the State Persistence in Blazor QueryBuilder Component.....	526
Readonly in Blazor QueryBuilder Component	527
Right to Left in Blazor QueryBuilder Component	528
Restrict the Groups in Blazor QueryBuilder Component.....	529

Sort the Columns in Blazor QueryBuilder Component	530
RadioButton	532
Getting Started with Blazor RadioButton Component	532
Importing Syncfusion Blazor component in the application.....	532
Adding component package to the application.....	533
Add SyncfusionBlazor service in Startup.cs	533
Adding Radio Button component to the application.....	533
Run the application	534
See Also	534
Native Events in Blazor RadioButton Component	534
List of Native events supported	534
How to bind click event to Radio Button	534
Label and Size in Blazor RadioButton Component.....	535
Label.....	535
Size	535
See Also	536
Styles and Appearances in Blazor RadioButton Component	536
How To	536
Customize Blazor RadioButton Appearance	536
Radio Button Model Binding in Blazor RadioButton Component.....	538
Right-To-Left in Blazor RadioButton Component	539
Set the disabled state in Blazor RadioButton Component	539
Range Selector	540
Blazor Range Selector Component in Server Side App using Visual Studio.....	540
Importing Syncfusion Blazor Range Navigator component in the application	540
Adding component package to the application.....	541
Adding SyncfusionBlazor service in Startup.cs	541
Adding Range Navigator component	541
Populate Range Navigator with Data.....	541
See also	542
Range in Blazor Range Selector Component	542
Value Binding	543
Lightweight in Blazor Range Selector Component.....	544
See Also	545
Series Type in Blazor Range Selector Component	545

Line.....	545
Area.....	546
Step Line.....	547
Type of Data in Blazor Range Selector Component.....	548
Numeric.....	548
Logarithmic	552
DateTime.....	555
Period Selector in Blazor Range Selector Component.....	558
Periods	558
Position	559
Height.....	560
Visibility.....	561
See Also	562
Labels in Blazor Range Selector Component	562
Multi-level labels.....	562
Grouping	563
Smart labels.....	564
Position	564
Labels Customization	565
Grid Lines and Tick Lines in Blazor Range Selector Component	566
Gridline Customization.....	566
Tickline Customization	567
Customization in Blazor Range Selector Component	567
Navigator Appearance	567
Thumb	568
Border	569
Snapping.....	570
Animation.....	570
See Also	571
Tooltip in Blazor Range Selector Component	571
Enable tooltip.....	571
Tooltip Customization	572
Label Format	573
RTL in Blazor Range Selector Component.....	574
Print and Export in Blazor Range Selector Component	575

Print.....	575
Export.....	576
Events in Blazor Range Selector Component.....	577
Loaded.....	577
OnPrintCompleted	578
Changed	579
Resized	580
LabelRender	580
TooltipRender	581
SelectorRender.....	582
Range Slider	583
Getting Started with Blazor Range Slider Component.....	583
Importing Syncfusion Blazor component in the application.....	583
Add Syncfusion Blazor service in Startup.cs (Server-side application)	587
Add Syncfusion Blazor service in Program.cs (Client-side application)	588
Adding component package to the application	588
Adding Slider component to the application	588
Run the application	589
See Also	589
Types in Blazor Range Slider Component	589
Tooltip in Blazor Range Slider Component	590
Buttons.....	590
Orientation in Blazor Range Slider Component.....	591
Ticks in Blazor Range Slider Component.....	591
Step	592
Min and Max	592
Formatting in Blazor Range Slider Component.....	593
Using format API	593
Limits in Blazor Range Slider Component	594
Default and MinRange slider limits.....	594
Range slider limits	595
Handle lock.....	595
CSS Structure in Blazor Range Slider Component.....	596
Customizing the slider track.....	596
Customizing the slider handle.....	596

Customizing the slider limits	596
Customizing the slider ticks	596
Customizing the slider buttons	596
Localization in Blazor Range Slider Component	597
Blazor server side	597
Blazor WebAssembly	599
Accessibility in Blazor Range Slider Component	601
Keyboard interaction	601
How To	601
Customize the bar in Blazor Range Slider Component	601
Customize the limits in Blazor Range Slider Component	606
Customize the thumb in Blazor Range Slider Component	608
Customize the tick label in Blazor Range Slider Component	611
Numeric Range Slider in Blazor Range Slider Component	615
Date Range Slider in Blazor Range Slider Component	617
Time Range Slider in Blazor Range Slider Component	618
Validation of Slider in Blazor Range Slider Component	619
RichTextEditor	621
Getting Started with Blazor RichTextEditor Component	621
Importing Syncfusion Blazor component in the application	621
Adding component package to the application	622
Add SyncfusionBlazor service in Startup.cs	622
Add Rich Text Editor component	623
Run the application	623
Configure the Toolbar	623
Insert images and links	624
Retrieve the formatted content	626
See Also	627
Data Binding in Blazor RichTextEditor Component	627
One-way data binding	627
Two-way data binding	627
Dynamic value binding	628
Editor Modes in Blazor RichTextEditor Component	628
HTML editor	628
Markdown editor	629

See Also	630
Toolbar in Blazor RichTextEditor Component.....	630
Expand Toolbar	630
Multi-row Toolbar	632
Floating Toolbar	633
Toolbar items	634
Custom Tool	637
Quick inline toolbar	638
See Also	640
Inline Mode in Blazor RichTextEditor Component.....	640
Show on select/click.....	640
See Also	641
Paste from Microsoft Word in Blazor RichTextEditor Component.....	641
Microsoft Word to HTML	641
Paste cleanup	641
Prompt dialog.....	642
Paste as plain text	642
Keep format	642
Denied tags	642
Denied attributes	643
Allowed style properties	643
Styling in Blazor RichTextEditor Component	644
Font name and size	644
Custom font and size	646
Formats	648
Font and Background color	650
Editor content styles	652
See Also	655
Image in Blazor RichTextEditor Component	655
Upload options.....	656
Image delete	658
Insert from web	659
Dimension	660
Caption and Alt Text.....	661
Display position.....	661

Image with link.....	661
Resize	662
See Also	662
Link in Blazor RichTextEditor Component.....	663
Insert link	663
Remove Link.....	664
Auto-link.....	664
Manipulation.....	664
See Also	666
Table in Blazor RichTextEditor Component	666
Insert table	667
Quick Toolbar	667
Table Header.....	668
Insert Rows.....	668
Insert Columns	668
Set Color	668
Delete Table	669
Vertical Align	669
Horizontal Align.....	669
Table Styles	669
Table Properties	670
Table cell merge and split	671
Markdown in Blazor RichTextEditor Component	672
Supported Commands	672
Table.....	674
See Also	675
Form validation in Blazor RichTextEditor Component.....	675
Render the editor in a form	675
Validation Rules	676
Validation Message.....	677
Custom Placement of Validation Message	678
Globalization in Blazor RichTextEditor Component.....	679
Localization	679
RTL.....	685
ExecCommand in Rich Text Editor in Blazor RichTextEditor Component.....	685

IFrame Rendering in Blazor RichTextEditor Component	688
IFrame attributes	689
Adding external CSS/Script File	690
See Also	690
Style and appearance in Blazor RichTextEditor Component	690
Customizing the Rich Text Editor's content	690
Customizing the Rich Text Editor's toolbar	691
Customizing the Rich Text Editor's character count	691
Keyboard support in Blazor RichTextEditor Component	692
HTML formation shortcut key.....	692
Markdown formation shortcut key.....	693
Custom key config.....	694
See Also	695
Miscellaneous in Blazor RichTextEditor Component	695
Placeholder	695
Character count	696
Code view.....	696
Undo/Redo Manager	697
Resizable support.....	698
Number and Bullet Format Lists	699
Accessibility in Blazor RichTextEditor Component	699
ARIA attributes.....	699
Events in Blazor RichTextEditor Component	701
OnActionBegin	701
OnActionComplete.....	701
OnDialogOpen.....	702
DialogOpened	702
OnDialogClose.....	702
DialogClosed.....	703
OnQuickToolbarOpen	703
QuickToolbarOpened	703
QuickToolbarClosed	703
OnImageSelected	704
BeforeUploadImage	704
OnImageUploadSuccess.....	704

OnImageUploadFailed	705
OnImageRemoving.....	705
OnImageDelete	705
Created.....	706
Destroyed.....	706
Blur	706
OnToolbarClick.....	707
Focus	707
ValueChange	707
OnResizeStart.....	707
OnResizeStop	708
AfterPasteCleanup	708
How To	708
Blazor RichTextEditor Component in WebAssembly App using Visual Studio	708
Add Google fonts in Blazor RichTextEditor Component	713
Change default font-family in Blazor RichTextEditor Component.....	715
Check image size in Blazor RichTextEditor Component.....	717
Customize placeholder style in Blazor RichTextEditor Component.....	717
Rename images before inserting it in Blazor RichTextEditor Component.....	718
Format code block using toolbar in Blazor RichTextEditor Component	720
Scheduler	721
Getting Started with Blazor Scheduler Component.....	721
Importing Syncfusion Blazor component in the application.....	721
Adding component package to the application.....	722
Add SyncfusionBlazor service in Startup file.....	722
Initialize the Scheduler component	723
Populating appointments	724
Setting date.....	725
Setting view.....	726
Individual view customization	727
See Also	727
Scheduler Interactions in Blazor Scheduler Component	727
Appointments in Blazor Scheduler Component	728
Normal events.....	729
Spanned events.....	729

All-day events.....	730
Recurring events	730
Event fields.....	737
Adding Custom fields	741
Customize the order of the overlapping events	741
Drag and drop appointments.....	743
Appointment Resizing	754
Appointment customization	757
Block Date and Time	762
Readonly	764
Make specific events readonly.....	765
Differentiate the past time events.....	766
Appointments occupying entire cell	767
Display tooltip for appointments.....	767
Appointment filtering	769
Appointment selection	773
Deleting multiple appointments.....	773
Retrieve event details from the UI of an event	774
Get the current view appointments	774
Get the entire appointment collections.....	775
Refresh appointments	776
Data Binding in Blazor Scheduler Component.....	776
Binding local data.....	776
Binding remote data	777
Binding ExpandoObject.....	781
Binding DynamicObject.....	781
Binding ObservableCollection.....	783
Performing CRUD using Entity Framework.....	785
Passing additional parameters to the server	788
Scheduler CRUD actions.....	789
Configuring Scheduler with Google API service.....	791
CRUD actions in Blazor Scheduler Component.....	791
Add	791
Edit	796
Delete.....	804

Drag and drop	809
Resize	810
Virtual Scrolling in Blazor Scheduler Component	810
Virtual scrolling with templates	812
Editor Window Customization in Blazor Scheduler Component	815
Event editor.....	815
Customizing event editor using template	822
How to add recurrence options within editor template.....	826
Apply validations on editor template fields.....	828
Quick popups	830
More events indicator and popup	847
Timezone in Blazor Scheduler	849
Create appointments in different time zones.....	849
Display appointments based on Scheduler time zone.....	851
Display Appointments based on client's time zone	852
Display appointments at same time everywhere regardless of client's time zone	853
Updating StartTime and EndTime after drag and drop appointment based on time zone.....	853
Views in Blazor Scheduler Component	853
Setting specific view on scheduler	853
View specific configuration	855
Extending view intervals	864
Resources and Grouping in Blazor Scheduler Component	865
Resource fields	865
Resource data binding	866
Binding DynamicObject.....	869
Binding ObservableCollection.....	870
Scheduler with multiple resources	873
Resource grouping	874
Working with shared events	884
Simple resource header customization	885
Customizing resource header with multiple columns	889
Expand and collapse resource fields.....	893
Displaying tooltip for resource headers.....	894
Choosing between resource colors for appointments	896
Setting different working days and hours for resources	898

Compact view in mobile.....	901
Adaptive UI in desktop.....	901
See Also	903
Timeline Header Rows in Blazor Scheduler Component	903
Display year and month rows in timeline views	904
Display week numbers in timeline views	905
Timeline view displaying dates of a complete year	905
Customizing the header rows using template	906
Row Auto Height in Blazor Scheduler Component	907
Timeline views with multiple resources	909
Appointments occupying entire cell	910
Header Customization in Blazor Scheduler Component.....	912
Show or Hide header bar	912
How to display the view options within the header bar popup	912
Date header customization.....	914
TimelineYear header customization	916
Customizing header indent cells	918
Timescale Customization in Blazor Scheduler Component	920
Setting different time slot duration	920
Customizing time cells using template	921
Hide the timescale	922
Highlighting current date and time.....	922
Working Days and Hours in Blazor Scheduler Component.....	923
Set working days	923
Hiding weekend days	924
Show week numbers.....	925
Set working hours	926
Scheduler displaying custom hours	927
Setting start day of the week	928
Scroll to specific time and date.....	928
See Also	929
Cell Customizations in Blazor Scheduler Component	929
Setting cell dimensions in Vertical Views.....	929
Setting cell dimensions in Timeline Views	930
Customizing cells using CellTemplate	931

Customizing cells using OnRenderCell event	933
Customizing the minimum and maximum date values	934
State Persistence in Blazor Scheduler Component.....	934
Exporting in Blazor Scheduler Component	935
Excel Exporting.....	936
Exporting calendar events as ICS file	945
Importing events from other calendars.....	947
How to print the Scheduler element	949
Scheduler Dimensions in Blazor Scheduler Component.....	951
Auto Height and Width	951
Height and Width in pixel	952
Height and Width in percentage.....	953
Globalization in Blazor Scheduler Component	953
Blazor Server Side	953
Blazor Web Assembly.....	955
Setting date format	956
Time mode	956
Displaying Scheduler in RTL mode	957
See Also	957
Accessibility in Blazor Scheduler Component.....	957
WAI-ARIA.....	958
Keyboard navigation	958
WebAssembly Performance in Blazor Scheduler Component.....	959
Avoid unnecessary component renders	959
Avoid unnecessary component renders after Scheduler events.....	960
Events in Blazor Scheduler Component.....	961
ActionCompleted	962
Created.....	963
DataBinding.....	964
DataBound	964
Destroyed.....	965
Dragged.....	966
EventRendered.....	967
MoreEventsClicked	968
Navigating	969

OnActionBegin	970
OnActionFailure	971
OnCellClick	972
OnCellDoubleClick.....	973
OnDragStart	974
OnEventClick	975
OnEventDoubleClick	975
OnPopupClose.....	976
OnPopupOpen	978
OnRenderCell	979
OnResizeStart.....	980
Resized	981
Context Menu in Blazor Scheduler Component	982
Style And Appearance in Blazor Scheduler Component	985
How To	987
Blazor Scheduler Component in WebAssembly (WASM) App.....	987
Open Editor Window on Single Click in Blazor Scheduler Component.....	996
Add Multi Events in different slots in Blazor Scheduler Component.....	997
Custom Header in Blazor Scheduler Component	999
Prevent Event Creation and Deletion in Blazor Scheduler Component	1001
Show half-yearly view in Blazor Scheduler Component	1002
Setting Minimum and Maximum Date in Blazor Scheduler Component.....	1004
Custom Editor With Validation in Blazor Scheduler Component	1005
Expand And Collapse Resource Dynamically in Blazor Scheduler Component.....	1010
Enable scroll option on all-day section in Blazor Scheduler Component	1012
Sidebar	1013
Getting Started with Blazor Sidebar Component	1013
Importing Syncfusion Blazor component in the application.....	1014
Add Syncfusion Blazor service in Startup.cs (Server-side application)	1017
Add Syncfusion Blazor service in Program.cs (Client-side application)	1017
Adding component namespace to the application.....	1018
Adding component to the Application.....	1018
Run the application	1019
Enable backdrop	1019
Animate.....	1020

Navigation with Sidebar	1021
Blazor Sidebar vs native sidebar	1041
See Also	1042
Sidebar for specific content in Blazor Sidebar Component	1042
Responsive Sidebar in Blazor Sidebar Component	1047
Dock in Blazor Sidebar Component	1049
Styles and Appearance in Blazor Sidebar Component.....	1052
Customizing the sidebar.....	1053
Customizing the sidebar based on the positions	1053
Customizing the sidebar based on the active state	1053
Customizing the sidebar with dock state.....	1054
Customizing the different types of sidebar.....	1054
Customizing the backdrop of the sidebar	1054
Customizing the sidebar in the RTL direction	1055
How To	1055
Initialize the Blazor Sidebar with ListView	1055
Open and close the Sidebar in Blazor Sidebar Component	1059
Multiple Sidebar in Blazor Sidebar Component.....	1060
Signature	1062
Getting Started with Blazor Signature Component	1062
Prerequisites	1062
Create a new Blazor App in Visual Studio	1062
Install Syncfusion Blazor Inputs NuGet in the App	1062
Add Style Sheet	1062
Add Script Reference	1063
Register Syncfusion Blazor Service.....	1063
Add Syncfusion Blazor Signature component.....	1065
See Also	1066
Smith Chart	1066
Getting Started with Blazor Smith Chart Component.....	1066
Importing Syncfusion Blazor Smith Chart component in the application.....	1066
Adding component package to the application.....	1067
Adding SyncfusionBlazor service in Startup.cs	1067
Adding Smith Chart component	1067
Adding series to Smith Chart	1068

Adding Title	1070
Enable Marker.....	1071
Enable Data Label.....	1072
Enable Legend	1073
Enable Tooltip	1074
See also	1075
Dimensions in Blazor Smith Chart Component.....	1075
Using CSS.....	1075
Using API	1076
Title and Subtitle in Blazor Smith Chart Component	1078
Enable Title.....	1078
Trim Title	1079
Title Customization	1081
Axis in Blazor Smith Chart Component	1082
Labels Customization	1082
Gridlines	1083
Axis Line	1085
Legend in Blazor Smith Chart Component.....	1086
Position	1086
Legend Alignment	1089
Customization	1090
Toggle Visibility	1096
Row and column	1097
Title	1098
Tooltip in Blazor Smith Chart Component	1100
Tooltip Customization	1101
Tooltip Template.....	1102
Marker and Data labels in Blazor Smith Chart Component.....	1103
Marker.....	1103
Data labels.....	1106
Series in Blazor Smith Chart Component.....	1111
Animation.....	1112
Print and Export in Blazor Smith Chart Component	1113
Print.....	1113
Export.....	1114

Events in Blazor Smith Chart Component.....	1115
Loaded.....	1115
OnPrintComplete	1116
OnExportComplete	1116
AxisLabelRendering.....	1117
LegendRendering	1118
SeriesRender	1119
TitleRendering.....	1119
SubtitleRendering	1120
TextRendering.....	1121
SizeChanged	1122
Sparkline	1123
Getting Started with Blazor Sparkline Component.....	1123
Importing Syncfusion Blazor Sparkline component in the application.....	1123
Adding component package to the application	1123
Adding SyncfusionBlazor service in Startup.cs	1123
Adding Sparkline Component	1124
Populate Sparkline with Data.....	1124
Change the type of Sparkline	1125
Adding Data Label	1126
Enable tooltip.....	1126
See also	1127
Dimensions in Blazor Sparkline Component.....	1127
Size for the container.....	1127
Size for Sparkline.....	1128
Chart Types in Blazor Sparkline Component.....	1129
Line.....	1129
Column.....	1130
Pie.....	1130
WinLoss	1131
Area	1131
Axis Customization in Blazor Sparkline Component	1132
Change the value type of the Sparkline Chart	1132
Change the min and the max values of axis.....	1135
Change value of axis.....	1135

Axis Line Customization	1136
Special Points Customization in Blazor Sparkline Component	1137
Add custom color for special points	1137
Tie point color	1138
Range Band in Blazor Sparkline Component.....	1138
Markers in Blazor Sparkline Component	1139
Adding markers.....	1139
Adding special point markers.....	1140
Markers customization	1140
Data Labels in Blazor Sparkline Component	1141
Enable Data Label.....	1141
Data Label customization.....	1142
Format.....	1143
User Interaction in Blazor Sparkline Component	1144
Tooltip	1144
Track Line	1146
Appearance in Blazor Sparkline Component	1147
Right-to-left (RTL).....	1147
Border	1148
Padding	1148
Background	1149
Globalization in Blazor Sparkline Component	1149
Events in Blazor Sparkline Component.....	1150
Loaded.....	1150
OnPointRendering.....	1150
OnPointRegionMouseClicked.....	1151
OnResizing.....	1151
OnSeriesRendering	1152
OnMarkerRendering	1152
OnDataLabelRendering	1153
Methods in Blazor Sparkline Component	1154
Refresh	1154
Spinner	1154
Getting Started with Blazor Spinner Component	1154
Importing Syncfusion Blazor component in the application.....	1154

Adding component package to the application	1155
Add SyncfusionBlazor service in Startup.cs	1155
Add Spinner component	1155
Run the application	1156
Customize the Spinner in Blazor Spinner Component.....	1156
Customize when initializing the Spinner component	1156
Customize after creating the Spinner component.....	1160
Spinner Integration in Blazor Spinner Component.....	1162
Style and appearance in Blazor Spinner Component	1164
Customizing the spinner	1164
Events in Blazor Spinner Component.....	1165
Created.....	1165
OnBeforeOpen	1165
OnBeforeClose	1165
Destroyed.....	1166
SplitButton	1166
Getting Started with Blazor SplitButton Component	1166
Importing Syncfusion Blazor component in the application.....	1166
Adding component package to the application.....	1167
Add SyncfusionBlazor service in Startup.cs	1167
Adding Split Button component to the application	1167
Run the application	1168
See Also	1168
Events in Blazor SplitButton Component.....	1168
List of events supported	1168
How to bind event to Split Button	1168
Icons And Separator in Blazor SplitButton Component.....	1169
Split Button Icons.....	1169
Vertical Button	1170
Separator.....	1171
See Also	1172
Popup Items in Blazor SplitButton Component	1172
Icons	1172
Template	1172
Accessibility in Blazor SplitButton Component.....	1173

ARIA attributes	1173
Keyboard interaction	1173
Styles and Appearances in Blazor SplitButton Component	1174
How To	1175
Create right-to-left Blazor SplitButton Component	1175
Group items in popup in Blazor SplitButton Component	1175
Open a dialog on popup item click in Blazor SplitButton Component	1176
Set the disabled state in Blazor SplitButton Component	1178
Add and Remove Items in Blazor SplitButton Component	1178
Splitter	1179
Getting Started with Blazor Splitter Component	1179
Importing Syncfusion Blazor component in the application	1180
Adding component package to the application	1180
Add SyncfusionBlazor service in Startup.cs	1180
Add Splitter component	1181
Run the application	1181
See Also	1182
Pane Sizing in Blazor Splitter Component	1182
Auto size panes	1182
Fixed pane	1183
Pane Content in Blazor Splitter Component	1185
HTML Markup	1185
Blazor UI components	1186
Plain content	1186
Integrate other Blazor component inside the pane of the Splitter	1187
Pane Settings in the Blazor Splitter Component	1189
Pane visibility	1189
Split Panes in Blazor Splitter Component	1190
Horizontal layout	1190
Vertical layout	1191
Separator	1193
Nested Splitter	1194
Add or remove pane	1196
See Also	1198
Expand and Collapse in Blazor Splitter Component	1198

Collapsible panes	1198
Programmatically control the expand and collapse action	1199
Specify initial state to panes	1201
See Also	1202
Two way Binding in Blazor Splitter Component	1202
See Also	1203
Resize in Blazor Splitter Component.....	1203
Min and Max validation	1204
Prevent resizing.....	1205
Refresh content on resizing	1206
Customize the resize grip and cursor.....	1206
See Also	1207
State Management in Blazor Splitter Component.....	1207
Enabling persistence in Splitter.....	1208
Different Layouts in Blazor Splitter Component.....	1209
Code editor style layout.....	1209
Outlook style layout.....	1211
See Also	1216
Resize Icon Template	1216
Style and appearance in Blazor Splitter Component	1217
Customizing the split bar	1217
Customizing the split bar resize handle	1217
Customizing the split bar arrows	1218
Events in Blazor Splitter Component	1219
Created.....	1219
OnResizeStart.....	1219
OnResizeStop	1219
Resizing	1220
OnCollapse	1220
OnExpand.....	1220
Collapsed.....	1221
Expanded.....	1221
Keyboard interaction	1221
Stock Chart.....	1222
Getting Started with Blazor Stock Chart Component	1222

Importing Syncfusion Blazor Stock Chart component in the application	1222
Adding component package to the application	1223
Adding SyncfusionBlazor service in Startup.cs	1223
Adding Stock Chart component	1223
Populate Stock Chart with Data	1223
Adding Title	1225
Adding Crosshair	1226
Adding Trackball.....	1227
See also	1228
Working with Data in Blazor Stock Chart Component.....	1228
Local Data.....	1228
Remote Data	1230
Entity Framework.....	1231
See Also	1235
Stock Chart Dimensions in Blazor Stock Chart Component.....	1235
Size for Container.....	1235
Size for Stock Chart	1236
Axis Types in Blazor Stock Chart Component	1239
DateTime Axis	1239
Logarithmic Axis	1240
See Also	1241
Axis Customization in Blazor Stock Chart Component	1241
Title	1241
Tick Lines Customization.....	1242
Grid Lines Customization	1243
Multiple Axis	1244
Inversed Axis	1246
Opposed Position.....	1247
Series Types in Blazor Stock Chart Component	1248
Trendlines in Blazor Stock Chart Component	1254
Linear.....	1254
Logarithmic	1256
Exponential	1257
Polynomial	1259
Power	1261

Moving Average	1261
Customization of Trendline.....	1262
Technical Indicators in Blazor Stock Chart Component	1264
Accumulation Distribution	1264
Average True Range (ATR)	1264
Exponential Moving Average (EMA)	1264
Momentum	1264
Moving Average Convergence Divergence (MACD)	1264
Relative Strength Index (RSI).....	1264
Simple Moving Average (SMA)	1264
Stochastic	1264
Triangular Moving Average (TMA).....	1265
Bollinger Band.....	1265
Tooltip in Blazor Stock Chart Component.....	1265
Default Tooltip	1265
Format the Tooltip	1266
Customize the Appearance of Tooltip	1267
Crosshair in Blazor Stock Chart Component	1268
Tooltip for axis	1270
Customization	1271
Add Trackball.....	1273
Period Selector in Blazor Stock Chart Component	1274
Periods	1274
Visibility of period selector	1276
Range Selector in Blazor Stock Chart Component.....	1278
Selecting Range.....	1278
Appearance in Blazor Stock Chart Component.....	1279
Stock Chart Title	1279
Title Customizations.....	1280
Stock Chart Theme	1281
See Also	1282
Print and Export in Blazor Stock Chart Component.....	1282
Disable Export and print	1284
Events in Blazor Stock Chart Component	1285
Loaded.....	1285

OnPointClick.....	1286
PointMoved.....	1287
RangeChange	1287
OnStockChartMouseClicked.....	1288
OnStockChartMouseDown	1289
OnStockChartMouseLeave.....	1289
OnStockChartMouseMove.....	1290
OnStockChartMouseUp	1291
Stock Events in Blazor Stock Chart Component.....	1292
Date.....	1292
Text	1292
Type.....	1292
Background	1292
Description	1292
Tabs.....	1295
Getting Started with Blazor Tabs Component	1295
Importing Syncfusion Blazor component in the application.....	1295
Adding component package to the application.....	1296
Add SyncfusionBlazor service in Startup file.....	1296
Adding Tabs component to the application.....	1297
Run the application	1298
Initialize Tab Content using Template	1298
Two way binding of SelectedItem.....	1299
See Also	1301
Responsive Modes in Blazor Tabs Component.....	1301
Scrollable.....	1301
Popup.....	1304
Header in Blazor Tabs Component	1306
Styles	1306
Icon positions	1309
See Also	1314
Content Render Modes in Blazor Tabs Component	1314
Dynamic rendering.....	1314
On Demand rendering or lazy loading	1316
On initial rendering	1317

Animations in Blazor Tabs Component.....	1319
Localization in Blazor Tabs Component	1322
Blazor Server Side	1322
Blazor Web Assembly.....	1325
Orientation in Blazor Tabs Component	1328
Drag and Drop in Blazor Tabs Component.....	1332
Accessibility in Blazor Tabs Component	1334
ARIA attributes.....	1334
Keyboard interaction	1334
Style and Appearance in Blazor Tabs Component.....	1335
Customizing Tab.....	1335
Customizing the Tab items.....	1335
Customizing Tab's header	1335
Customizing Tab's header icon	1336
Customizing Tab's content.....	1336
Customizing the hover state of Tab control.....	1336
Customizing selected item of Tab control	1336
How To	1337
Add/Remove Tab items in Blazor Tabs Component	1337
Show/Hide Tab item in Blazor Tabs Component	1340
Enable/Disable Tab item in Blazor Tabs Component.....	1341
Add nested tabs in Blazor Tabs Component	1343
Prevent content swipe selection in Blazor Tabs Component	1345
Create wizard in Blazor Tabs Component.....	1346
Style Customization for active Item in Blazor Tabs Component.....	1356
Set state persistence in Blazor Tabs Component.....	1358
Customize the Scrolling distance in Blazor Tabs Component	1360
Prevent reorder active tab in Blazor Tabs Component.....	1362
Find interaction in Blazor Tabs Component.....	1364
TextBox	1366
Getting Started with Blazor TextBox Component.....	1366
Importing Syncfusion Blazor component in the application.....	1366
Adding component package to the application.....	1367
Add SyncfusionBlazor service in Program.cs	1367
Adding TextBox component to the application	1367

Run the application	1368
Adding icons to the TextBox	1368
Floating label.....	1368
See Also	1368
Data Binding in Blazor TextBox Component	1369
One-way binding	1369
Two-way data binding.....	1369
Dynamic value binding.....	1369
Complex data binding	1370
Multiline in Blazor TextBox Component	1370
Create multiline textbox	1370
Implementing floating label	1371
Disable resizing	1372
Sizing in Blazor TextBox Component	1372
Groups in Blazor TextBox Component	1373
With icon and floating label	1374
With clear button and floating label	1376
Multi-line input with floating label	1376
Validation in Blazor TextBox Component	1377
Native Events in Blazor TextBox Component.....	1377
Bind native events to textbox	1377
Pass event data to event handler	1378
List of Native events supported	1378
Events in Blazor TextBox Component	1378
Blur	1378
Created.....	1379
Destroyed.....	1379
Focus	1379
Input.....	1379
ValueChange	1380
ValueChanged	1380
How To	1380
Set the Rounded Corner in Blazor TextBox Component.....	1380
Set the Disabled State in Blazor TextBox Component	1381
Set the Read-only TextBox in Blazor TextBox Component	1381

Change the Floating Label Color of the Blazor TextBox Component	1381
Change the Color of the Text Based on its Value in Blazor TextBox	1382
Customize Background and Text Color in Blazor TextBox Component	1383
Create TextBox component dynamically in Blazor TextBox Component	1384
TimePicker.....	1385
Getting Started with Blazor TimePicker Component.....	1385
Importing Syncfusion Blazor component in the application.....	1385
Adding component package to the application	1386
Add SyncfusionBlazor service in Program.cs	1386
Adding TimePicker component to the application	1387
Run the application	1387
Setting the time format	1387
See Also	1388
Data Binding in Blazor TimePicker Component	1388
One-Way Binding	1388
Two-Way Data Binding.....	1388
Dynamic Value Binding	1389
Time Range in Blazor TimePicker Component.....	1389
Globalization in Blazor TimePicker Component.....	1391
Blazor server side	1391
Blazor WebAssembly	1393
Customize the localized text	1394
Right-To-Left	1395
Native Events in Blazor TimePicker Component.....	1396
Bind native events to TimePicker.....	1396
Pass event data to event handler	1396
List of Native events supported	1397
Strict Mode in Blazor TimePicker Component.....	1397
Accessibility in Blazor TimePicker Component	1399
Keyboard interaction	1399
Events in Blazor TimePicker Component	1400
Blur	1401
ValueChange	1401
OnClose	1401
Created.....	1401

Destroyed.....	1402
Focus	1402
OnItemRender.....	1402
OnOpen.....	1403
How To	1403
CSS Customization in Blazor TimePicker Component.....	1403
Limitations of TimeSpan DataType in Blazor TimePicker Component.....	1405
Toast.....	1406
Getting Started with Blazor Toast Component.....	1406
Importing Syncfusion Blazor component in the application.....	1406
Adding component package to the application.....	1406
Add SyncfusionBlazor service in Startup.cs	1407
Add Toast component.....	1407
Run the application	1408
See Also	1408
Configuring options in Blazor Toast Component.....	1408
Title and content template	1408
Specifying custom target	1409
Close button.....	1409
Progress bar	1409
Newest on top.....	1409
Width and height	1411
Positioning in Blazor Toast Component.....	1414
Predefined.....	1414
Custom	1414
Action Buttons in Blazor Toast Component.....	1418
See Also	1419
Timeout in Blazor Toast Component	1419
Static toast	1421
Template in Blazor Toast Component	1422
Animation in Blazor Toast Component.....	1425
Style and appearance in Blazor Toast Component.....	1427
Customizing the toast title.....	1427
Customizing the toast content.....	1427
Customizing the toast icon.....	1428

Customizing the toast background	1428
Accessibility in Blazor Toast Component	1428
ARIA attributes	1428
Events in Blazor Toast Component	1429
Created	1429
Destroyed	1429
Opened	1430
OnOpen	1430
Closed	1430
OnClick	1430
How To	1431
Show different types of toast in Blazor Toast Component	1431
Show multiple toasts in various positions in Blazor Toast Component	1432
Show toast content dynamically in Blazor Toast Component	1433
Close the toast with click/tap in Blazor Toast Component	1433
Add dynamic template in Blazor Toast Component	1434
Access specific toast in Blazor Toast Component	1435
Prevent toast close with mobile swipe in Blazor Toast Component	1436
Toggle Switch Button	1437
Getting Started with Blazor Toggle Switch Button Component	1437
Importing Syncfusion Blazor component in the application	1437
Adding component package to the application	1438
Add SyncfusionBlazor service in Startup.cs	1438
Adding Toggle Switch Button component to the application	1438
Run the application	1439
Set text on Switch	1439
See Also	1439
Events in Blazor Toggle Switch Button Component	1439
ValueChange Event	1439
Native Events	1440
List of Native events supported	1440
How to bind focus event to Toggle Switch Button	1440
Accessibility in Blazor Toggle Switch Button Component	1440
Keyboard interaction	1441
Styles and Appearances in Blazor Toggle Switch Button Component	1441

How To	1441
Change Size in Blazor Toggle Switch Button Component	1441
Customize the appearance of a Blazor Toggle Switch Button Component	1442
Enable RTL in Blazor Toggle Switch Button Component.....	1446
Change Blazor Toggle Switch Button state using toggle method	1446
Set disabled state in Blazor Toggle Switch Button Component.....	1446
Model binding in Blazor Toggle Switch Button Component.....	1447
Toolbar	1448
Getting Started with Blazor Toolbar Component	1448
Importing Syncfusion Blazor component in the application.....	1448
Adding component package to the application.....	1449
Add SyncfusionBlazor service in Startup file.....	1449
Adding Toolbar component to the application.....	1450
Run the application	1450
See Also	1450
Item Configuration in Blazor Toolbar Component.....	1450
Align	1451
CssClass	1451
Disabled.....	1451
HtmlAttributes	1451
Id	1451
Overflow.....	1452
PrefixIcon	1452
ShowAlwaysInPopup.....	1452
ShowTextOn	1453
SuffixIcon.....	1453
Template	1454
Text	1454
TooltipText	1454
Type.....	1455
Visible	1458
Width	1458
Responsive Mode in Blazor Toolbar Component	1458
Scrollable.....	1459
Popup	1460

MultiRow	1463
Extended	1464
Accessibility in Blazor Toolbar Component.....	1465
ARIA attributes.....	1465
Keyboard interaction	1465
Style and Appearance in Blazor Toolbar Component.....	1466
Customizing Toolbar	1466
Customizing the Toolbar items	1466
Customizing Toolbar's item icon.....	1466
Customizing the hover state of Toolbar control.....	1467
Customizing selected item of Toolbar control.....	1467
How To	1467
Add/Remove Toolbar items in Blazor Toolbar Component.....	1467
Show/Hide Toolbar item in Blazor Toolbar Component.....	1469
Enable/Disable Toolbar item in Blazor Toolbar Component	1469
Set command customization in Blazor Toolbar Component	1470
Set Tooltip to the commands in Blazor Toolbar Component	1471
Set item-wise custom template in Blazor Toolbar Component.....	1472
Add Toggle Button in Blazor Toolbar Component	1472
Customize the Scrolling distance in Blazor Toolbar Component	1477
Tooltip	1478
Getting Started with Blazor Tooltip Component	1478
Importing Syncfusion Blazor component in the application.....	1478
Add Syncfusion Blazor service in Startup.cs (Server-side application)	1481
Add Syncfusion Blazor service in Program.cs (Client-side application)	1482
Adding component package to the application.....	1482
Adding Tooltip component to the application.....	1482
Run the application	1483
See Also	1483
Setting Dimension in Blazor Tooltip Component.....	1483
Height and width.....	1483
Content in Blazor Tooltip Component	1485
Template content.....	1485
Position in Blazor Tooltip Component	1486
Mouse trailing	1487

Setting offset values.....	1488
Open Mode in Blazor Tooltip Component.....	1489
Sticky mode.....	1491
Open or Close tooltip with delay	1492
Customization in Blazor Tooltip Component.....	1493
Tip pointer customization	1493
Tooltip customization	1494
Styles and Appearance in Blazor Tooltip Component	1496
Customizing the tooltip.....	1496
Customizing the tooltip popup	1496
Customizing the tooltip content	1497
Customizing the tooltip arrow tip.....	1497
Customizing the tooltip inner tip	1497
Customizing the tooltip outer tip.....	1498
Template in Blazor Tooltip Component.....	1498
TreeGrid	1500
Getting Started with Blazor TreeGrid Component	1500
Importing Syncfusion Blazor component in the application.....	1500
Add SyncfusionBlazor service in Startup.cs	1501
Adding component package to the application.....	1501
Add Tree Grid component to the application.....	1501
Defining columns	1502
Enable paging.....	1503
Enable sorting	1504
Run the application	1505
See also	1505
Data Binding in Blazor TreeGrid Component.....	1505
List binding.....	1506
Remote Service binding	1512
Entity Framework.....	1520
Custom Binding in Blazor TreeGrid Component.....	1523
Data binding.....	1524
Inject service into Custom Adaptor	1527
Custom adaptor as Component	1529
CRUD operation	1532

Columns in Blazor TreeGrid Component	1535
Complex data binding	1535
Header template	1538
Header text	1541
Format.....	1543
AutoFit specific columns	1547
Reorder	1549
Column resizing.....	1554
Column template	1560
Column type.....	1564
Checkbox column	1565
Responsive columns.....	1567
Controlling treegrid actions	1569
Show or Hide Columns by external button.....	1570
How to render boolean values as checkbox	1572
Rows in Blazor TreeGrid Component.....	1574
Customize rows.....	1574
Styling alternate rows	1576
Row height	1578
Row template.....	1580
Detail template	1585
Drag and drop	1589
Templates in Blazor TreeGrid Component.....	1608
Template ModelType.....	1609
Template Context.....	1611
TreeGridTemplates component.....	1613
Cell in Blazor TreeGrid Component.....	1615
Displaying the HTML content.....	1615
Customize cell styles	1617
Auto wrap	1619
Grid lines	1621
Clip Mode	1623
Editing in Blazor TreeGrid Component	1625
Toolbar with edit option	1627
Cell edit type and its params.....	1629

Cell Edit Template	1632
Edit Modes	1634
Dialog template.....	1642
Adding row position.....	1645
Command column.....	1647
Confirmation messages.....	1651
Entity Framework.....	1653
Default column values on add new.....	1658
Disable editing for particular column	1659
Troubleshoot: Editing works only for first row	1660
Filtering in Blazor TreeGrid Component	1661
Filter hierarchy modes	1662
Initial filter	1663
Filter operators	1665
Filter bar template with custom component.....	1666
Filter menu	1669
Custom component in filter menu.....	1671
Excel like filter	1675
Sorting in Blazor TreeGrid Component.....	1676
Initial sort	1678
Sorting events	1680
Touch interaction	1682
Searching in Blazor TreeGrid Component.....	1682
Initial search	1684
Search operators.....	1686
Search by external button.....	1686
Search specific columns	1688
Paging in Blazor TreeGrid Component.....	1689
Page size mode	1691
Pager with page size dropdown.....	1693
Scrolling in Blazor TreeGrid Component.....	1696
Set width and height.....	1696
Responsive with parent container	1697
Frozen rows and columns	1700
Virtualization in Blazor TreeGrid Component.....	1704

Row Virtualization	1704
Column Virtualization	1708
Aggregate in Blazor TreeGrid Component.....	1711
Built-in aggregate types	1712
Footer aggregate.....	1712
How to format aggregate value	1714
Limitations.....	1717
Globalization in Blazor TreeGrid Component	1717
Localization	1717
Internationalization.....	1724
Right to left (RTL)	1764
Selection in Blazor TreeGrid Component.....	1767
Selection mode	1769
Cell selection	1770
Checkbox selection	1773
Toggle Selection	1776
Select row at initial rendering.....	1778
Get selected row indexes.....	1780
Touch interaction	1782
Toolbar in Blazor TreeGrid Component.....	1782
Built-in toolbar items	1782
Enable/Disable Toolbar Items.....	1784
Excel Export in Blazor TreeGrid Component.....	1787
To customize excel export	1789
PDF Export in Blazor TreeGrid Component.....	1797
To customize PDF export	1799
Print in Blazor TreeGrid Component.....	1810
Page setup.....	1811
Print using an external button	1812
Print the visible page.....	1814
Print large number of columns	1816
Limitations of printing large data.....	1817
Clipboard in Blazor TreeGrid Component.....	1817
Copy to clipboard by external buttons	1819
Copy Hierarchy Modes.....	1821

Paste.....	1823
Accessibility in Blazor TreeGrid Component.....	1825
WAI-ARIA.....	1825
Keyboard navigation	1825
Context Menu in Blazor TreeGrid Component	1827
Custom context menu items.....	1829
Styling and Appearance in Blazor TreeGrid Component	1831
Events in Blazor TreeGrid Component.....	1832
OnActionBegin	1832
OnActionComplete.....	1833
OnActionFailure	1835
Created.....	1836
OnLoad.....	1838
Destroyed.....	1839
OnDataBound.....	1841
DataBound	1842
RowDataBound	1843
DetailDataBound.....	1845
HeaderCellInfo	1846
QueryCellInfo	1848
OnBeginEdit	1849
OnBatchAdd	1851
OnBatchSave	1852
OnBatchDelete.....	1854
OnCellEdit.....	1855
OnCellSave	1857
CellSaved	1858
RowSelecting.....	1859
RowSelected.....	1861
RowDeselecting.....	1862
RowDeselected	1864
CellSelecting.....	1865
CellSelected.....	1866
CellDeselecting.....	1868
CellDeselected.....	1869

OnRecordDoubleClick	1871
OnToolbarClick	1872
CommandClicked	1874
ColumnMenuItemClicked	1875
ContextMenuItemClicked	1877
ContextMenuOpen	1878
OnPdfExport.....	1880
PdfQueryCellInfoEvent.....	1881
OnExcelExport.....	1883
ExcelQueryCellInfoEvent.....	1884
OnResizeStart.....	1886
ResizeStopped.....	1887
Expanding.....	1889
Expanded.....	1890
How To	1891
Show or Hide columns in Dialog editing in Blazor TreeGrid Component	1891
Create custom toolbar with drop-down list in Blazor TreeGrid Component	1894
Customize Column Styles in Blazor TreeGrid Component	1897
Access public methods in Tree Grid in Blazor TreeGrid Component.....	1899
Change datasource dynamically in Blazor TreeGrid Component	1900
Custom control in Tree Grid toolbar in Blazor TreeGrid Component.....	1902
Tree Grid customization in Blazor TreeGrid Component.....	1905
Prevent default Tree Grid action in Blazor TreeGrid Component	1907
Get index value of selected row cell in Blazor TreeGrid Component	1909
Select rows based on certain condition in Blazor TreeGrid Component.....	1910
Customize column menu icon in Blazor TreeGrid Component.....	1912
Calculate column value based on other columns in Blazor TreeGrid	1922
Using dictionary values as datasource in Blazor TreeGrid Component.....	1924
Custom toolbar items in Blazor TreeGrid Component	1926
Render Blazor TreeGrid Component inside the Tab with specific height.....	1929
Single click editing with Batch mode in Blazor TreeGrid Component	1940
Editing with template column in Blazor TreeGrid Component.....	1942
Hide Tree Grid Header in Blazor TreeGrid Component	1944
Display Custom Tooltip in Tree Grid cell in Blazor TreeGrid Component.....	1945
TreeMap.....	1947

Blazor TreeMap Component in Server Side App using Visual Studio	1947
Importing Syncfusion Blazor TreeMap component in the application	1947
Adding component package to the application	1948
Adding SyncfusionBlazor Service in Startup.cs	1948
Adding TreeMap component	1948
Adding labels in TreeMap items	1949
Adding title to TreeMap	1950
Apply color mapping	1951
Enable legend	1952
Enable tooltip	1953
See also	1954
Data Binding in Blazor TreeMap Component	1954
Populate data	1954
Local data	1958
Remote data	1960
Entity Framework	1963
Layout in Blazor TreeMap Component	1967
Types of layout	1967
Rendering direction	1970
Leaf Item in Blazor TreeMap Component	1973
Customization	1973
Label position and format	1974
Gap between the leaf items	1975
Levels in Blazor TreeMap Component	1976
Group path	1976
Gap between groups	1977
Header height and style	1978
Customization	1979
Color Mapping in Blazor TreeMap Component	1980
Range color mapping	1980
Equal color mapping	1981
Desaturation color mapping	1982
Desaturation with multiple colors	1983
Palette color mapping	1984
Color for items excluded from color mapping	1984

Bind the colors to items from the data source	1985
Labels in Blazor TreeMap Component	1986
Formatting labels	1986
Label position	1987
Intersect action	1988
Legend in Blazor TreeMap Component	1989
Types of legend	1989
Position and Alignment	1991
Legend size	1993
Excluded legend items from the color mapping	1995
Hide desired legend items	1996
Hide legend items based on the data source value	1997
Bind legend item text from the data source	1998
Hide duplicate legend items	1999
Positioning based on size	1999
Legend with RTL support	2001
Drill-down in Blazor TreeMap Component	2001
Perform drill-down	2002
On-demand data loading	2003
Breadcrumb support	2004
Initial drill-down	2005
Tooltip in Blazor TreeMap Component	2006
Default tooltip	2006
Customization	2007
Formatting tooltip content	2007
Tooltip template	2008
Selection and Highlight in Blazor TreeMap Component	2009
Selection	2009
Highlight	2010
Print and Export in Blazor TreeMap Component	2011
Print	2011
Export	2012
Accessibility in Blazor TreeMap Component	2013
Globalization in Blazor TreeMap Component	2014
Events in Blazor TreeMap Component	2015

ItemHighlighted	2015
ItemRendering	2015
ItemSelected	2015
LegendRendering	2015
LegendItemRendering.....	2016
Loaded.....	2016
Load.....	2016
OnPrint.....	2016
OnClick	2016
OnDoubleClick.....	2016
DrillCompleted	2017
OnDrillStart	2017
OnItemClick.....	2017
OnItemMove.....	2017
OnRightClick.....	2017
Resizing	2017
TooltipRendering	2018
Methods in Blazor TreeMap Component	2018
Print.....	2018
Export.....	2019
Refresh	2019
SelectItem	2020
TreeView	2021
Getting Started with Blazor TreeView Component	2021
Importing Syncfusion Blazor component in the application.....	2021
Add Syncfusion Blazor service in Startup.cs (Server-side application)	2024
Add Syncfusion Blazor service in Program.cs (Client-side application)	2025
Adding component package to the application	2025
Adding TreeView component to the application.....	2025
Run the application	2026
See Also	2027
Data Binding in Blazor TreeView Component.....	2027
Local data	2027
Remote data.....	2032
Entity Framework.....	2033

CheckBox in Blazor TreeView Component.....	2040
Node Editing in Blazor TreeView Component	2042
MultiSelection in Blazor TreeView Component.....	2044
Drag and Drop in Blazor TreeView Component.....	2046
Multiple-node drag and drop.....	2049
Accessibility in Blazor TreeView Component.....	2051
ARIA attributes	2051
Keyboard interaction	2051
Template in Blazor TreeView Component	2052
How To	2055
Customize the expand and collapse icons in Blazor TreeView Component	2055
Customize the tree nodes based on levels in Blazor TreeView Component	2067
Process the Blazor TreeView node operations using context menu	2070
Check/uncheck on clicking the tree node text in Blazor TreeView	2075
Validate the text when renaming the tree node in Blazor TreeView	2078
Customize treeview as accordion in Blazor TreeView Component	2080
Set tooltip for tree nodes in Blazor TreeView Component.....	2085
Remove the checkbox of the parent node in Blazor TreeView Component	2088
Get iconCss dynamically in Blazor TreeView Component	2090
Get all child nodes through parentID in Blazor TreeView Component.....	2093

Menu Bar

Getting Started with Blazor Menu Bar Component

This section briefly explains about how to include Menu Bar Component in your Blazor server-side application. You can refer [Getting Started with Syncfusion Blazor for Server-side in Visual Studio](#) page for the introduction and configuring the common specifications.

To get start quickly with Menu Bar Component using Blazor, you can check on this video:

{% youtube

"youtube:https://www.youtube.com/watch?v=TBoUetqLnps"%}

Importing Syncfusion Blazor component in the application

1. Install the **Syncfusion.Blazor** NuGet package to the application by using the **NuGet Package Manager**.
2. You can add the client-side style resources through [CDN](#) or from [NuGet](#) package in the element of the `~/Pages/_Host.cshtml` page.

Please ensure to check the **Include prerelease** option.

ASPX-CS

```
<head>
<link href="_content/Syncfusion.Blazor/styles/bootstrap4.css"
rel="stylesheet" />
@*<link href="https://cdn.syncfusion.com/blazor/{{ site.blazorversion
}}/bootstrap4.css" rel="stylesheet" />*@
</head>
```

For Internet Explorer 11 kindly refer the polyfills. Refer the [documentation](#) for more information.

ASPX-CS

```
<head>
<link href="_content/Syncfusion.Blazor/styles/bootstrap4.css"
rel="stylesheet" />
<script
src="https://github.com/Daddoon/Blazor.Polyfill/releases/download/3.0.1/blaz
or.polyfill.min.js"></script>
</head>
```

Adding component package to the application

Open `/_Imports.razor` file and import the **Syncfusion.Blazor.Navigations** package.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
```

Add SyncfusionBlazor service in Startup.cs

Open the **Startup.cs** file and add services required by Syncfusion components.

Add **services.AddSyncfusionBlazor()** method in the ConfigureServices function as follows.

C#

```
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void ConfigureServices(IServiceCollection services)
        {
            ....
            ....
            services.AddSyncfusionBlazor();
        }
    }
}
```

To enable custom client side resource loading from CRG or CDN. You need to disable resource loading by **AddSyncfusionBlazor(true)** and load the scripts in the HEAD element of the **~/Pages/_Host.cshtml** page.

ASPX-CS

```
<head>
<environment include="Development">
<script src="https://cdn.syncfusion.com/blazor/{{ site.blazorversion
}}/syncfusion-blazor.min.js">
</script>
</environment>
</head>
```

Adding Menu Bar component to the application

Now, add the Blazor Menu Bar component in **razor** page in the **Pages** folder. For example, the Menu Bar component is added in the **~/Pages/Index.razor** page.

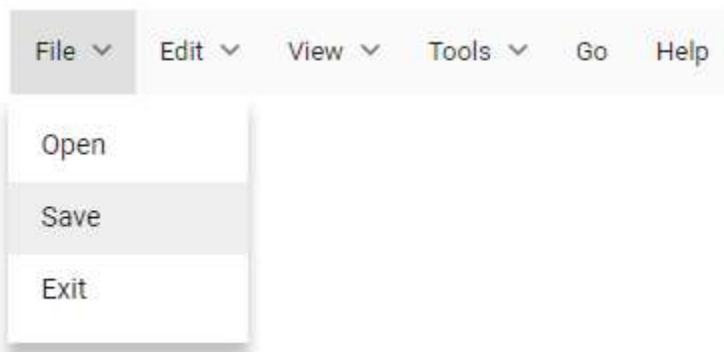
ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<SfMenu TValue="MenuItem">
<MenuItems>
<MenuItem Text="File">
<MenuItems>
<MenuItem Text="Open"></MenuItem>
<MenuItem Text="Save"></MenuItem>
<MenuItem Text="Exit"></MenuItem>
</MenuItems>
</MenuItem>
<MenuItem Text="Edit">
<MenuItems>
<MenuItem Text="Cut"></MenuItem>
<MenuItem Text="Copy"></MenuItem>
<MenuItem Text="Paste"></MenuItem>
</MenuItems>
</MenuItem>
<MenuItem Text="View">
```

```
<MenuItems>
<MenuItem Text="Toolbars"></MenuItem>
<MenuItem Text="Zoomr"></MenuItem>
<MenuItem Text="Full Screen"></MenuItem>
</MenuItems>
</MenuItem>
<MenuItem Text="Tools">
<MenuItems>
<MenuItem Text="Spelling & Grammar"></MenuItem>
<MenuItem Text="Customize"></MenuItem>
<MenuItem Text="Options"></MenuItem>
</MenuItems>
</MenuItem>
<MenuItem Text="Go"></MenuItem>
<MenuItem Text="Help"></MenuItem>
</MenuItems>
</SfMenu>
```

Run the application

After successful compilation of your application, simply press F5 to run the application. The Blazor Menu Bar component will render in the web browser as shown below



See Also

- [Getting Started with Syncfusion Blazor for Client-Side in .NET Core CLI](#)
- [Getting Started with Syncfusion Blazor for Server-Side in Visual Studio](#)
- [Getting Started with Syncfusion Blazor for Server-Side in .NET Core CLI](#)

Accessibility in Blazor Menu Bar Component

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.

The Menu Bar provides a built-in compliance with WAI-ARIA specifications. The WAI-ARIA support is achieved using the attributes such as `aria-orientation`, `aria-label`, `aria-expanded`, `aria-disabled`, and `aria-haspopup` applied for Menu Bar item in Menu Bar. It helps the people with disabilities by providing

information about the widget for assistive technology in the screen readers. The Menu Bar component contains the `menubar`, `menu`, and `menuitem` roles.

| Properties | Functionality |

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

| `menubar` | This role will be specified for root Menu Bar. |

| `menu` | This role will be specified for an item that have sub menu. |

| `menuitem` | This role will be specified for an item that do not have sub menus. |

| `aria-haspopup` | Indicates the availability and type of interactive popup element. |

| `aria-expanded` | Indicates whether the subtree can be expanded or collapsed, and indicates whether its current state can be expanded or collapsed. |

| `aria-orientation` | Indicates whether the orientation is horizontal or vertical. The default orientation is horizontal. |

| `aria-label` | Indicates the Menu Bar item text. |

| `aria-disabled` | Indicates the state of Menu Bar item whether it is disabled. |

User interaction with keyboard

<!-- markdownlint-disable MD033 -->

Keyboard shortcuts	Actions
Esc	Closes the sub menu that contains focus and returns focus to the parent element.
Enter	Opens the sub menu if focused menu item has sub menu, and places focus on its first item or activates the item and closes the sub menu.
Up	Navigates up or to the previous menu item.
Down	Navigates down or to the next menu item.
Left	Closes the current sub menu and navigates to the parent menu.
Right	Navigates and open the next sub menu.
Home	Focuses the first item.
End	Focuses the last item.

DataSource Binding and Custom Items in Blazor Menu Bar Component

The Menu Bar supports data source bindings such as data source that can be structured as `Self-referential` data.

Self-referential data

Menu Bar can be populated from self-referential data structure that contains data with `ParentId` mapping.

In the following example, the `id`, `pId`, and `text` columns from self-referential data have been mapped to the `ItemId`, `ParentId`, and `Text` fields, respectively.

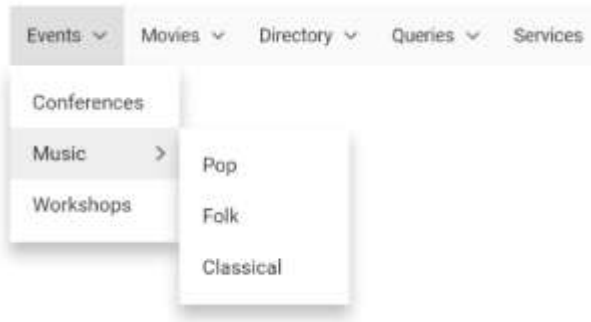
ASPX-CS

```

@using Syncfusion.Blazor.Navigations
<SfMenu Items="@MenuItems">
<MenuFieldSettings ItemId="Id" Text="Text"
ParentId="ParentId"></MenuFieldSettings>
</SfMenu>
@code {
public List<CustomMenuItem> MenuItems = new List<CustomMenuItem>
{
new CustomMenuItem{ Id = "parent1", Text = "Events" },
new CustomMenuItem{ Id = "parent2", Text = "Movies" },
new CustomMenuItem{ Id = "parent3", Text = "Directory" },
new CustomMenuItem{ Id = "parent4", Text = "Queries", ParentId = "null" },
new CustomMenuItem{ Id = "parent5", Text = "Services", ParentId = "null" },
new CustomMenuItem{ Id = "parent6", Text = "Conferences", ParentId =
"parent1" },
new CustomMenuItem{ Id = "parent7", Text = "Music", ParentId = "parent1" },
new CustomMenuItem{ Id = "parent8", Text = "Workshops", ParentId = "parent1"
},
new CustomMenuItem{ Id = "parent9", Text = "Now Showing", ParentId =
"parent2" },
new CustomMenuItem{ Id = "parent10", Text = "Coming Soon", ParentId =
"parent2" },
new CustomMenuItem{ Id = "parent10", Text = "Media Gallery", ParentId =
"parent3" },
new CustomMenuItem{ Id = "parent11", Text = "Newsletters", ParentId =
"parent3" },
new CustomMenuItem{ Id = "parent12", Text = "Our Policy", ParentId =
"parent4" },
new CustomMenuItem{ Id = "parent13", Text = "Site Map", ParentId = "parent4"
},
new CustomMenuItem{ Id = "parent14", Text = "Pop", ParentId = "parent7" },
new CustomMenuItem{ Id = "parent15", Text = "Folk", ParentId = "parent7" },
new CustomMenuItem{ Id = "parent16", Text = "Classical", ParentId =
"parent7" }
};
public class CustomMenuItem
{
public string Id { get; set; }
public string Text { get; set; }
public string ParentId { get; set; }
}
}

```

Output be like



Custom Menu Bar Items

To customize Menu Bar items in your application, set your customized template using [MenuTemplates](#). In the following example, the Menu Bar has been rendered with customized Menu Bar items.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
@inject Microsoft.AspNetCore.Components.NavigationManager UriHelper;
<div class="menu-control">
<SfMenu Items="@data">
<MenuFieldSettings Text="Value" Children="Options"></MenuFieldSettings>
<MenuTemplates TValue="CategoryModel">
<Template>
@{
var MenuItems = context;
if (MenuItems.Value != null && MenuItems.Count == null && MenuItems.Url ==
null)
{
<div>@MenuItems.Value</div>
}
else if (MenuItems.Value != null)
{
<div style="width: 100%;display: flex;justify-content: space-between;">
@{
if (MenuItems.Url != null)
{

}
<span style="width:100%;">@MenuItems.Value</span>
if (MenuItems.Count != null)
{
<span class="e-badge e-badge-success">@MenuItems.Count</span>
}
}
</div>
}
else
{
<div tabindex="0" class="e-card">
<div class="e-card-header">
<div class="e-card-header-caption">
<div class="e-card-header-title">About Us</div>
</div>
</div>
}
</div>
```



```

<div class="e-card-content">
@MenuItems.About
</div>
<div class="e-card-actions">
<input type="button" class="e-btn e-outline" style="pointer-events: auto;"
value="Read More" />
</div>
</div>
}
}
</Template>
</MenuTemplates>
</SfMenu>
</div>
@code{
private List<CategoryModel> data = new List<CategoryModel>
{
    new CategoryModel {
        Value = "Products",
        Options = new List<CategoryModel>
        {
            new CategoryModel { Value= "JavaScript", Url= "javascript" },
            new CategoryModel { Value= "Angular", Url= "angular" },
            new CategoryModel { Value= "ASP.NET Core", Url= "core" },
            new CategoryModel { Value= "ASP.NET MVC", Url= "mvc" }
        }
    },
    new CategoryModel {
        Value = "Services",
        Options = new List<CategoryModel>
        {
            new CategoryModel { Value= "Application Development", Count= "1200+" },
            new CategoryModel { Value= "Maintenance & Support", Count= "3700+" },
            new CategoryModel { Value= "Quality Assurance" },
            new CategoryModel { Value= "Cloud Integration", Count= "900+" }
        }
    },
    new CategoryModel {
        Value = "About Us",
        Options = new List<CategoryModel>
        {
            new CategoryModel {
                Id = "about",
                About = "We are on a mission to provide world-class best software solutions
for web, mobile and desktop platforms. Around 900+ applications are desgined
and delivered to our customers to make digital & strengthen their
businesses."
            }
        }
    },
    new CategoryModel { Value = "Careers" },
    new CategoryModel { Value = "Sign In" }
};
private class CategoryModel
{
    public List<CategoryModel> Options { get; set; }
    public string Value { get; set; }
}

```

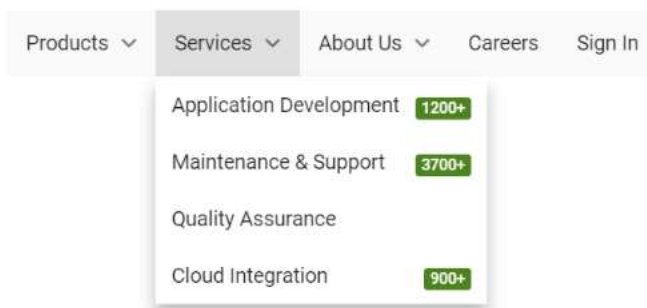
```
public string Url { get; set; }
public string Count { get; set; }
public string About { get; set; }
public string Id { get; set; }
}
}
<style>
.menu-control {
margin-top: 45px;
text-align: center;
}
/* Common ul & li styles */
.mobile .e-menu-container ul.e-ul,
.e-menu-container ul.e-ul {
padding: 0;
}
.mobile .e-menu-container ul.e-ul .e-menu-item,
.e-menu-container ul.e-ul .e-menu-item {
display: flex;
padding: 0 10px;
outline-color: transparent;
}
/** Avatar customization */
.e-menu-container ul .e-menu-item .e-avatar {
background-color: inherit;
font-size: 8px;
margin-right: 8px;
align-self: center;
width: auto;
overflow: visible;
}
.mobile .e-menu-container ul .e-menu-item .e-avatar {
font-size: 10px;
}
/** Badge customization */
.e-menu-container ul .e-menu-item .e-badge {
margin-left: 10px;
align-self: center;
overflow: visible;
}
/** Card customization */
.e-menu-container ul.e-ul .e-menu-item .e-card {
width: 290px;
font-size: inherit;
background-color: inherit;
border-color: transparent;
}
.mobile .e-menu-container ul.e-ul .e-menu-item .e-card {
width: 320px;
}
.e-menu-container ul.e-ul .e-menu-item .e-card .e-card-content {
white-space: normal;
color: inherit;
padding-top: 0;
text-align: justify;
font-size: inherit;
}
}
```

```

.e-menu-container ul.e-ul .e-menu-item#about {
height: auto;
padding: 0;
}
.e-menu-container ul.e-ul .e-menu-item#about.e-focused {
background-color: transparent;
outline-color: transparent;
pointer-events: none;
}
.e-menu-container .e-ul .e-menu-item {
height: 36px;
line-height: 36px;
}
</style>

```

Output be like



Icons and submenu Items in Blazor Menu Bar Component

Icons

The menu item contains an icon/image in it to provide a visual representation of an action. To place the icon on a menu item, set the [IconCss](#) property with the required icon CSS. By default, the icon is positioned at the left of the menu item. In the following sample, the icons of **File** and **Edit** menu items and **Open**, **Save**, **Cut**, **Copy**, and **Paste** sub menu items are added using the [IconCss](#) property.

ASPX-CS

```

<SfMenu TValue="MenuItem">
  <MenuItems>
    <MenuItem Text="File" IconCss="e-icons e-file">
      <MenuItems>
        <MenuItem Text="Open" IconCss="e-icons e-open"></MenuItem>
        <MenuItem Text="Save" IconCss="e-icons e-save"></MenuItem>
        <MenuItem Separator="true"></MenuItem>
        <MenuItem Text="Exit"></MenuItem>
      </MenuItems>
    </MenuItem>
    <MenuItem Text="Edit" IconCss="e-icons e-edit">
      <MenuItems>
        <MenuItem Text="Cut" IconCss="e-icons e-cut"></MenuItem>
        <MenuItem Text="Copy" IconCss="e-icons e-copy"></MenuItem>
        <MenuItem Text="Paste" IconCss="e-icons e-paste"></MenuItem>
      </MenuItems>
    </MenuItem>
  </MenuItems>
</SfMenu>

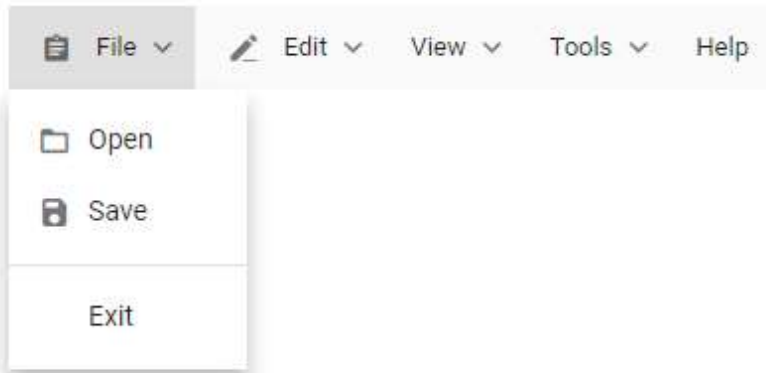
```

```

</MenuItem>
<MenuItem Text="View">
<MenuItems>
<MenuItem Text="Toolbars">
<MenuItems>
<MenuItem Text="Menu Bar"></MenuItem>
<MenuItem Text="Bookmarks Toolbar"></MenuItem>
<MenuItem Text="Customize"></MenuItem>
</MenuItems>
</MenuItem>
</MenuItems>
</MenuItem>
<MenuItem Text="Zoom">
<MenuItems>
<MenuItem Text="Zoom In"></MenuItem>
<MenuItem Text="Zoom Out"></MenuItem>
<MenuItem Text="Reset"></MenuItem>
</MenuItems>
</MenuItem>
<MenuItem Text="Tools">
<MenuItems>
<MenuItem Text="Spelling & Grammar"></MenuItem>
<MenuItem Text="Customize"></MenuItem>
<MenuItem Separator="true"></MenuItem>
<MenuItem Text="Options"></MenuItem>
</MenuItems>
</MenuItem>
<MenuItem Text="Help"></MenuItem>
</MenuItems>
</SfMenu>
<style>
.e-file::before {
content: '\e7cb';
}
.e-edit::before {
content: '\e78f';
}
.e-open::before {
content: '\e70f';
}
.e-save::before {
content: '\e74d';
}
.e-cut::before {
content: '\e73f';
}
.e-copy::before {
content: '\e77b';
}
.e-paste::before {
content: '\e739';
}
</style>

```

Output be like



Navigation

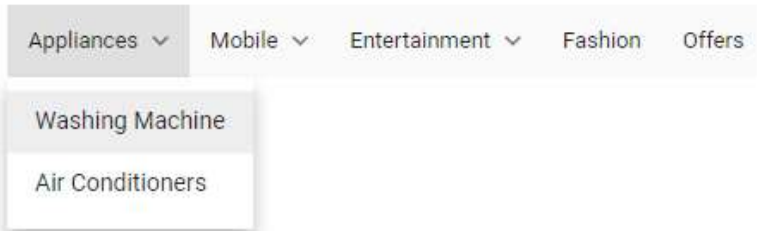
Navigation in Menu Bar is used to navigate to the other web page when a Menu Bar item is clicked. It can be achieved by providing a link to the Menu Bar item using the [Url](#) property. In the following sample, the Navigation URL is added to sub menu Bar items using the [Url](#) property.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<SfMenu TValue="MenuItem">
  <MenuItems>
    <MenuItem Text="Appliances">
      <MenuItems>
        <MenuItem Text="Washing Machine"
          Url="https://www.google.com/search?q=washing+machine"></MenuItem>
        <MenuItem Text="Air Conditioners"
          Url="https://www.google.com/search?q=air+conditioners"></MenuItem>
      </MenuItems>
    </MenuItem>
    <MenuItem Text="Mobile">
      <MenuItems>
        <MenuItem Text="Headphones"
          Url="https://www.google.com/search?q=headphones"></MenuItem>
        <MenuItem Text="Memory Cards"
          Url="https://www.google.com/search?q=memory+cards"></MenuItem>
        <MenuItem Text="Power Banks"
          Url="https://www.google.com/search?q=power+banks"></MenuItem>
      </MenuItems>
    </MenuItem>
    <MenuItem Text="Entertainment">
      <MenuItems>
        <MenuItem Text="Televisions"
          Url="https://www.google.com/search?q=televisions"></MenuItem>
        <MenuItem Text="Home Theatres"
          Url="https://www.google.com/search?q=home+theatres"></MenuItem>
        <MenuItem Text="Gaming Laptops"
          Url="https://www.google.com/search?q=gaming+laptops"></MenuItem>
      </MenuItems>
    </MenuItem>
    <MenuItem Text="Fashion"
          Url="https://www.google.com/search?q=fashion"></MenuItem>
    <MenuItem Text="Offers"
          Url="https://www.google.com/search?q=offers"></MenuItem>
  </MenuItems>
</SfMenu>
```

```
</MenuItem>
</SfMenu>
```

Output be like



Multilevel nesting

The Menu Bar supports multiple level nesting, and it can be achieved by mapping the [Items](#) property inside the parent `MenuItems`. In the following sample, three-level nesting of Menu Bar has been provided.

ASPX-CS

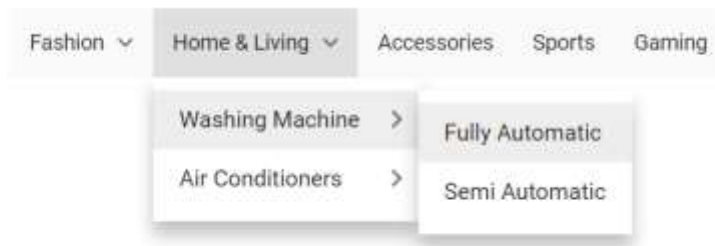
```
@using Syncfusion.Blazor.Navigations
<SfMenu TValue="MenuItem">
  <MenuItems>
    <MenuItem Text="Fashion">
      <MenuItems>
        <MenuItem Text="Men Fashion">
          <MenuItems>
            <MenuItem Text="Personal Care">
              <MenuItems>
                <MenuItem Text="Trimmers"></MenuItem>
                <MenuItem Text="Shavers"></MenuItem>
              </MenuItems>
            </MenuItem>
          </MenuItems>
        </MenuItem>
        <MenuItem Text="Clothing">
          <MenuItems>
            <MenuItem Text="shirts"></MenuItem>
            <MenuItem Text="Jackets"></MenuItem>
            <MenuItem Text="TrackSuits"></MenuItem>
          </MenuItems>
        </MenuItem>
      </MenuItems>
    </MenuItem>
    <MenuItem Text="Women Fashion">
      <MenuItems>
        <MenuItem Text="Clothing">
          <MenuItems>
            <MenuItem Text="Salwars"></MenuItem>
            <MenuItem Text="Kurtas"></MenuItem>
            <MenuItem Text="Sarees"></MenuItem>
          </MenuItems>
        </MenuItem>
        <MenuItem Text="Jewellery">
          <MenuItems>
```

```

<MenuItem Text="Nosepin"></MenuItem>
<MenuItem Text="Anklelets"></MenuItem>
</MenuItems>
</MenuItem>
</MenuItems>
</MenuItem>
</MenuItems>
</MenuItem>
<MenuItem Text="Home & Living">
<MenuItems>
<MenuItem Text="Washing Machine">
<MenuItems>
<MenuItem Text="Fully Automatic"></MenuItem>
<MenuItem Text="Semi Automatic"></MenuItem>
</MenuItems>
</MenuItem>
<MenuItem Text="Air Conditioners">
<MenuItems>
<MenuItem Text="Inverter AC"></MenuItem>
<MenuItem Text="Split AC"></MenuItem>
</MenuItems>
</MenuItem>
</MenuItems>
</MenuItem>
<MenuItem Text="Accessories"></MenuItem>
<MenuItem Text="Sports"></MenuItem>
<MenuItem Text="Gaming"></MenuItem>
</MenuItems>
</SfMenu>

```

Output be like



Use Case Scenarios in Blazor Menu Bar Component

Scrollable Menu Bar

The Menu Bar component supports horizontal and vertical scrolling to render large Menu Bars and sub menus in an adaptive way. This can be achieved by enabling the [EnableScrolling](#) property and by restricting the corresponding Menu Bar/Sub Menu Bar size.

ASPX-CS

```

@using Syncfusion.Blazor.Navigations
<SfMenu TValue="MenuItem" EnableScrolling="true" CssClass="e-scrollable-menu">
<MenuItems>
<MenuItem Text="File">

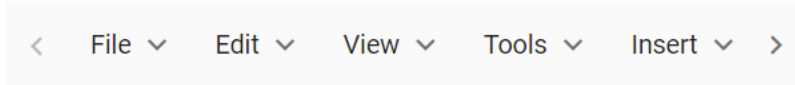
```

```

<MenuItems>
  <MenuItem Text="Open"></MenuItem>
  <MenuItem Text="Save"></MenuItem>
  <MenuItem Text="Save As"></MenuItem>
  <MenuItem Text="Print"></MenuItem>
  <MenuItem Text="Share"></MenuItem>
  <MenuItem Text="Info"></MenuItem>
  <MenuItem Text="Exit"></MenuItem>
</MenuItems>
</MenuItem>
<MenuItem Text="Edit">
  <MenuItems>
    <MenuItem Text="Cut"></MenuItem>
    <MenuItem Text="Copy"></MenuItem>
    <MenuItem Text="Paste"></MenuItem>
  </MenuItems>
</MenuItem>
<MenuItem Text="View">
  <MenuItems>
    <MenuItem Text="Toolbars"></MenuItem>
    <MenuItem Text="Zoom"></MenuItem>
    <MenuItem Text="Full Screen"></MenuItem>
  </MenuItems>
</MenuItem>
<MenuItem Text="Tools">
  <MenuItems>
    <MenuItem Text="Spelling & Grammar"></MenuItem>
    <MenuItem Text="Customize"></MenuItem>
    <MenuItem Text="Options"></MenuItem>
  </MenuItems>
</MenuItem>
<MenuItem Text="Insert">
  <MenuItems>
    <MenuItem Text="Comment"></MenuItem>
    <MenuItem Text="Links"></MenuItem>
    <MenuItem Text="Table"></MenuItem>
  </MenuItems>
</MenuItem>
<MenuItem Text="Design"></MenuItem>
<MenuItem Text="Go"></MenuItem>
<MenuItem Text="Layout"></MenuItem>
<MenuItem Text="Help"></MenuItem>
</MenuItems>
</SfMenu>
<style>
.e-scrollable-menu:not(.e-menu-popup) {
width: 400px;
}
</style>

```

Output be like



Hamburger Menu

The following example demonstrates the use case of [HamburgerMenu](#) mode.

ASPX-CS

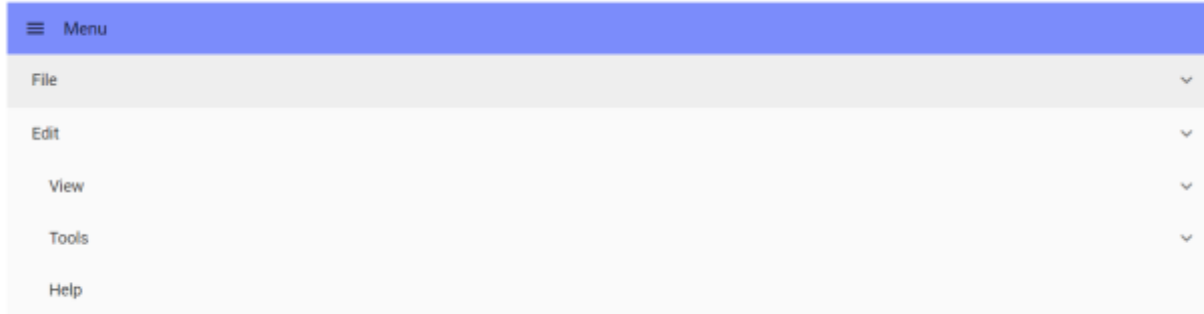
```
@using Syncfusion.Blazor.Navigations;
<SfMenu TValue="MenuItem" HamburgerMode="true" ShowItemOnClick="true">
  <MenuItems>
    <MenuItem Text="File">
      <MenuItems>
        <MenuItem Text="Open"></MenuItem>
        <MenuItem Text="Save"></MenuItem>
        <MenuItem Text="Exit"></MenuItem>
      </MenuItems>
    </MenuItem>
    <MenuItem Text="Edit">
      <MenuItems>
        <MenuItem Text="Cut"></MenuItem>
        <MenuItem Text="Copy"></MenuItem>
        <MenuItem Text="Paste"></MenuItem>
      </MenuItems>
    </MenuItem>
    <MenuItem Text="View">
      <MenuItems>
        <MenuItem Text="Toolbars">
          <MenuItems>
            <MenuItem Text="Menu Bar"></MenuItem>
            <MenuItem Text="Bookmarks Toolbar"></MenuItem>
            <MenuItem Text="Customize"></MenuItem>
          </MenuItems>
        </MenuItem>
        <MenuItem Text="Zoom">
          <MenuItems>
            <MenuItem Text="Zoom In"></MenuItem>
            <MenuItem Text="Zoom Out"></MenuItem>
            <MenuItem Text="Reset"></MenuItem>
          </MenuItems>
        </MenuItem>
        <MenuItem Text="Full Screen"></MenuItem>
        <MenuItem Text="Tools">
          <MenuItems>
            <MenuItem Text="Spelling & Grammar"></MenuItem>
            <MenuItem Text="Customize"></MenuItem>
            <MenuItem Separator="true"></MenuItem>
            <MenuItem Text="Options"></MenuItem>
          </MenuItems>
        </MenuItem>
        <MenuItem Text="Help"></MenuItem>
      </MenuItems>
    </MenuItem>
  </MenuItems>
```

```

</SfMenu>
<style>
.e-menu-header {
width: 100%;
background-color: #7b8cfb;
}
</style>

```

Output be like



Mobile View

The following example demonstrates the use case of Menu Bar in Mobile mode with hamburger.

ASPX-CS

```

@using Syncfusion.Blazor.Navigations;
<div class="menu-control">
<div id='layoutcontainer' class="deviceLayout">
<div class="speaker">
<div class="camera"></div>
</div>
<div class="layout">
<div id="container">
<SfMenu TValue="MenuItem" HamburgerMode="true" ShowItemOnClick="true">
<MenuItems >
<MenuItem Text="File">
<MenuItems>
<MenuItem Text="Open"></MenuItem>
<MenuItem Text="Save"></MenuItem>
<MenuItem Separator="true"></MenuItem>
<MenuItem Text="Exit"></MenuItem>
</MenuItems>
</MenuItem>
<MenuItem Text="Edit">
<MenuItems>
<MenuItem Text="Cut"></MenuItem>
<MenuItem Text="Copy"></MenuItem>
<MenuItem Text="Paste"></MenuItem>
</MenuItems>
</MenuItem>
<MenuItem Text="View">
<MenuItems>
<MenuItem Text="Toolbars">

```

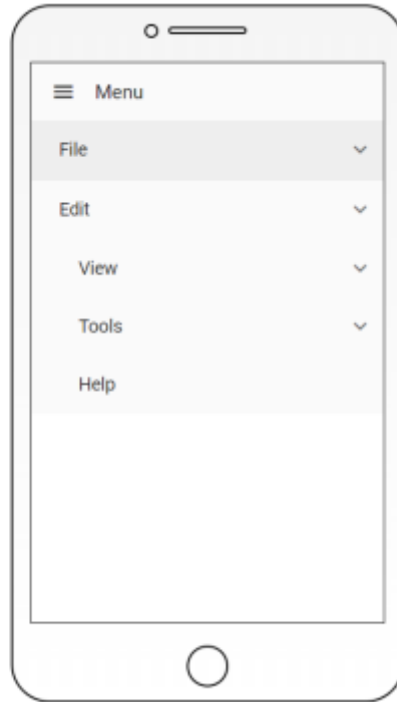
```

<MenuItems>
<MenuItem Text="Menu Bar"></MenuItem>
<MenuItem Text="Bookmarks Toolbar"></MenuItem>
<MenuItem Text="Customize"></MenuItem>
</MenuItems>
</MenuItem>
</MenuItems>
</MenuItem>
<MenuItem Text="Zoom">
<MenuItems>
<MenuItem Text="Zoom In"></MenuItem>
<MenuItem Text="Zoom Out"></MenuItem>
<MenuItem Text="Reset"></MenuItem>
</MenuItems>
</MenuItem>
<MenuItem Text="Full Screen"></MenuItem>
<MenuItem Text="Tools">
<MenuItems>
<MenuItem Text="Spelling & Grammar"></MenuItem>
<MenuItem Text="Customize"></MenuItem>
<MenuItem Separator="true"></MenuItem>
<MenuItem Text="Options"></MenuItem>
</MenuItems>
</MenuItem>
<MenuItem Text="Help"></MenuItem>
</MenuItems>
</SfMenu>
</div>
</div>
<div class="outerButton"> </div>
</div>
</div>
<style>
.deviceLayout #menu {
-ms-overflow-style: none;
scrollbar-width: none;
height: 363px;
}
.deviceLayout #menu::-webkit-scrollbar {
width: 0;
}
.menu-control {
text-align: center;
}
#layoutcontainer:not(.deviceLayout) {
margin-top: 45px;
}
.deviceLayout {
line-height: initial;
border: 1px solid black;
width: 285px;
height: 505px;
margin: auto;
margin-bottom: 15px;
border-radius: 28px;
position: relative;
background-image: linear-gradient(to top, #ffffff, #f5f5f5);

```

```
}  
.deviceLayout .speaker {  
border: 1px solid black;  
border-radius: 5px;  
width: 20%;  
height: 5px;  
margin: 15px auto 0px auto;  
position: relative;  
}  
.deviceLayout .outerButton {  
width: 30px;  
height: 30px;  
border: 1px solid black;  
border-radius: 50%;  
position: absolute;  
bottom: calc(0% + 10px);  
left: calc(50% - 15px);  
}  
.deviceLayout .camera {  
position: absolute;  
left: calc(-15% - 10px);  
top: -100%;  
width: 10px;  
height: 10px;  
border-radius: 50%;  
border: 1px solid black;  
}  
.deviceLayout .layout {  
border: 1px solid black;  
margin: 20px 13px 0px 13px;  
}  
.layout #container {  
height: 405px;  
background-color: white;  
overflow: hidden;  
}  
</style>
```

Output be like



Styles and Appearances in Blazor Menu Bar Component

To modify the Menu appearance, you need to override the default CSS of Menu component. Please find the list of CSS classes and its corresponding section in Menu 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-menu-container | To customize the menu wrapper |

| .e-menu-container.e-menu-popup | To customize the menu popup |

| .e-menu-container .e-ul .e-menu-item | To customize the menu items |

| .e-menu-container .e-ul .e-menu-item.e-focused | To customize the menu items on focus |

| .e-menu-container ul .e-menu-item .e-caret::before | To customize the menu items caret icon |

| .e-menu-container .e-ul .e-menu-item.e-selected | To customize selected menu item |

How To

Change Orientation in Blazor Menu Bar Component

Orientation in menu items can be changed horizontally or vertically using the [Orientation](#) property. By default, it is horizontally aligned.

ASPX-CS

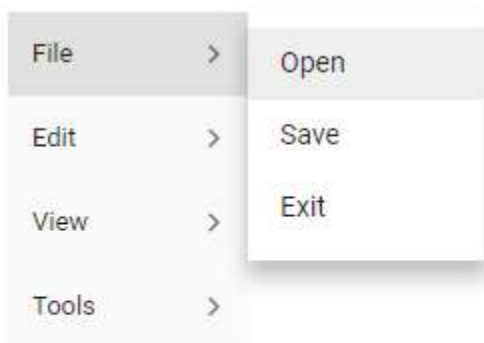
```
<SfMenu TValue="MenuItem"
Orientation="Syncfusion.Blazor.Navigations.Orientation.Vertical">
  <MenuItems>
    <MenuItem Text="File">
    <MenuItems>
```

```

<MenuItem Text="Open"></MenuItem>
<MenuItem Text="Save"></MenuItem>
<MenuItem Text="Exit"></MenuItem>
</MenuItems>
</MenuItem>
<MenuItem Text="Edit">
<MenuItems>
<MenuItem Text="Cut"></MenuItem>
<MenuItem Text="Copy"></MenuItem>
<MenuItem Text="Paste"></MenuItem>
</MenuItems>
</MenuItem>
<MenuItem Text="View">
<MenuItems>
<MenuItem Text="Toolbar"></MenuItem>
<MenuItem Text="Sidebar"></MenuItem>
<MenuItem Text="Full Screen"></MenuItem>
</MenuItems>
</MenuItem>
<MenuItem Text="Tools">
<MenuItems>
<MenuItem Text="Spelling & Grammer"></MenuItem>
<MenuItem Text="Customize"></MenuItem>
<MenuItem Text="Options"></MenuItem>
</MenuItems>
</MenuItem>
</MenuItems>
</SfMenu>

```

Output be like



Customize Menu Bar Items in Blazor Menu Bar Component

[Add or Remove Menu Items](#)

Menu items can be added or removed directly by using Add or Remove methods.

In the following example, the **Corporate** menu items are added and the **Company** items are removed from menu.

ASPX-CS

```

@using Syncfusion.Blazor.Navigations
<SfMenu Items="@MenuItems" @ref="MenuObj">
<MenuEvents TValue="DataList" Created="Created"></MenuEvents>
<MenuFieldSettings Text="@nameof(DataList.Data)"
Children="@nameof(DataList.SubDatas)"></MenuFieldSettings>
</SfMenu>
@code{
SfMenu<DataList> MenuObj;
public List<DataList> MenuItems = new List<DataList>
{
    new DataList{ Data = "Company", SubDatas = new List<DataList>{
        new DataList{ Data= "Overview" },
        new DataList{ Data= "About" },
        new DataList{ Data= "Careers" }}
    },
    new DataList{ Data = "Services", SubDatas = new List<DataList>{
        new DataList{ Data= "Consulting" },
        new DataList{ Data= "Education" },
        new DataList{ Data= "Health" }}
    },
    new DataList{ Data = "Products", SubDatas = new List<DataList>{
        new DataList{ Data = "Hardware" },
        new DataList{ Data = "Software" }}
    },
    new DataList{ Data = "Contact Us" }
};
public class DataList
{
    public string Data { get; set; }
    public List<DataList> SubDatas { get; set; }
}
public void Created()
{
    this.MenuObj.Items.Add(new DataList
    {
        Data = "Corporate",
        SubDatas = new List<DataList>
        {
            new DataList{ Data = "Leadership"},
            new DataList{ Data = "Vision"}
        }
    });
    var item = this.MenuObj.Items.First(x => x.Data == "Company");
    this.MenuObj.Items.Remove(item);
}
}

```

Output be like

Services ▾ Products ▾ Contact Us Corporate ▾

Enable or Disable Menu Items

You can enable and disable the menu items using the [Disabled](#) property in Menu items. To disable menuitems, set the **Disabled** property to true and vice-versa.

In the following example, the Directory header item, Conferences, and Music sub menu items are disabled.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
@using Syncfusion.Blazor.Buttons
<SfMenu TValue="MenuItem">
  <MenuEvents TValue="MenuItem" Created="@Created"></MenuEvents>
  <MenuItems>
    <MenuItem Text="Events">
      <MenuItems>
        <MenuItem Text="Conferences" Disabled="@disableItem"></MenuItem>
        <MenuItem Text="Music" Disabled="@disableItem"></MenuItem>
        <MenuItem Text="Workshops"></MenuItem>
      </MenuItems>
    </MenuItem>
    <MenuItem Text="Movies">
      <MenuItems>
        <MenuItem Text="Now Showing"></MenuItem>
        <MenuItem Text="Coming Soon"></MenuItem>
      </MenuItems>
    </MenuItem>
    <MenuItem Text="Directory" Disabled="@disableItem">
      <MenuItems>
        <MenuItem Text="Newsletter"></MenuItem>
        <MenuItem Text="Media Gallery"></MenuItem>
      </MenuItems>
    </MenuItem>
    <MenuItem Text="Queries">
      <MenuItems>
        <MenuItem Text="Our Policy"></MenuItem>
        <MenuItem Text="Site Map"></MenuItem>
      </MenuItems>
    </MenuItem>
    <MenuItem Text="Services"></MenuItem>
  </MenuItems>
</SfMenu>
<SfButton @onclick="EnableItems">Enable all items</SfButton>
@code {
  private bool disableItem;
  public void Created()
  {
    disableItem = true;
  }
  public void EnableItems()
  {
    disableItem = false;
  }
}
```

Output be like



Show or Hide Menu Items

You can show or hide the menu items using the [Hidden](#) property in Menu items. To hide the menu items, set the `Hidden` property to true and vice-versa.

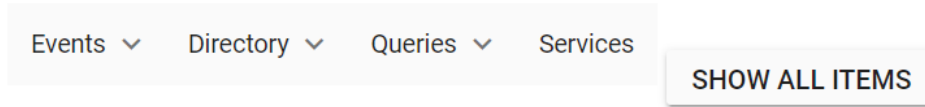
In the following example, the Movies header item, Workshops, and Music sub menu items are hidden in menu.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
@using Syncfusion.Blazor.Buttons
<SfMenu TValue="MenuItem">
  <MenuEvents TValue="MenuItem" Created="@Created"></MenuEvents>
  <MenuItems>
    <MenuItem Text="Events">
    </MenuItem>
    <MenuItem Text="Conferences"></MenuItem>
    <MenuItem Text="Music" Hidden="@hideItem"></MenuItem>
    <MenuItem Text="Workshops" Hidden="@hideItem"></MenuItem>
  </MenuItems>
  <MenuItem Text="Movies" Hidden="@hideItem">
    <MenuItems>
      <MenuItem Text="Now Showing"></MenuItem>
      <MenuItem Text="Coming Soon"></MenuItem>
    </MenuItems>
  </MenuItem>
  <MenuItem Text="Directory">
    <MenuItems>
      <MenuItem Text="Newsletter"></MenuItem>
      <MenuItem Text="Media Gallery"></MenuItem>
    </MenuItems>
  </MenuItem>
  <MenuItem Text="Queries">
    <MenuItems>
      <MenuItem Text="Our Policy"></MenuItem>
      <MenuItem Text="Site Map"></MenuItem>
    </MenuItems>
  </MenuItem>
  <MenuItem Text="Services"></MenuItem>
</SfMenu>
<SfButton @onclick="ShowItems">Show all items</SfButton>
@code {
  private bool hideItem;
  public void Created()
  {
    hideItem = true;
  }
  public void ShowItems()
```

```
{
  hideItem = false;
}
```

Output be like



Customize Menu Bar Using Events in Blazor Menu Bar Component

The Menu Bar provides a set of events to enable users to customize it.

The available events are [OnOpen](#), [OnClose](#), [Closed](#), [Opened](#), and [ItemSelected](#).

ASPX-CS

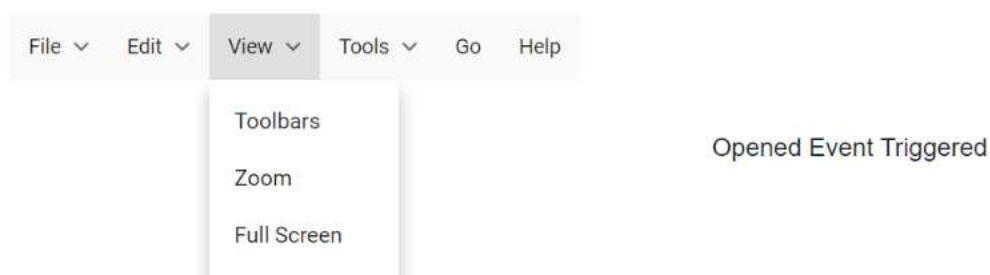
```
@using Syncfusion.Blazor.Navigations
<SfMenu TValue="MenuItem">
  <MenuItems>
    <MenuItem Text="File">
      <MenuItems>
        <MenuItem Text="Open"></MenuItem>
        <MenuItem Text="Save"></MenuItem>
        <MenuItem Text="Exit"></MenuItem>
      </MenuItems>
    </MenuItem>
    <MenuItem Text="Edit">
      <MenuItems>
        <MenuItem Text="Cut"></MenuItem>
        <MenuItem Text="Copy"></MenuItem>
        <MenuItem Text="Paste"></MenuItem>
      </MenuItems>
    </MenuItem>
    <MenuItem Text="View">
      <MenuItems>
        <MenuItem Text="Toolbars"></MenuItem>
        <MenuItem Text="Zoomr"></MenuItem>
        <MenuItem Text="Full Screen"></MenuItem>
      </MenuItems>
    </MenuItem>
    <MenuItem Text="Tools">
      <MenuItems>
        <MenuItem Text="Spelling & Grammar"></MenuItem>
        <MenuItem Text="Customize"></MenuItem>
        <MenuItem Text="Options"></MenuItem>
      </MenuItems>
    </MenuItem>
    <MenuItem Text="Go"></MenuItem>
    <MenuItem Text="Help"></MenuItem>
  </MenuItems>
```

```

<MenuEvents TValue="MenuItem" OnOpen="onOpen" OnClose="onClose"
Opened="opened" Closed="closed" ItemSelected="itemSelected"></MenuEvents>
</SfMenu>
<div id="preview">@eventName Event Triggered</div>
@code{
private string eventName = "No";
private void onOpen()
{
this.eventName = "OnOpen";
}
private void onClose()
{
this.eventName = "OnClose";
}
private void opened()
{
this.eventName = "Opened";
}
private void closed()
{
this.eventName = "Closed";
}
private void itemSelected()
{
this.eventName = "ItemSelected";
}
}
<style>
#preview {
float: right;
padding: 0 350px 0 0;
}
</style>

```

Output be like



Menu Bar with Rounded Corner in Blazor Menu Bar Component

The rounded corner can be achieved by using the [CssClass](#) property. Add a custom class to the menu bar component and customize it using the `border-radius` CSS property. For more information, refer to the [styles](#) specified in the below sample.

ASPX-CS

```

@using Syncfusion.Blazor.Navigations
<SfMenu TValue="MenuItem" CssClass="e-rounded-menu">

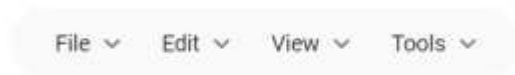
```

```

<MenuItems>
  <MenuItem Text="File">
    <MenuItems>
      <MenuItem Text="Open"></MenuItem>
      <MenuItem Text="Save"></MenuItem>
      <MenuItem Text="Exit"></MenuItem>
    </MenuItems>
  </MenuItem>
  <MenuItem Text="Edit">
    <MenuItems>
      <MenuItem Text="Cut"></MenuItem>
      <MenuItem Text="Copy"></MenuItem>
      <MenuItem Text="Paste"></MenuItem>
    </MenuItems>
  </MenuItem>
  <MenuItem Text="View">
    <MenuItems>
      <MenuItem Text="Toolbar"></MenuItem>
      <MenuItem Text="Sidebar"></MenuItem>
      <MenuItem Text="Full Screen"></MenuItem>
    </MenuItems>
  </MenuItem>
  <MenuItem Text="Tools">
    <MenuItems>
      <MenuItem Text="Spelling & Grammer"></MenuItem>
      <MenuItem Text="Customize"></MenuItem>
      <MenuItem Text="Options"></MenuItem>
    </MenuItems>
  </MenuItem>
</MenuItems>
</SfMenu>
<style>
/* Styles to achieve rounder corner in menu */
.e-menu-container.e-rounded-menu:not(.e-menu-popup),
.e-menu-container.e-rounded-menu .e-menu {
border-radius: 20px;
}
/* Increased the menu component left and right padding for better rounded
corner UI */
.e-menu-container.e-rounded-menu ul.e-menu {
padding: 0 14px;
}
</style>

```

Output be like



Open Sub Menu on Menu Item Click in Blazor Menu Bar Component

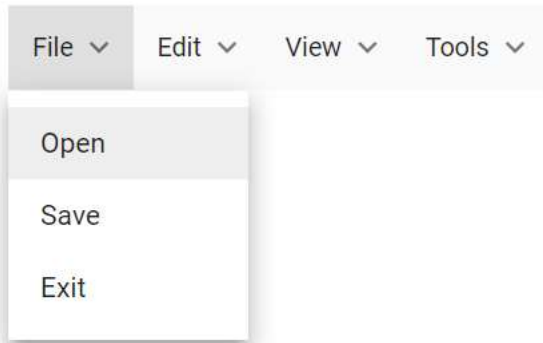
This section explains about how to open a sub menu on Menu item click. This can be achieved by using `ShowItemOnClick` property of the Menu.

In the following sample, Sub Menu will open while clicking `File` menu.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<SfMenu TValue="MenuItem" ShowItemOnClick="true">
  <MenuItems>
    <MenuItem Text="File">
      <MenuItems>
        <MenuItem Text="Open"></MenuItem>
        <MenuItem Text="Save"></MenuItem>
        <MenuItem Text="Exit"></MenuItem>
      </MenuItems>
    </MenuItem>
    <MenuItem Text="Edit">
      <MenuItems>
        <MenuItem Text="Cut"></MenuItem>
        <MenuItem Text="Copy"></MenuItem>
        <MenuItem Text="Paste"></MenuItem>
      </MenuItems>
    </MenuItem>
    <MenuItem Text="View">
      <MenuItems>
        <MenuItem Text="Toolbar"></MenuItem>
        <MenuItem Text="Sidebar"></MenuItem>
        <MenuItem Text="Full Screen"></MenuItem>
      </MenuItems>
    </MenuItem>
    <MenuItem Text="Tools">
      <MenuItems>
        <MenuItem Text="Spelling & Grammer"></MenuItem>
        <MenuItem Text="Customize"></MenuItem>
        <MenuItem Text="Options"></MenuItem>
      </MenuItems>
    </MenuItem>
  </MenuItems>
</SfMenu>
```

Output be like



Right to Left in Blazor Menu Bar Component

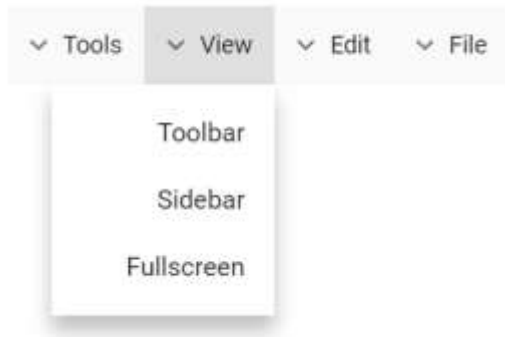
Menu Bar component has RTL support. This can be achieved by setting [EnableRtl](#) as `true`.

The following example illustrates how to enable right-to-left support in Menu Bar component.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<SfMenu TValue="MenuItem" EnableRtl="true">
  <MenuItems>
    <MenuItem Text="File">
      <MenuItems>
        <MenuItem Text="Open"></MenuItem>
        <MenuItem Text="Save"></MenuItem>
        <MenuItem Text="Exit"></MenuItem>
      </MenuItems>
    </MenuItem>
    <MenuItem Text="Edit">
      <MenuItems>
        <MenuItem Text="Cut"></MenuItem>
        <MenuItem Text="Copy"></MenuItem>
        <MenuItem Text="Paste"></MenuItem>
      </MenuItems>
    </MenuItem>
    <MenuItem Text="View">
      <MenuItems>
        <MenuItem Text="Toolbar"></MenuItem>
        <MenuItem Text="Sidebar"></MenuItem>
        <MenuItem Text="Full Screen"></MenuItem>
      </MenuItems>
    </MenuItem>
    <MenuItem Text="Tools">
      <MenuItems>
        <MenuItem Text="Spelling & Grammer"></MenuItem>
        <MenuItem Text="Customize"></MenuItem>
        <MenuItem Text="Options"></MenuItem>
      </MenuItems>
    </MenuItem>
  </MenuItems>
</SfMenu>
```

Output be like



MultiSelect Dropdown

Getting Started with Blazor MultiSelect Dropdown Component

This section briefly explains about how to include a [Blazor MultiSelect](#) Component in your Blazor client-side application. You can refer to the [Getting Started with Syncfusion Blazor for Client-side in Visual Studio 2019](#) page for introduction and configure the common specifications.

Importing Syncfusion Blazor component in the application

- Install `Syncfusion.Blazor.DropDowns` NuGet package to the application by using the `NuGet Package Manager`.

Please ensure to check the `Include prerelease` option for our Beta release.

- You can add the client-side resources through [CDN](#) or from [NuGet](#) package in the **HEAD** element of the `~/Pages/_Host.cshtml` page.

HTML

```
<head>
<link href="_content/Syncfusion.Blazor.Themes/bootstrap4.css"
rel="stylesheet" />
@*<link
href="https://cdn.syncfusion.com/blazor/{{version}}/styles/{{theme}}.css"
rel="stylesheet" />*@
</head>
```

For Internet Explorer 11 kindly refer the polyfills. Refer the [documentation](#) for more information.

HTML

```
<head>
<link href="_content/Syncfusion.Blazor.Themes/bootstrap4.css"
rel="stylesheet" />
<script
src="https://github.com/Daddoon/Blazor.Polyfill/releases/download/3.0.1/blaz
or.polyfill.min.js"></script>
</head>
```

Adding component package to the application

Open `~/_Imports.razor` file and import the `Syncfusion.Blazor.DropDowns` package.

ASPX-CS

```
@using Syncfusion.Blazor.DropDowns
```

Add SyncfusionBlazor service in Startup.cs

Open the **Startup.cs** file and add services required by Syncfusion components using **services.AddSyncfusionBlazor()** method. Add this method in the **ConfigureServices** function as follows.

CSHARP

```
using Syncfusion.Blazor;
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void ConfigureServices(IServiceCollection services)
        {
            ....
            ....
            services.AddSyncfusionBlazor();
        }
    }
}
```

To enable custom client side resource loading from CRG or CDN. You need to disable resource loading by `AddSyncfusionBlazor(true)` and load the scripts in the **HEAD** element of the `~/Pages/_Host.cshtml` page.

HTML

```
<head>
<script src="https://cdn.syncfusion.com/blazor/{ site.blazorversion
}/syncfusion-blazor.min.js"></script>
</head>
```

Adding MultiSelect component to the application

To initialize the MultiSelect component add the below code to your `Index.razor` view page which is present under `~/Pages` folder.

ASPX-CS

```
<SfMultiSelect TValue="string[]" TItem="string" Placeholder='First
Name'></SfMultiSelect>
```

Output be like below

First Name

Binding data source

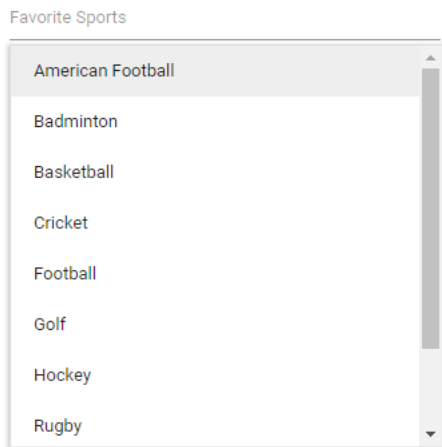
After initialization, populate the MultiSelect with data using the [DataSource](#) property. Here, an array of string values is passed to the MultiSelect component. `TItem` specifies the type of the Datasource in MultiSelect.

The following example illustrates the output in your browser.

ASPX-CS

```
<SfMultiSelect TValue="string[]" TItem="Games" Placeholder="Favorite Sports"
DataSource="@LocalData">
<MultiSelectFieldSettings Text="Text" Value="ID"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
public class Games
{
public string ID { get; set; }
public string Text { get; set; }
}
List<Games> LocalData = new List<Games> {
new Games() { ID= "Game1", Text= "American Football" },
new Games() { ID= "Game2", Text= "Badminton" },
new Games() { ID= "Game3", Text= "Basketball" },
new Games() { ID= "Game4", Text= "Cricket" },
new Games() { ID= "Game5", Text= "Football" },
new Games() { ID= "Game6", Text= "Golf" },
new Games() { ID= "Game7", Text= "Hockey" },
new Games() { ID= "Game8", Text= "Rugby" },
new Games() { ID= "Game9", Text= "Snooker" },
new Games() { ID= "Game10", Text= "Tennis" },
};
}
```

The output will be as follows.



Configure the popup list

By default, the width of the popup list automatically adjusts according to the MultiSelect input element's width, and the height auto adjust's according to the height of the popup list items.

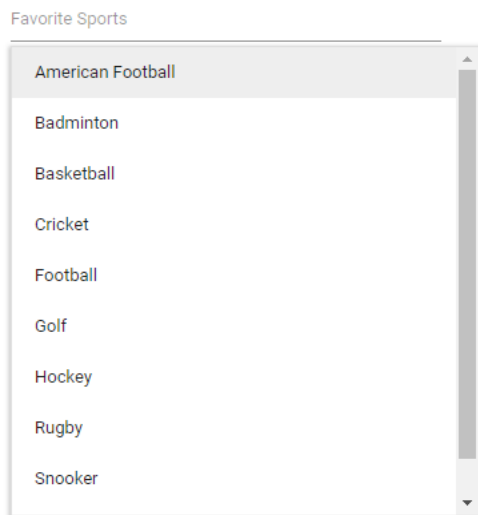
The height and width of the popup list can also be customized using the [PopupHeight](#) and [PopupWidth](#) properties respectively.

In the following sample, popup list's width and height are configured.

ASPX-CS

```
<SfMultiSelect TValue="string[]" TItem="Games" Placeholder="Favorite Sports"
PopupHeight="350px" PopupWidth="350px" DataSource="@LocalData">
  <MultiSelectFieldSettings Text="Text" Value="ID"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
public class Games
{
public string ID { get; set; }
public string Text { get; set; }
}
List<Games> LocalData = new List<Games> {
new Games() { ID= "Game1", Text= "American Football" },
new Games() { ID= "Game2", Text= "Badminton" },
new Games() { ID= "Game3", Text= "Basketball" },
new Games() { ID= "Game4", Text= "Cricket" },
new Games() { ID= "Game5", Text= "Football" },
new Games() { ID= "Game6", Text= "Golf" },
new Games() { ID= "Game7", Text= "Hockey" },
new Games() { ID= "Game8", Text= "Rugby"},
new Games() { ID= "Game9", Text= "Snooker" },
new Games() { ID= "Game10", Text= "Tennis"},
};
}
```

The output will be as follows.



You can also explore our [Blazor MultiSelect Dropdown example](#) that shows how to present and manipulate data.

See Also

- [Getting Started with Syncfusion Blazor for Client-Side in .NET Core CLI](#)
- [Getting Started with Syncfusion Blazor for Server-side in Visual Studio](#)
- [Getting Started with Syncfusion Blazor for Server-Side in .NET Core CLI](#)

Data Binding in Blazor MultiSelect Dropdown Component

Data binding can be achieved by using the **bind-Value** attribute and its supports string, int, Enum, DateTime, bool types. If component value has been changed, it will affect the all places where we bind the variable for the **bind-value** attribute.

ASPX-CS

```
@using Syncfusion.Blazor.DropDowns
@foreach (var SelectedValue in MultiVal)
{
    <p>MultiSelect value is:<strong>@SelectedValue</strong></p>
}
<SfMultiSelect Placeholder="e.g. Australia" @bind-Value="@MultiVal"
DataSource="@Country">
    <MultiSelectFieldSettings Value="Name"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
    public string[] MultiVal { get; set; } = new string[] { };
    public class Countries
    {
        public string Name { get; set; }
        public string Code { get; set; }
    }
    List<Countries> Country = new List<Countries>
    {
        new Countries() { Name = "Australia", Code = "AU" },
        new Countries() { Name = "Bermuda", Code = "BM" },
        new Countries() { Name = "Canada", Code = "CA" },
        new Countries() { Name = "Cameroon", Code = "CM" },
    };
}
```

Data Source in Blazor MultiSelect Dropdown Component

The MultiSelect 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 MultiSelect 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 | **int 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. |

When binding complex data to the MultiSelect, 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 object

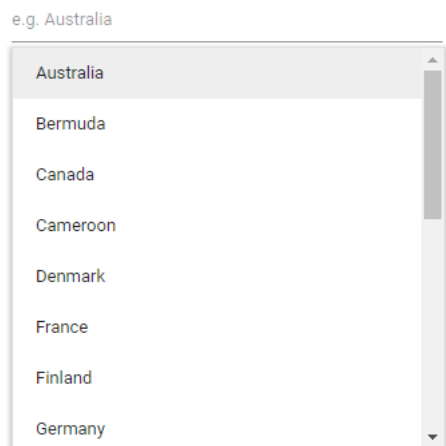
The MultiSelect can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [Fields](#) property.

In the following example, **Name** column from complex data have been mapped to the **Value** field.

ASPX-CS

```
@using Syncfusion.Blazor.DropDowns
<SfMultiSelect TValue="string[]" TItem="Countries" Placeholder="e.g.
Australia" DataSource="@Country">
  <MultiSelectFieldSettings Text="Name"
  Value="Code"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
public class Countries
{
public string Name { get; set; }
public string Code { get; set; }
}
List<Countries>Country = new List<Countries>
{
new Countries() { Name = "Australia", Code = "AU" },
new Countries() { Name = "Bermuda", Code = "BM" },
new Countries() { Name = "Canada", Code = "CA" },
new Countries() { Name = "Cameroon", Code = "CM" },
new Countries() { Name = "Denmark", Code = "DK" },
new Countries() { Name = "France", Code = "FR" },
new Countries() { Name = "Finland", Code = "FI" },
new Countries() { Name = "Germany", Code = "DE" },
new Countries() { Name = "Greenland", Code = "GL" },
new Countries() { Name = "Hong Kong", Code = "HK" },
new Countries() { Name = "India", Code = "IN" },
new Countries() { Name = "Italy", Code = "IT" },
new Countries() { Name = "Japan", Code = "JP" },
new Countries() { Name = "Mexico", Code = "MX" },
new Countries() { Name = "Norway", Code = "NO" },
new Countries() { Name = "Poland", Code = "PL" },
new Countries() { Name = "Switzerland", Code = "CH" },
new Countries() { Name = "United Kingdom", Code = "GB" },
new Countries() { Name = "United States", Code = "US" },
};
}
```

The output will be as follows.



2. Array of complex object

The MultiSelect 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.

ASPX-CS

```
@using Syncfusion.Blazor.DropDowns
<SfMultiSelect TValue="string[]" TItem="Complex" Placeholder="e.g. Select a
country" DataSource="@LocalData">
<MultiSelectFieldSettings Text="Country.CountryID"
Value="Code.ID"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
public List<Complex> LocalData { get; set; } = new Complex().GetData();
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;
}
}
}

```

The output will be as follows.



Binding remote data

The MultiSelect 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 MultiSelect.

The following sample displays the first 6 contacts from **Customers** table of the **Northwind** Data Service.

ASPX-CS

```

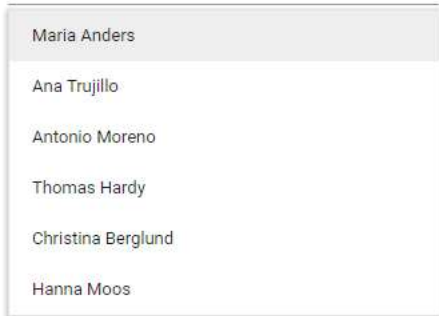
@using Syncfusion.Blazor.Data
@using Syncfusion.Blazor.DropDowns
<SfMultiSelect TValue="string[]" TItem="OrderDetails" Placeholder="Select a
customer" Query="@Query">
<SfDataManager
Url="https://services.odata.org/V4/Northwind/Northwind.svc/Orders"
Adaptor="Adaptors.ODataV4Adaptor" CrossDomain=true></SfDataManager>
<MultiSelectFieldSettings Text="CustomerID"
Value="OrderID"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
public Query Query = new Query().Select(new List<string> { "CustomerID",
"OrderID" }).Take(6).RequiresCount();
public class OrderDetails
{
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; }
}
}
```

The output will be as follows.

Select a customer



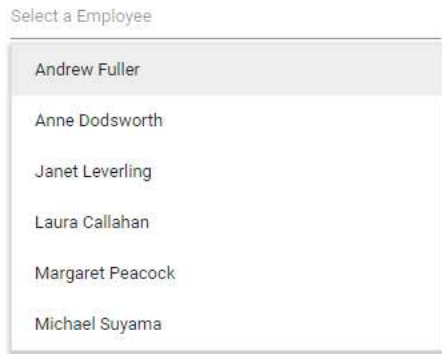
Web API Adaptor

Use the **WebApiAdaptor** to bind MultiSelect with Web API created using OData.

ASPX-CS

```
@using Syncfusion.Blazor.Data
@using Syncfusion.Blazor.DropDowns
<SfMultiSelect TValue="string[]" Placeholder="Select a Employee"
Query="@Query">
<SfDataManager Url="https://ej2services.syncfusion.com/production/web-
services/api/Employees" Adaptor="Adaptors.WebApiAdaptor"
CrossDomain=true></SfDataManager>
<MultiSelectFieldSettings Text="FirstName"
Value="EmployeeID"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
public Query Query = new Query();
public class EmployeeData
{
public int EmployeeID { get; set; }
public string FirstName { get; set; }
public string Designation { get; set; }
public string Country { get; set; }
}
}
```

The output will be as follows.



Custom Adaptor

The [SfDataManager](#) has custom adaptor support which allows you to perform manual operations on the data. This can be utilized for implementing custom data binding and editing operations in the MultiSelect component.

For implementing custom data binding in MultiSelect, the **DataAdaptor** class is used. This abstract class acts as a base class for the custom adaptor.

The **DataAdaptor** abstract class has both synchronous and asynchronous method signatures which can be overridden in the custom adaptor. Following are the method signatures present in this class,

CSHARP

```
public abstract class DataAdaptor
{
    /// <summary>
    /// Performs data Read operation synchronously.
    /// </summary>
    public virtual object Read(DataManagerRequest dataManagerRequest, string key
    = null)
    /// <summary>
    /// Performs data Read operation asynchronously.
    /// </summary>
    public virtual Task<object> ReadAsync(DataManagerRequest dataManagerRequest,
    string key = null)
}
```

The custom data binding can be performed in the MultiSelect component by providing the custom adaptor class and overriding the Read or ReadAsync method of the DataAdaptor abstract class.

The following sample code demonstrates implementing custom data binding using custom adaptor,

ASPX-CS

```
<SfMultiSelect TValue="string[]" TItem="Orders">
  <SfDataManager AdaptorInstance="@typeof(CustomAdaptor) "
  Adaptor="Adaptors.CustomAdaptor"></SfDataManager>
  <MultiSelectFieldSettings Value="CustomerID"></MultiSelectFieldSettings>
</SfMultiSelect>
@code{
public class Orders
{
    public Orders() { }
```



```

public Orders(int OrderID, string CustomerID)
{
    this.OrderID = OrderID;
    this.CustomerID = CustomerID;
}
public int OrderID { get; set; }
public string CustomerID { get; set; }
}
public class CustomAdaptor : DataAdaptor
{
    public static List<OrdersDetails> order = OrdersDetails.GetAllRecords();
    public override object Read(DataManagerRequest dm, string key = null)
    {
        IEnumerable<OrdersDetails> DataSource = order;
        if (dm.Search != null && dm.Search.Count > 0)
        {
            DataSource = DataOperations.PerformSearching(DataSource, dm.Search);
            //Search
        }
        if (dm.Sorted != null && dm.Sorted.Count > 0) //Sorting
        {
            DataSource = DataOperations.PerformSorting(DataSource, dm.Sorted);
        }
        if (dm.Where != null && dm.Where.Count > 0) //Filtering
        {
            DataSource = DataOperations.PerformFiltering(DataSource, dm.Where,
            dm.Where[0].Operator);
        }
        int count = DataSource.Cast<OrdersDetails>().Count();
        if (dm.Skip != 0)
        {
            DataSource = DataOperations.PerformSkip(DataSource, dm.Skip);
            //Paging
        }
        if (dm.Take != 0)
        {
            DataSource = DataOperations.PerformTake(DataSource, dm.Take);
        }
        return dm.RequiresCounts ? new DataResult() { Result = DataSource, Count =
        count } : (object)DataSource;
    }
}

```

Offline mode

To avoid post back for every action, set the MultiSelect to load all data on initialization and make the actions process in client-side. To enable this behaviour, use the **Offline** property of **DataManager**.

The following example for remote data binding and enabled offline mode.

ASPX-CS

```

@using Syncfusion.Blazor.Data
@using Syncfusion.Blazor.DropDowns
<SfMultiSelect TValue="string[]" TItem="EmployeeData" Placeholder="Select a
Employee" Query="@Query">

```

```

<SfDataManager Url="https://ej2services.syncfusion.com/production/web-
services/api/Employees" Adaptor="Adaptors.WebApiAdaptor" CrossDomain=true
Offline=true></SfDataManager>
<MultiSelectFieldSettings Text="FirstName"
Value="EmployeeID"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
public Query Query = new Query();
public class EmployeeData
{
public int EmployeeID { get; set; }
public string FirstName { get; set; }
public string Designation { get; set; }
public string Country { get; set; }
}
}

```

The output will be as follows.

Select a Employee



ValueTuple data binding

You can bind [ValueTuple](#) data to the MultiSelect component. The following code helps you to get a string value from the enumeration data by using [ValueTuple](#).

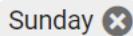
CSHARP

```

@using Syncfusion.Blazor.DropDowns;
<SfMultiSelect TItem="(DayOfWeek, string)" Width="250px"
TValue="DayOfWeek[]"
DataSource="@ (Enum.GetValues<DayOfWeek>().Select(e => (e, e.ToString())))">
<MultiSelectFieldSettings Value="Item1" Text="Item2" />
</SfMultiSelect>

```

The output will shown as follows,



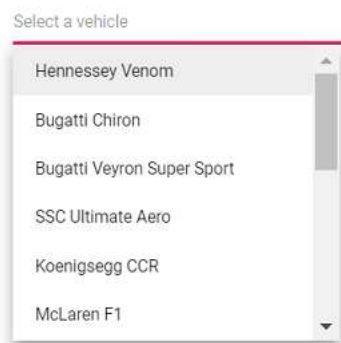
Binding ExpandoObject

You can bind [ExpandoObject](#) data to the MultiSelect component. The following example `ExpandoObject` is bound to the collection of vehicles data.

CSHARP

```
@using Syncfusion.Blazor.DropDowns
@using System.Dynamic
<SfMultiSelect TItem="ExpandoObject" TValue="string[]" PopupHeight="230px"
Placeholder="Select a vehicle" DataSource="@VehicleData">
<MultiSelectFieldSettings Text="Text" Value="ID"></MultiSelectFieldSettings>
</SfMultiSelect>
@code{
public List<ExpandoObject> VehicleData { get; set; } = new
List<ExpandoObject>();
protected override void OnInitialized()
{
VehicleData = Enumerable.Range(1, 15).Select((x) =>
{
dynamic d = new ExpandoObject();
d.ID = (1000 + x).ToString();
d.Text = (new string[] { "Hennessey Venom", "Bugatti Chiron", "Bugatti
Veyron Super Sport", "SSC Ultimate Aero", "Koenigsegg CCR", "McLaren F1",
"Aston Martin One- 77", "Jaguar XJ220", "McLaren P1", "Ferrari LaFerrari",
"Mahindra Jaguar", "Hyundai Toyota", "Jeep Volkswagen", "Tata Maruti
Suzuki", "Audi Mercedes Benz" }[x - 1]);
return d;
}).Cast<ExpandoObject>().ToList<ExpandoObject>();
}
}
```

The output will shown as follows,



Binding DynamicObject

You can bind [DynamicObject](#) data to the MultiSelect component. The following example [DynamicObject](#) is bound to the collection of customers data.

CSHARP

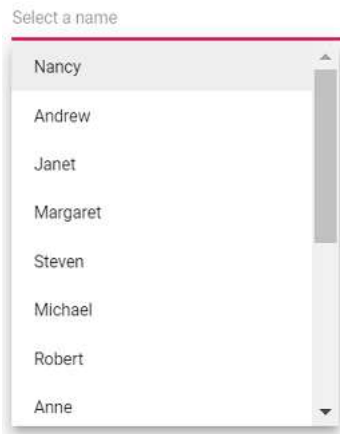
```
@using Syncfusion.Blazor.DropDowns
@using System.Dynamic
<SfMultiSelect TValue="string[]" TItem="DynamicDictionary"
Placeholder="Select a name" DataSource="@Orders">
<MultiSelectFieldSettings Text="CustomerName"
Value="CustomerName"></MultiSelectFieldSettings>
</SfMultiSelect>
@code{
    public List<DynamicDictionary> Orders = new List<DynamicDictionary>() { };
    protected override void OnInitialized()
    {
        Orders = Enumerable.Range(1, 15).Select((x) =>
        {
            dynamic d = new DynamicDictionary();
            d.OrderID = 1000 + x;
            d.CustomerName = (new string[] { "Nancy", "Andrew", "Janet", "Margaret",
            "Steven", "Michael", "Robert", "Anne", "Nige", "Fuller", "Dodsworth",
            "Leverling", "Callahan", "Suyama", "Davolio" }[x - 1]);
            return d;
        }).Cast<DynamicDictionary>().ToList<DynamicDictionary>();
    }
    public class DynamicDictionary : System.Dynamic.DynamicObject
    {
        Dictionary<string, object> dictionary = new Dictionary<string, object>();
        public override bool TryGetMember(GetMemberBinder binder, out object result)
        {
            string name = binder.Name;
            return dictionary.TryGetValue(name, out result);
        }
        public override bool TrySetMember(SetMemberBinder binder, object value)
        {
            dictionary[binder.Name] = value;
            return true;
        }
    }
    //The GetDynamicMemberNames method of DynamicObject class must be overridden
    and return the property names to perform data operation and editing while
    using DynamicObject.
```

```

public override System.Collections.Generic.IEnumerable<string>
GetDynamicMemberNames()
{
    return this.dictionary?.Keys;
}
}
}

```

The output will shown as follows,



Binding ObservableCollection

You can bind [ObservableCollection](#) data to the MultiSelect component. The following example **Observable Data** is bound to a collection of colors data.

CSHARP

```

@using Syncfusion.Blazor.DropDowns
@using System.Collections.ObjectModel;
<SfMultiSelect TValue="string[]" TItem="Colors" PopupHeight="230px"
Placeholder="Select a color" DataSource="@ColorsData">
<MultiSelectFieldSettings Text="Color"
Value="Code"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
    public class Colors
    {
        public string Code { get; set; }
        public string Color { get; set; }
    }
    private ObservableCollection<Colors> ColorsData = new
    ObservableCollection<Colors>()
    {
        new Colors() { Color = "Chocolate", Code = "#75523C" },
        new Colors() { Color = "CadetBlue", Code = "#3B8289" },
        new Colors() { Color = "DarkOrange", Code = "#FF843D" },
        new Colors() { Color = "DarkRed", Code = "#CA3832" },
        new Colors() { Color = "Fuchsia", Code = "#D44FA3" },
        new Colors() { Color = "HotPink", Code = "#F23F82" },
        new Colors() { Color = "Indigo", Code = "#2F5D81" },
        new Colors() { Color = "LimeGreen", Code = "#4CD242" },
        new Colors() { Color = "OrangeRed", Code = "#FE2A00" },
    }
}

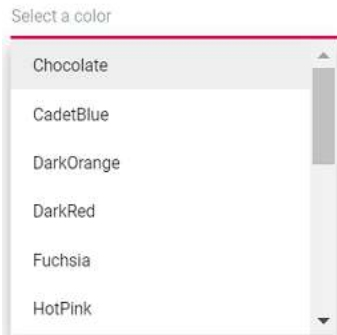
```

```

new Colors() { Color = "Tomato", Code = "#FF745C" },
new Colors() { Color = "Brown", Code = "#A52A2A" },
new Colors() { Color = "Maroon", Code = "#800000" },
new Colors() { Color = "Green", Code = "#008000" },
new Colors() { Color = "Pink", Code = "#FFC0CB" },
new Colors() { Color = "Purple", Code = "#800080" }
};
}

```

The output will shown as follows,



Entity Framework

You need to follow the below steps to consume data from the **Entity Framework** in the MultiSelect component.

Create DbContext class

The first step is to create a DbContext class called **OrderContext** to connect to a Microsoft SQL Server database.

CSHARP

```

using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using EFDropDown.Shared.Models;
namespace EFDropDown.Shared.DataAccess
{
    public class OrderContext : DbContext
    {
        public virtual DbSet<Shared.Models.Order> Orders { get; set; }
        protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
        {
            if (!optionsBuilder.IsConfigured)
            {
                optionsBuilder.UseSqlServer(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=D:\Blazor\DropDownList\EFDrop
Down\Shared\App_Data\NORTHWND.MDF;Integrated Security=True;Connect
Timeout=30");
            }
        }
    }
}

```

```
}
```

Create data access layer to perform data operation

Now you need to create a class named **OrderDataAccessLayer**, which act as data access layer for retrieving the records from the database table.

CSHARP

```
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using EFDropDown.Shared.Models;
namespace EFDropDown.Shared.DataAccess
{
    public class OrderDataAccessLayer
    {
        OrderContext db = new OrderContext();
        //To Get all Orders details
        public DbSet<Order> GetAllOrders()
        {
            try
            {
                return db.Orders;
            }
            catch
            {
                throw;
            }
        }
    }
}
```

Creating Web API Controller

A Web API Controller has to be created which allows MultiSelect directly to consume data from the Entity framework.

CSHARP

```
using EFDropDown.Shared.DataAccess;
using EFDropDown.Shared.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Primitives;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Web;
using Microsoft.AspNetCore.Http;
namespace EFDropDown.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    //TreeGrid
```

```

public class DefaultController : ControllerBase
{
    OrderDataAccessLayer db = new OrderDataAccessLayer();
    [HttpGet]
    public object Get()
    {
        IQueryable<Order> data = db.GetAllOrders().AsQueryable();
        var count = data.Count();
        var queryString = Request.Query;
        if (queryString.Keys.Contains("$inlinecount"))
        {
            StringValues Skip;
            StringValues Take;
            int skip = (queryString.TryGetValue("$skip", out Skip)) ?
            Convert.ToInt32(Skip[0]) : 0;
            int top = (queryString.TryGetValue("$top", out Take)) ?
            Convert.ToInt32(Take[0]) : data.Count();
            return new { Items = data.Skip(skip).Take(top), Count = count };
        }
        else
        {
            return data;
        }
    }
}

```

Configure MultiSelect component using Web API adaptor

Now you can configure the MultiSelect using the '**SfDataManager**' to interact with the created Web API and consume the data appropriately. To interact with web api, you need to use WebApiAdaptor.

ASPX-CS

```

@using Syncfusion.Blazor.Data
@using Syncfusion.Blazor.DropDowns
<SfMultiSelect TValue="string[]" TItem="Order" Placeholder="Select a
Country">
<SfDataManager Url="api/Default" Adaptor="Adaptors.WebApiAdaptor"
CrossDomain="true"></SfDataManager>
<MultiSelectFieldSettings Text="ShipCountry"
Value="OrderID"></MultiSelectFieldSettings>
</SfMultiSelect>
@code{
public class Order
{
    public int? OrderID { get; set; }
    public string[] ShipCountry { get; set; }
}
}

```

Templates in Blazor MultiSelect Dropdown Component

The MultiSelect has been provided with several options to customize each list item, group title, selected value, header, and footer elements.

Item template

The content of each list item within the MultiSelect 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.

ASPX-CS

```
@using Syncfusion.Blazor.Data
@using Syncfusion.Blazor.DropDowns
@using Syncfusion.Blazor
<SfMultiSelect Placeholder="Select a employee" TValue="string[]"
TItem="EmployeeData" Query="@Query">
<MultiSelectTemplates TItem="EmployeeData">
<ItemTemplate>
<span><span class='name'>@((context as EmployeeData).FirstName)</span><span
class='country'>@((context as EmployeeData).Country)</span></span>
</ItemTemplate>
</MultiSelectTemplates>
<SfDataManager Url="https://ej2services.syncfusion.com/production/web-
services/api/Employees" Adaptor="Adaptors.WebApiAdaptor"
CrossDomain=true></SfDataManager>
<MultiSelectFieldSettings Text="FirstName"
Value="Country"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
public class EmployeeData
{
public string FirstName { get; set; }
public string Country { get; set; }
}
public Query Query = new Query();
}
<style>
.country {
right: 15px;
position: absolute;
}
</style>
```

The output will be as follows.

Select a customer

Andrew Fuller	England
Anne Dodsworth	USA
Janet Leverling	USA
Laura Callahan	USA
Margaret Peacock	USA
Michael Suyama	USA

Value template

The currently selected value that is displayed by default on the MultiSelect 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 **Designation** in the MultiSelect input, which is separated by a hyphen.

ASPX-CS

```
@using Syncfusion.Blazor.Data
@using Syncfusion.Blazor.DropDowns
@using Syncfusion.Blazor

<SfMultiSelect TValue="string[]" TItem="EmployeeData" Placeholder="Select a
employee" Query="@Query">
  <MultiSelectTemplates TItem="EmployeeData">
    <ItemTemplate>
      <span><span class='name'>@((context as EmployeeData).FirstName)</span><span
class='destination'>@((context as EmployeeData).Designation)</span></span>
    </ItemTemplate>
    <ValueTemplate>
      <span>@((context as EmployeeData).FirstName) - @((context as
EmployeeData).Designation)</span>
    </ValueTemplate>
  </MultiSelectTemplates>
</SfMultiSelect>

<SfDataManager Url="https://ej2services.syncfusion.com/production/web-
services/api/Employees" Adaptor="Adaptors.WebApiAdaptor"
CrossDomain=true></SfDataManager>

<MultiSelectFieldSettings Text="FirstName"
Value="Designation"></MultiSelectFieldSettings>
</SfMultiSelect>

@code {
public class EmployeeData
{
public string FirstName { get; set; }
public string Designation { get; set; }
}
public Query Query = new Query();
}

<style>
.destination {
right: 15px;
position: absolute;
}
</style>
```

The output will be as follows.

Select a customer

Andrew Fuller	Team Lead
Anne Dodsworth	Developer
Janet Leverling	HR
Laura Callahan	Product Manager
Margaret Peacock	Developer
Michael Suyama	Team Lead

Header template

The header element is shown statically at the top of the popup list items within the MultiSelect, 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.

ASPX-CS

```
@using Syncfusion.Blazor.Data
@using Syncfusion.Blazor.DropDowns
@using Syncfusion.Blazor
<SfMultiSelect TValue="string[]" TItem="EmployeeData" Placeholder="Select a
employee" Query="@Query">
<MultiSelectTemplates TItem="EmployeeData">
<ItemTemplate>
<span class='item'><span class='name'>@((context as
EmployeeData).FirstName)</span><span class='city'>@((context as
EmployeeData).Country)</span></span>
</ItemTemplate>
<HeaderTemplate>
<span class='head'><span class='name'>Name</span><span
class='city'>Country</span></span>
</HeaderTemplate>
</MultiSelectTemplates>
<SfDataManager Url="https://ej2services.syncfusion.com/production/web-
services/api/Employees" Adaptor="Adaptors.WebApiAdaptor"
CrossDomain=true></SfDataManager>
<MultiSelectFieldSettings Value="Country"
Text="FirstName"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
public class EmployeeData
{
public string FirstName { get; set; }
public string Country { get; set; }
}
public Query Query = new Query();
}
<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>

```

The output will be as follows.

Select a customer

Name	Country
Andrew Fuller	England
Anne Dodsworth	USA
Janet Leverling	USA
Laura Callahan	USA
Margaret Peacock	USA
Michael Suyama	USA

Footer template

The MultiSelect 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 MultiSelect.

ASPX-CS

```

@using Syncfusion.Blazor.Data
@using Syncfusion.Blazor.DropDowns
@using Syncfusion.Blazor
<SfMultiSelect TValue="string[]" TItem="EmployeeData" Query="@Query"
Placeholder="Select a customer">
<MultiSelectTemplates TItem="EmployeeData">
<FooterTemplate>
<span class='footer'>Total list Item: 6 </span>
</FooterTemplate>
</MultiSelectTemplates>

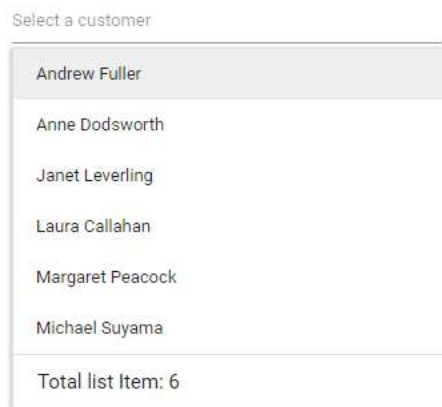
```

```

<SfDataManager Url="https://ej2services.syncfusion.com/production/web-
services/api/Employees" Adaptor="Adaptors.WebApiAdaptor"
CrossDomain=true></SfDataManager>
<MultiSelectFieldSettings Value="Country"
Text="FirstName"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
public class EmployeeData
{
public string FirstName { get; set; }
}
public Query Query = new Query();
}
<style>
.footer {
text-indent: 1.2em;
display: block;
font-size: 15px;
line-height: 40px;
border-top: 1px solid #e0e0e0;
}
</style>

```

The output will be as follows.



No records template

The MultiSelect is provided with support to custom design the popup list content when no data is found and no matches found on search with the help of [NoRecordsTemplate](#) property.

In the following sample, popup list content displays the notification of no data available.

ASPX-CS

```

@using Syncfusion.Blazor.DropDowns
<SfMultiSelect TValue="string[]" TItem="EmployeeData" Placeholder="Select a
employee" CssClass="e-custom" DataSource="@employee">
<MultiSelectTemplates TItem="EmployeeData">
<NoRecordsTemplate>
<span class='norecord'> NO DATA AVAILABLE</span>
</NoRecordsTemplate>
</MultiSelectTemplates>
</SfMultiSelect>

```

```
@code {
public class EmployeeData { }
List<EmployeeData> employee = new List<EmployeeData> { };
}
```

The output will be as follows.

Select a customer

NO DATA AVAILABLE

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 MultiSelect displays the notification.

ASPX-CS

```
@using Syncfusion.Blazor.Data
@using Syncfusion.Blazor.DropDowns
@using Syncfusion.Blazor
<SfMultiSelect TValue="string[]" TItem="EmployeeData" Placeholder="Select a customer" Query="@Query">
  <MultiSelectTemplates TItem="EmployeeData">
    <ActionFailureTemplate>
      <span class='norecord'>Data fetch get fails </span>
    </ActionFailureTemplate>
  </MultiSelectTemplates>
  <SfDataManager
    Url="https://services.odata.org/V4/Northwind/Northwind.svcs/Employees"
    Adaptor="Adaptors.ODataV4Adaptor" CrossDomain=true></SfDataManager>
  <MultiSelectFieldSettings Value="Country"
    Text="FirstName"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
public class EmployeeData
{
public string FirstName { get; set; }
}
public Query Query = new Query();
}
```

The output will be as follows.

Select a customer

Data fetch get fails

Grouping in Blazor MultiSelect Dropdown Component

The MultiSelect 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

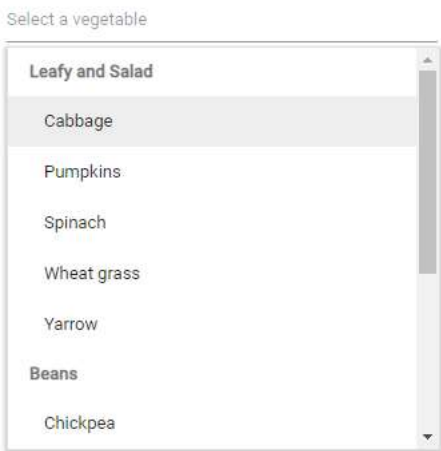
is displayed both as inline and fixed headers. The fixed group header content is updated dynamically on scrolling the popup list with its category value.

In the following sample, vegetables are grouped according on its category using `GroupBy` field.

ASPX-CS

```
@using Syncfusion.Blazor.DropDowns
<SfMultiSelect TValue="string[]" TItem="Vegetables" Placeholder="Select a
vegetable" DataSource="@LocalData">
<MultiSelectFieldSettings GroupBy="Category"
Value="Vegetable"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
public object LocalData { get; set; } = new Vegetables().VegetablesList();
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;
}
}
```

The output will be as follows.



Filtering in Blazor MultiSelect Dropdown Component

The MultiSelect 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 MultiSelect input.

ASPX-CS

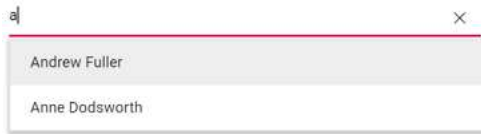
```
@using Syncfusion.Blazor.DropDowns
<SfMultiSelect TValue="string[]" TItem="Countries" Placeholder="Select a
country" AllowFiltering="true" DataSource="@Country">
<MultiSelectFieldSettings Text="Name"
Value="Code"></MultiSelectFieldSettings>
</SfMultiSelect>
@code{
public class Countries
{
public string Name { get; set; }
public string Code { get; set; }
}
private List<Countries> Country = new List<Countries>
{
new Countries() { Name = "Australia", Code = "AU" },
new Countries() { Name = "Bermuda", Code = "BM" },
new Countries() { Name = "Canada", Code = "CA" },
new Countries() { Name = "Cameroon", Code = "CM" },
new Countries() { Name = "Denmark", Code = "DK" },
new Countries() { Name = "France", Code = "FR" },
new Countries() { Name = "Finland", Code = "FI" },
new Countries() { Name = "Germany", Code = "DE" },
new Countries() { Name = "Greenland", Code = "GL" },
new Countries() { Name = "Hong Kong", Code = "HK" },
new Countries() { Name = "India", Code = "IN" },
new Countries() { Name = "Italy", Code = "IT" },
new Countries() { Name = "Japan", Code = "JP" },
new Countries() { Name = "Mexico", Code = "MX" },
new Countries() { Name = "Norway", Code = "NO" },
new Countries() { Name = "Poland", Code = "PL" },
new Countries() { Name = "Switzerland", Code = "CH" },
new Countries() { Name = "United Kingdom", Code = "GB" },
new Countries() { Name = "United States", Code = "US" },
};
```



```
}

```

The output will be as follows.



Custom Filtering

The MultiSelect component filter queries can be customized. You can also use your own filter libraries to filter data like Fuzzy search.

ASPX-CS

```
@using Syncfusion.Blazor.Data
@using Syncfusion.Blazor.DropDowns
<SfMultiSelect TValue="string[]" @ref="mulObj" TItem="Countries"
Placeholder="e.g. Australia" DataSource="@Country" AllowFiltering="true">
<MultiSelectFieldSettings Text="Name"
Value="Code"></MultiSelectFieldSettings>
<MultiSelectEvents TValue="string[]" TItem="Countries"
Filtering="OnFilter"></MultiSelectEvents>
</SfMultiSelect>
@code {
SfMultiSelect<string[], Countries> mulObj { get; set; }
public class Countries
{
public string Name { get; set; }
public string Code { get; set; }
}
List<Countries> Country = new List<Countries>
{
new Countries() { Name = "Australia", Code = "AU" },
new Countries() { Name = "Bermuda", Code = "BM" },
new Countries() { Name = "Canada", Code = "CA" },
new Countries() { Name = "Cameroon", Code = "CM" },
new Countries() { Name = "Denmark", Code = "DK" }
};
List<Countries> Country1 = new List<Countries>
{
new Countries() { Name = "France", Code = "FR" },
new Countries() { Name = "Finland", Code = "FI" },
new Countries() { Name = "Germany", Code = "DE" },
new Countries() { Name = "Greenland", Code = "GL" }
};
private async Task OnFilter(FilteringEventArgs args)
{
args.PreventDefaultAction = true;
var query = new Query().Where(new WhereFilter() { Field = "Name", Operator =
"contains", value = args.Text, IgnoreCase = true });
query = !string.IsNullOrEmpty(args.Text) ? query : new Query();
await mulObj.FilterAsync(Country1, query);
}
}
```

Custom Value in Blazor MultiSelect Dropdown Component

The MultiSelect allows the user to select and tag the typed custom value that is not present in the data source when you set the [AllowCustomValue](#) as true. The selected custom value is added to the suggestion list alone. It will not affect the original data source. The `CustomValueSpecifier` event will trigger when you select or tag the typed custom value.

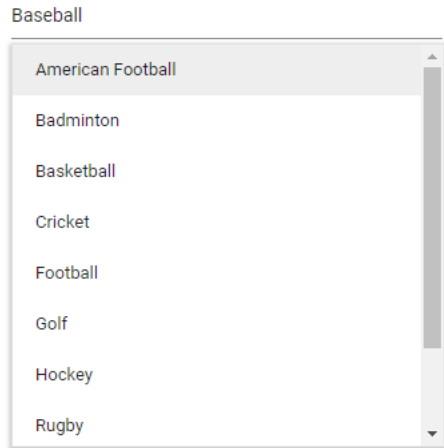
The `Value` field, `Text` field, and `Value` property must be of `string` type when you enable the custom value. For other types, you must provide the custom data for the typed custom value in the `CustomValueSpecifier` event. Please find the details on [Value as non-string type](#) section.

The following sample demonstrates configuration of custom value support with the MultiSelect component.

ASPX-CS

```
@using Syncfusion.Blazor.DropDowns
<SfMultiSelect TValue="string[]" Placeholder="Favorite Sports"
AllowCustomValue=true DataSource="@LocalData">
<MultiSelectFieldSettings Text="Text" Value="ID"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
public class Games
{
public string ID { get; set; }
public string Text { get; set; }
}
List<Games> LocalData = new List<Games> {
new Games() { ID= "Game1", Text= "American Football" },
new Games() { ID= "Game2", Text= "Badminton" },
new Games() { ID= "Game3", Text= "Basketball" },
new Games() { ID= "Game4", Text= "Cricket" },
new Games() { ID= "Game5", Text= "Football" },
new Games() { ID= "Game6", Text= "Golf" },
new Games() { ID= "Game7", Text= "Hockey" },
new Games() { ID= "Game8", Text= "Rugby"},
new Games() { ID= "Game9", Text= "Snooker" },
new Games() { ID= "Game10", Text= "Tennis"},
};
}
```

The output will be as follows.



Value as non-string type

By default, the typed custom value is updated to both the Value and Text field of the custom data. If the TValue type is a non-string type, then you have to provide the custom data for the typed custom value in the CustomValueSpecifier event like given below.

In the CustomValueSpecifier event, you will get the typed custom value in the Text argument of the event. Based on that text value, you should form the data for the custom value and set it to the NewData argument of the event.

The following sample demonstrates configuration of custom value in CustomValueSpecifier event.

ASPX-CS

```
@using Syncfusion.Blazor.DropDowns
<SfMultiSelect TValue="int[]" TItem="Games" Placeholder="Favorite Sports"
AllowCustomValue=true DataSource="@LocalData">
  <MultiSelectFieldSettings Text="Text" Value="ID"></MultiSelectFieldSettings>
  <MultiSelectEvents TValue="int[]" TItem="Games"
CustomValueSpecifier="@CustomValueHandler"></MultiSelectEvents>
</SfMultiSelect>
@code {
public class Games
{
public int ID { get; set; }
public string Text { get; set; }
}

List<Games> LocalData = new List<Games> {
new Games() { ID= 1, Text= "American Football" },
new Games() { ID= 2, Text= "Badminton" },
new Games() { ID= 3, Text= "Basketball" },
new Games() { ID= 4, Text= "Cricket" },
new Games() { ID= 5, Text= "Football" },
new Games() { ID= 6, Text= "Golf" },
new Games() { ID= 7, Text= "Hockey" },
new Games() { ID= 8, Text= "Rugby"},
new Games() { ID= 9, Text= "Snooker" },
new Games() { ID= 10, Text= "Tennis"},
};
private void CustomValueHandler(CustomValueEventArgs<Games> args)
{
}
```

```
System.Random random = new System.Random();
args.NewData = new Games() { ID = random.Next(100), Text = args.Text };
}
}
```

Checkbox Grouping in Blazor MultiSelect Component

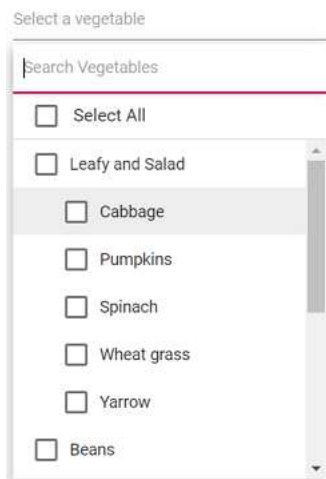
You can arrange the datasource items by grouping them with checkbox mode in MultiSelect. Clicking the checkbox in group will select all the items grouped under it. Click the MultiSelect element and then type the character in the search box. It will display the filtered list items based on the typed characters and then select the multiple values through the checkbox.

ASPX-CS

```
@using Syncfusion.Blazor.DropDowns
<SfMultiSelect TValue="string[]" AllowFiltering="true" TItem="Vegetables"
Mode="@VisualMode.CheckBox" Width="250px" Placeholder="Select a vegetable"
DataSource="@VegetablesList" ShowSelectAll="@ShowSelectAllCheckBox"
EnableSelectionOrder="@EnableSelectionOrders" FilterBarPlaceholder="Search
Vegetables" EnableGroupCheckBox="@EnableCheckBox" PopupHeight="350px">
<MultiSelectFieldSettings GroupBy="Category"
Value="Vegetable"></MultiSelectFieldSettings>
</SfMultiSelect>
@code{
public bool ShowSelectAllCheckBox { get; set; } = true;
public bool EnableSelectionOrders { get; set; } = false;
public bool EnableCheckBox { get; set; } = true;
public class Vegetables
{
public string Vegetable { get; set; }
public string Category { get; set; }
public string ID { get; set; }
}
List<Vegetables> VegetablesList = new List<Vegetables>()
{
new Vegetables() { Vegetable = "Cabbage", Category = "Leafy and Salad", ID =
"item1" },
new Vegetables() { Vegetable = "Chickpea", Category = "Beans", ID = "item2"
},
new Vegetables() { Vegetable = "Garlic", Category = "Bulb and Stem", ID =
"item3" },
new Vegetables() { Vegetable = "Green bean", Category = "Beans", ID =
"item4" },
new Vegetables() { Vegetable = "Horse gram", Category = "Beans", ID =
"item5" },
new Vegetables() { Vegetable = "Nopal", Category = "Bulb and Stem", ID =
"item6" },
new Vegetables() { Vegetable = "Onion", Category = "Bulb and Stem", ID =
"item7" },
new Vegetables() { Vegetable = "Pumpkins", Category = "Leafy and Salad", ID =
"item8" },
new Vegetables() { Vegetable = "Spinach", Category = "Leafy and Salad", ID =
"item9" },
new Vegetables() { Vegetable = "Wheat grass", Category = "Leafy and Salad",
ID = "item10" },
}
```

```
new Vegetables() { Vegetable = "Yarrow", Category = "Leafy and Salad", ID = "item11" },
};
}
```

The output will be as follows.



CheckBox in Blazor MultiSelect Dropdown Component

The MultiSelect has built-in support to select multiple values through checkbox, when the [Mode](#) property is set to `CheckBox`.

To use checkbox, inject the `CheckBoxSelection` module in the MultiSelect.

ASPX-CS

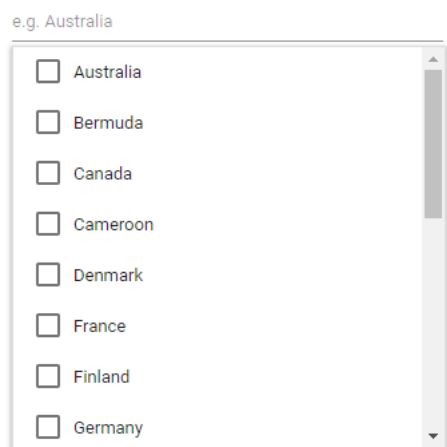
```
@using Syncfusion.Blazor.DropDowns
<SfMultiSelect TValue="string[]" Placeholder="e.g. Australia"
Mode="VisualMode.CheckBox" DataSource="@Country">
<MultiSelectFieldSettings Value="Code"
Text="Name"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
public class Countries
{
public string Name { get; set; }
public string Code { get; set; }
}
List<Countries>Country = new List<Countries>
{
new Countries() { Name = "Australia", Code = "AU" },
new Countries() { Name = "Bermuda", Code = "BM" },
new Countries() { Name = "Canada", Code = "CA" },
new Countries() { Name = "Cameroon", Code = "CM" },
new Countries() { Name = "Denmark", Code = "DK" },
new Countries() { Name = "France", Code = "FR" },
new Countries() { Name = "Finland", Code = "FI" },
new Countries() { Name = "Germany", Code = "DE" },
new Countries() { Name = "Greenland", Code = "GL" },
new Countries() { Name = "Hong Kong", Code = "HK" },
new Countries() { Name = "India", Code = "IN" },
}
```

```

new Countries() { Name = "Italy", Code = "IT" },
new Countries() { Name = "Japan", Code = "JP" },
new Countries() { Name = "Mexico", Code = "MX" },
new Countries() { Name = "Norway", Code = "NO" },
new Countries() { Name = "Poland", Code = "PL" },
new Countries() { Name = "Switzerland", Code = "CH" },
new Countries() { Name = "United Kingdom", Code = "GB" },
new Countries() { Name = "United States", Code = "US" },
};
}

```

The output will be as follows.



Select All

The MultiSelect component has in-built support to select the all list items using **Select All** options in the header. When the [ShowSelectAll](#) property is set to true, by default Select All text will show. You can customize the name attribute of the Select All option by using [SelectAllText](#).

For the unSelect All option, by default unSelect All text will show. You can customize the name attribute of the unSelect All option by using [UnSelectAllText](#).

ASPX-CS

```

@using Syncfusion.Blazor.DropDowns
<SfMultiSelect TValue="string[]" Placeholder="e.g. Australia"
ShowSelectAll=true SelectAllText="Select All" UnSelectAllText="unSelect All"
Mode="VisualMode.CheckBox" DataSource="@Country">
  <MultiSelectFieldSettings Text="Name"
  Value="Code"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
public class Countries
{
public string Name { get; set; }
public string Code { get; set; }
}
List<Countries>Country = new List<Countries>
{
new Countries() { Name = "Australia", Code = "AU" },
new Countries() { Name = "Bermuda", Code = "BM" },

```

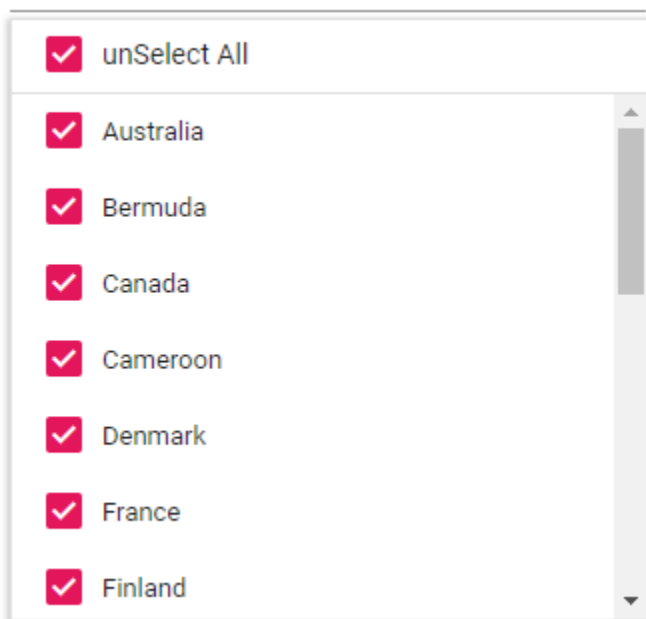
```

new Countries() { Name = "Canada", Code = "CA" },
new Countries() { Name = "Cameroon", Code = "CM" },
new Countries() { Name = "Denmark", Code = "DK" },
new Countries() { Name = "France", Code = "FR" },
new Countries() { Name = "Finland", Code = "FI" },
new Countries() { Name = "Germany", Code = "DE" },
new Countries() { Name = "Greenland", Code = "GL" },
new Countries() { Name = "Hong Kong", Code = "HK" },
new Countries() { Name = "India", Code = "IN" },
new Countries() { Name = "Italy", Code = "IT" },
new Countries() { Name = "Japan", Code = "JP" },
new Countries() { Name = "Mexico", Code = "MX" },
new Countries() { Name = "Norway", Code = "NO" },
new Countries() { Name = "Poland", Code = "PL" },
new Countries() { Name = "Switzerland", Code = "CH" },
new Countries() { Name = "United Kingdom", Code = "GB" },
new Countries() { Name = "United States", Code = "US" },
};
}

```

The output will be as follows.

Australia, Bermuda, Canada +16 more..



Selection Limit

Defines the limit of the selected items using [MaximumSelectionLength](#).

ASPX-CS

```

<SfMultiSelect TValue="string[]" Placeholder="e.g. Australia"
MaximumSelectionLength=3 Mode="VisualMode.CheckBox" DataSource="@Country">
  <MultiSelectFieldSettings Text="Name"
  Value="Code"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {

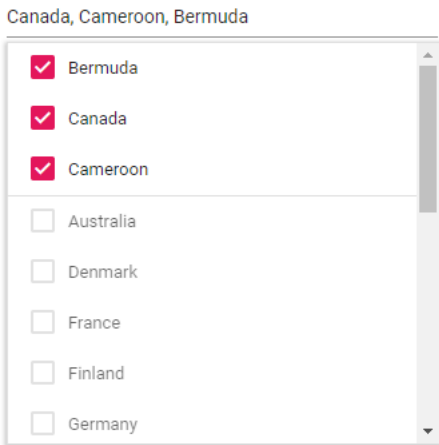
```

```

public class Countries
{
    public string Name { get; set; }
    public string Code { get; set; }
}
List<Countries>Country = new List<Countries>
{
    new Countries() { Name = "Australia", Code = "AU" },
    new Countries() { Name = "Bermuda", Code = "BM" },
    new Countries() { Name = "Canada", Code = "CA" },
    new Countries() { Name = "Cameroon", Code = "CM" },
    new Countries() { Name = "Denmark", Code = "DK" },
    new Countries() { Name = "France", Code = "FR" },
    new Countries() { Name = "Finland", Code = "FI" },
    new Countries() { Name = "Germany", Code = "DE" },
    new Countries() { Name = "Greenland", Code = "GL" },
    new Countries() { Name = "Hong Kong", Code = "HK" },
    new Countries() { Name = "India", Code = "IN" },
    new Countries() { Name = "Italy", Code = "IT" },
    new Countries() { Name = "Japan", Code = "JP" },
    new Countries() { Name = "Mexico", Code = "MX" },
    new Countries() { Name = "Norway", Code = "NO" },
    new Countries() { Name = "Poland", Code = "PL" },
    new Countries() { Name = "Switzerland", Code = "CH" },
    new Countries() { Name = "United Kingdom", Code = "GB" },
    new Countries() { Name = "United States", Code = "US" },
};
}

```

The output will be as follows.



Selection Reordering

Using [EnableSelectionOrder](#) to Reorder the selected items in popup visibility state.

ASPX-CS

```

<SfMultiSelect TValue="string[]" Placeholder="e.g. Australia"
EnableSelectionOrder=false Mode="VisualMode.CheckBox"
DataSource="@Country">

```

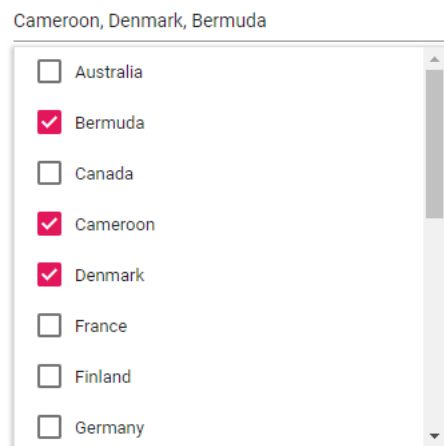


```

<MultiSelectFieldSettings Text="Name"
Value="Code"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
public class Countries
{
public string Name { get; set; }
public string Code { get; set; }
}
List<Countries>Country = new List<Countries>
{
new Countries() { Name = "Australia", Code = "AU" },
new Countries() { Name = "Bermuda", Code = "BM" },
new Countries() { Name = "Canada", Code = "CA" },
new Countries() { Name = "Cameroon", Code = "CM" },
new Countries() { Name = "Denmark", Code = "DK" },
new Countries() { Name = "France", Code = "FR" },
new Countries() { Name = "Finland", Code = "FI" },
new Countries() { Name = "Germany", Code = "DE" },
new Countries() { Name = "Greenland", Code = "GL" },
new Countries() { Name = "Hong Kong", Code = "HK" },
new Countries() { Name = "India", Code = "IN" },
new Countries() { Name = "Italy", Code = "IT" },
new Countries() { Name = "Japan", Code = "JP" },
new Countries() { Name = "Mexico", Code = "MX" },
new Countries() { Name = "Norway", Code = "NO" },
new Countries() { Name = "Poland", Code = "PL" },
new Countries() { Name = "Switzerland", Code = "CH" },
new Countries() { Name = "United Kingdom", Code = "GB" },
new Countries() { Name = "United States", Code = "US" },
};
}

```

The output will be as follows.



Style and appearance in Blazor MultiSelect Dropdown 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 of wrapper element

Use the following CSS to customize the background color of wrapper element.

CSS

```
.e-multiselect.e-input-group .e-multi-select-wrapper {  
  background-color: red;  
}
```

Customizing the appearance of the delimiter wrapper element

Use the following CSS to customize the appearance of delimiter wrapper element.

CSS

```
.e-multiselect .e-delim-values {  
  -webkit-text-fill-color: blue;  
  font-size: 16px;  
  font-family: cursive;  
}
```

Customizing the appearance of chips

Use the following CSS to customize the appearance of selected chips.

CSS

```
.e-multiselect .e-multi-select-wrapper .e-chips .e-chipcontent {  
  font-family: cursive;  
  font-size: 20px;  
  -webkit-text-fill-color: blue;  
}  
.e-multi-select-wrapper .e-chips {  
  background-color: aqua;  
  height: 26px;  
}
```

Customizing the dropdown icon's color

Use the following CSS to customize the dropdown icon's color.

CSS

```
.e-multiselect .e-input-group-icon.e-ddl-icon.e-icons, .e-multiselect .e-  
input-group-icon.e-ddl-icon.e-icons:hover {  
  color: red;  
  font-size: 13px;  
}
```

Customizing the focus color

Use the following CSS to customize the focusing color of input element.

CSS

```
.e-multiselect.e-input-group.e-control-wrapper.e-input-focus::before, .e-  
multiselect.e-input-group.e-control-wrapper.e-input-focus::after {  
  background: #c000ff;  
}
```

Customizing the disabled component's text color

Use the following CSS to customize the text color when the component is disabled.

CSS

```
.e-multiselect.e-disabled .e-multi-select-wrapper .e-delim-values {  
  -webkit-text-fill-color: red;  
}
```

Customizing the color of the placeholder text

Use the following CSS to customize the text color of placeholder.

CSS

```
.e-multiselect input.e-multiselect::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-control-container.e-float-input .e-float-  
text::after {  
  content: "*";  
  color: red;  
}
```

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-multiselect.e-input-group.e-control-wrapper.e-control-container.e-float-  
input.e-input-focus .e-float-text.e-label-top {  
  color: #2319b8;  
}
```

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 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
}
```

Customizing the color of the checkbox

Use the following CSS to customize the color of checkbox.

CSS

```
.e-popup .e-checkbox-wrapper .e-frame.e-check, .e-popup .e-checkbox-
wrapper:hover .e-frame.e-check {
background-color: green;
color: white;
}
```

Virtualization in Blazor MultiSelect Dropdown Component

The MultiSelect has been provided virtualization to improve the UI performance for a large amount of data when [EnableVirtualization](#) is true. This feature doesn't render out the entire data source on initial component rendering. It loads the N number of items in the popup on initial rendering and the remaining set number of items will load on each scrolling action in the popup. It can work with both local and remote data.

In the following code 150 items bound to the component, but only 6 items will load to the popup when you open the popup. Remaining set number of items will load on each scrolling action in the popup.

ASPX-CS

```
@using Syncfusion.Blazor.DropDowns
@using Syncfusion.Blazor.Data
<SfMultiSelect TValue="string[]" TItem="Record" Placeholder="Select an item"
DataSource="@Records" Query="@LocalDataQuery" PopupHeight="130px"
EnableVirtualization="true">
<MultiSelectFieldSettings Text="Text" Value="ID"></MultiSelectFieldSettings>
</SfMultiSelect>
@code{
public Query LocalDataQuery = new Query().Take(6);
public class Record
{
public string ID { get; set; }
public string Text { get; set; }
}
public List<Record> Records { get; set; }
protected override void OnInitialized()
{
this.Records = Enumerable.Range(1, 150).Select(i => new Record()
{
ID = i.ToString(),
Text = "Item " + i,
}).ToList();
}
}
```

The output will shown as follows,



Localization in Blazor MultiSelect Dropdown Component

Blazor server side

Add `UseRequestLocalization` middle-ware in Configure method in **Startup.cs** file to get browser Culture Info.

Refer the following code to add configuration in Startup.cs file

CSHARP

```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Localization;
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            app.UseRequestLocalization();
            ....
            ....
        }
    }
}
```

The **Localization** library allows you to localize default text content. The MultiSelect Dropdown component has static text that can be changed to other cultures (Arabic, Deutsch, French, etc.).

In the following examples, demonstrate how to enable **Localization** for MultiSelect Dropdown in server side Blazor samples. Here, we have used Resource file to translate the static text.

The Resource file is an XML file which contains the strings(key and value pairs) that you want to translate into different language. You can also refer Localization [link](#) to know more about how to configure and use localization in the ASP.NET Core application framework.

- Open the **Startup.cs** file and add the below configuration in the **ConfigureServices** function as follows.

CSHARP

```
using Syncfusion.Blazor;
using System.Globalization;
using Microsoft.AspNetCore.Localization;
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void ConfigureServices(IServiceCollection services)
        {
            ....
            ....
            services.AddSyncfusionBlazor();
            services.AddLocalization(options => options.ResourcesPath = "Resources");
            services.Configure<RequestLocalizationOptions>(options =>
            {
                // define the list of cultures your app will support
                var supportedCultures = new List<CultureInfo>()
                {
                    new CultureInfo("de")
                }
            })
        }
    }
}
```

```

};
// set the default culture
options.DefaultRequestCulture = new RequestCulture("de");
options.SupportedCultures = supportedCultures;
options.SupportedUICultures = supportedCultures;
options.RequestCultureProviders = new List<IRequestCultureProvider>() {
    new QueryStringRequestCultureProvider() // Here, You can also use other
    localization provider
};
});
services.AddSingleton(typeof(ISyncfusionStringLocalizer),
    typeof(SampleLocalizer));
}
}
}

```

- Then, write a **class** by inheriting **ISyncfusionStringLocalizer** interface and override the **Manager** property to get the resource file details from the application end.

CSHARP

```

using Syncfusion.Blazor;
namespace blazorDropdowns
{
    public class SampleLocalizer : ISyncfusionStringLocalizer
    {
        public string Get(string key)
        {
            return this.Manager.GetString(key);
        }
        public System.Resources.ResourceManager Manager
        {
            get
            {
                return blazorDropdowns.Resources.SyncfusionBlazorLocale.ResourceManager;
            }
        }
    }
}

```

- Add **.resx** file to Resource folder and enter the key value (Locale Keywords) in the **Name** column and the translated string in the Value column as follows.

Name	Value (in Deutsch culture)
---	---
MultiSelect_ActionFailureTemplate	Die Anfrage ist fehlgeschlagen
MultiSelect_NoRecordsTemplate	Keine Aufzeichnungen gefunden
MultiSelect_OverflowCountTemplate	+\$ {count} mehr ..
MultiSelect_SelectAllText	Wählen Sie Alle

| MultiSelect_TotalCountTemplate | \${count} ausgewählt |

| MultiSelect_UnSelectAllText | Alles widerrufen |

- Finally, Specify the culture for MultiSelect Dropdown using `locale` property.

ASPX-CS

```
@using Syncfusion.Blazor.DropDowns
<SfMultiSelect TValue="string[]" TItem="Games" Placeholder="Favorite Sports"
Locale="de" AllowFiltering="true" DataSource="@LocalData">
<MultiSelectFieldSettings Text="Text" Value="ID"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
public class Games
{
public string ID { get; set; }
public string Text { get; set; }
}
List<Games> LocalData = new List<Games> {
new Games() { ID= "Game1", Text= "American Football" },
new Games() { ID= "Game2", Text= "Badminton" },
new Games() { ID= "Game3", Text= "Basketball" },
new Games() { ID= "Game4", Text= "Cricket" },
new Games() { ID= "Game5", Text= "Football" },
};
}
```

Blazor WebAssembly

The Localization library allows you to localize static text content of the [NoRecordsTemplate](#) and [ActionFailureTemplate](#) properties according to the culture currently assigned to the MultiSelect.

| Locale key | en-US (default)

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

| NoRecordsTemplate | No records found

| ActionFailureTemplate | The request failed

| OverflowCountTemplate | +\${count} more..

| SelectAllText | Select All

| UnSelectAllText | Unselect All

| TotalCountTemplate | \${count} selected

The following steps explain how to render the MultiSelect in French culture (fr) in Blazor Web Assembly application.

- Open the **program.cs** file and add the below configuration in the **Builder ConfigureServices** function as follows.

CSHARP

```
using Syncfusion.Blazor;
```



```

using Microsoft.AspNetCore.Builder;
namespace WebAssemblyLocale
{
    public class Program
    {
        public static async Task Main(string[] args)
        {
            ....
            ....
            builder.Services.Configure<RequestLocalizationOptions>(options =>
            {
                // Define the list of cultures your app will support
                var supportedCultures = new List<System.Globalization.CultureInfo>()
                {
                    new System.Globalization.CultureInfo("en-US"),
                    new System.Globalization.CultureInfo("fr"),
                };
                // Set the default culture
                options.DefaultRequestCulture = new
                Microsoft.AspNetCore.Localization.RequestCulture("fr");
                options.SupportedCultures = supportedCultures;
                options.SupportedUICultures = supportedCultures;
                options.RequestCultureProviders = new
                List<Microsoft.AspNetCore.Localization.IRequestCultureProvider>() {
                    new Microsoft.AspNetCore.Localization.QueryStringRequestCultureProvider()
                };
            });
            ....
            ....
        }
    }
}

```

- To download the locale definition of Blazor components, use this [link](#).
- After downloading the `blazor-locale` package, copy the `blazor-locale` folder with required local definition file into `wwwroot` folder.
- By default, the `blazor-locale` package contains the localized text for static text present in components like button text, placeholder, tooltip, and more.
- Set the culture by using the `SetCulture` method.

In the following sample, French culture is set to the MultiSelect 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.

ASPX-CS

```

@using Syncfusion.Blazor.Data
@using Syncfusion.Blazor.DropDowns
@inject HttpClient Http;
<SfMultiSelect TValue="string[]" TItem="EmployeeData" Query="@Query"
Placeholder="Select a customer" Locale="fr" AllowFiltering="true" >
<SfDataManager Url="https://ej2services.syncfusion.com/production/web-
services/api/Employees" Adaptor="Adaptors.WebApiAdaptor"
CrossDomain=true></SfDataManager>

```

```

<MultiSelectFieldSettings Value="Country"
Text="FirstName"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
[Inject]
protected IJSRuntime JsRuntime { get; set; }
protected override async Task OnInitializedAsync()
{
this.JsRuntime.Sf().LoadLocaleData(await Http.GetJsonAsync<object>("blazor-
locale/src/fr.json")).SetCulture("fr");
}
public Query Query = new Query();
public class EmployeeData
{
public int EmployeeID { get; set; }
public string FirstName { get; set; }
public string Designation { get; set; }
public string Country { get; set; }
}
}

```

The output will be as follows.

Select a customer

La demande a échoué

Accessibility in Blazor MultiSelect Dropdown Component

The MultiSelect component has been designed, keeping in mind the WAI-ARIA specifications, and applies the WAI-ARIA roles, states, and properties along with keyboard support. This component 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.

ARIA attributes

The MultiSelect component uses the **Listbox** role, and each list item has an **option** role. The following **ARIA attributes** denotes the MultiSelect state:

| Properties | Functionalities |

| --- | --- |

| aria-haspopup | Indicates whether the MultiSelect 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 MultiSelect element. |

| aria-disabled | Indicates whether the MultiSelect component is in 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 MultiSelect without interruptions:

| **Keyboard shortcuts** | **Actions** |

| --- | --- |

| Arrow Down | Sets focus at the first item in the MultiSelect when no item selected. Otherwise, moves focus next to the currently selected item. |

| Arrow Up | Moves focus previous to the currently selected one. |

| Page Down | Scrolls down to the next page and set focus to the first item when popup list opens. |

| Page Up | Scrolls up to the previous page and set focus to the first item when popup list opens. |

| Enter | Selects the focused item, and when it is in an close state the popup list opens. |

| 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 component. |

| 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 component. |

| 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 | Sets focus to the first item. |

| End | Sets focus to the last item. |

Events in Blazor MultiSelect Component

This section explains the list of events of the MultiSelect component which will be triggered for appropriate MultiSelect actions.

Blur

Blur event triggers when the input loses focus.

ASPX-CS

```
@using Syncfusion.Blazor.DropDowns
<SfMultiSelect TItem="GameFields" TValue="string[]" DataSource="@Games">
  <MultiSelectEvents TItem="GameFields" TValue="string[]"
  Blur="@BlurHandler"></MultiSelectEvents>
  <MultiSelectFieldSettings Text="Text" Value="ID"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
  public class GameFields
  {
    public string ID { get; set; }
  }
}
```

```

public string Text { get; set; }
}
private List<GameFields> Games = new List<GameFields>() {
new GameFields(){ ID= "Game1", Text= "American Football" },
new GameFields(){ ID= "Game2", Text= "Badminton" },
new GameFields(){ ID= "Game3", Text= "Basketball" },
new GameFields(){ ID= "Game4", Text= "Cricket" },
};
private void BlurHandler(Object args)
{
// Here you can customize your code
}
}

```

ValueChange

ValueChange event triggers when the MultiSelect value is changed.

ASPX-CS

```

@using Syncfusion.Blazor.DropDowns
<SfMultiSelect TItem="GameFields" TValue="string[]" DataSource="@Games">
  <MultiSelectEvents TItem="GameFields" TValue="string[]"
  ValueChange="@ValueChangeHandler"></MultiSelectEvents>
  <MultiSelectFieldSettings Text="Text" Value="ID"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
public class GameFields
{
public string ID { get; set; }
public string Text { get; set; }
}
private List<GameFields> Games = new List<GameFields>() {
new GameFields(){ ID= "Game1", Text= "American Football" },
new GameFields(){ ID= "Game2", Text= "Badminton" },
new GameFields(){ ID= "Game3", Text= "Basketball" },
new GameFields(){ ID= "Game4", Text= "Cricket" },
};
private void ValueChangeHandler(MultiSelectChangeEventArgs<string[]> args)
{
// Here you can customize your code
}
}

```

Closed

Closed event triggers after the popup has been closed.

ASPX-CS

```

@using Syncfusion.Blazor.DropDowns
<SfMultiSelect TItem="GameFields" TValue="string[]" DataSource="@Games">
  <MultiSelectEvents TItem="GameFields" TValue="string[]"
  Closed="@CloseHandler"></MultiSelectEvents>
  <MultiSelectFieldSettings Text="Text" Value="ID"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {

```

```

public class GameFields
{
    public string ID { get; set; }
    public string Text { get; set; }
}
private List<GameFields> Games = new List<GameFields>() {
    new GameFields(){ ID= "Game1", Text= "American Football" },
    new GameFields(){ ID= "Game2", Text= "Badminton" },
    new GameFields(){ ID= "Game3", Text= "Basketball" },
    new GameFields(){ ID= "Game4", Text= "Cricket" },
};
private void CloseHandler(ClosedEventArgs args)
{
    // Here you can customize your code
}
}

```

Created

Created event triggers when the component is created.

ASPX-CS

```

@using Syncfusion.Blazor.DropDowns
<SfMultiSelect TItem="GameFields" TValue="string[]" DataSource="@Games">
    <MultiSelectEvents TItem="GameFields" TValue="string[]"
        Created="@CreatedHandler"></MultiSelectEvents>
    <MultiSelectFieldSettings Text="Text" Value="ID"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
    public class GameFields
    {
        public string ID { get; set; }
        public string Text { get; set; }
    }
    private List<GameFields> Games = new List<GameFields>() {
        new GameFields(){ ID= "Game1", Text= "American Football" },
        new GameFields(){ ID= "Game2", Text= "Badminton" },
        new GameFields(){ ID= "Game3", Text= "Basketball" },
        new GameFields(){ ID= "Game4", Text= "Cricket" },
    };
    private void CreatedHandler(Object args)
    {
        // Here you can customize your code
    }
}

```

Destroyed

Destroyed event triggers when the component is destroyed.

ASPX-CS

```

@using Syncfusion.Blazor.DropDowns
<SfMultiSelect TItem="GameFields" TValue="string[]" DataSource="@Games">
    <MultiSelectEvents TItem="GameFields" TValue="string[]"
        Destroyed="@DestroyedHandler"></MultiSelectEvents>

```

```

<MultiSelectFieldSettings Text="Text" Value="ID"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
public class GameFields
{
public string ID { get; set; }
public string Text { get; set; }
}
private List<GameFields> Games = new List<GameFields>() {
new GameFields(){ ID= "Game1", Text= "American Football" },
new GameFields(){ ID= "Game2", Text= "Badminton" },
new GameFields(){ ID= "Game3", Text= "Basketball" },
new GameFields(){ ID= "Game4", Text= "Cricket" },
};
private void DestroyedHandler(Object args)
{
// Here you can customize your code
}
}

```

Focus

Focus event triggers when the input gets focus.

ASPX-CS

```

@using Syncfusion.Blazor.DropDowns
<SfMultiSelect TItem="GameFields" TValue="string[]" DataSource="@Games">
<MultiSelectEvents TItem="GameFields" TValue="string[]"
Focus="@FocusHandler"></MultiSelectEvents>
<MultiSelectFieldSettings Text="Text" Value="ID"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
public class GameFields
{
public string ID { get; set; }
public string Text { get; set; }
}
private List<GameFields> Games = new List<GameFields>() {
new GameFields(){ ID= "Game1", Text= "American Football" },
new GameFields(){ ID= "Game2", Text= "Badminton" },
new GameFields(){ ID= "Game3", Text= "Basketball" },
new GameFields(){ ID= "Game4", Text= "Cricket" },
};
private void FocusHandler(Object args)
{
// Here you can customize your code
}
}

```

OnOpen

OnOpen event triggers when the popup is opened. If you cancel this event, the popup remains closed.

ASPX-CS

```

@using Syncfusion.Blazor.DropDowns

```

```

<SfMultiSelect TItem="GameFields" TValue="string[]" DataSource="@Games">
  <MultiSelectEvents TItem="GameFields" TValue="string[]"
    OnOpen="@OnOpenHandler"></MultiSelectEvents>
  <MultiSelectFieldSettings Text="Text" Value="ID"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
public class GameFields
{
public string ID { get; set; }
public string Text { get; set; }
}
private List<GameFields> Games = new List<GameFields>() {
new GameFields(){ ID= "Game1", Text= "American Football" },
new GameFields(){ ID= "Game2", Text= "Badminton" },
new GameFields(){ ID= "Game3", Text= "Basketball" },
new GameFields(){ ID= "Game4", Text= "Cricket" },
};
private void OnOpenHandler(BeforeOpenEventArgs args)
{
// Here you can customize your code
}
}

```

OnClose

OnClose event triggers before the popup is closed. If you cancel this event, the popup will remain open.

ASPX-CS

```

@using Syncfusion.Blazor.DropDowns
<SfMultiSelect TItem="GameFields" TValue="string[]" DataSource="@Games">
  <MultiSelectEvents TItem="GameFields" TValue="string[]"
    OnClose="@OnCloseHandler"></MultiSelectEvents>
  <MultiSelectFieldSettings Text="Text" Value="ID"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
public class GameFields
{
public string ID { get; set; }
public string Text { get; set; }
}
private List<GameFields> Games = new List<GameFields>() {
new GameFields(){ ID= "Game1", Text= "American Football" },
new GameFields(){ ID= "Game2", Text= "Badminton" },
new GameFields(){ ID= "Game3", Text= "Basketball" },
new GameFields(){ ID= "Game4", Text= "Cricket" },
};
private void OnCloseHandler(PopupEventArgs args)
{
// Here you can customize your code
}
}

```

DataBound

DataBound event triggers when the data source is populated in the popup list.

ASPX-CS

```

@using Syncfusion.Blazor.DropDowns
<SfMultiSelect TItem="GameFields" TValue="string[]" DataSource="@Games">
  <MultiSelectEvents TItem="GameFields" TValue="string[]"
    DataBound="@DataBoundHandler"></MultiSelectEvents>
  <MultiSelectFieldSettings Text="Text" Value="ID"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
public class GameFields
{
public string ID { get; set; }
public string Text { get; set; }
}
private List<GameFields> Games = new List<GameFields>() {
new GameFields(){ ID= "Game1", Text= "American Football" },
new GameFields(){ ID= "Game2", Text= "Badminton" },
new GameFields(){ ID= "Game3", Text= "Basketball" },
new GameFields(){ ID= "Game4", Text= "Cricket" },
};
private void DataBoundHandler(DataBoundEventArgs args)
{
// Here you can customize your code
}
}

```

Filtering

Filtering event triggers on typing a character in the filter bar when the AllowFiltering is enabled.

ASPX-CS

```

@using Syncfusion.Blazor.DropDowns
<SfMultiSelect TItem="GameFields" TValue="string[]" AllowFiltering="true"
  DataSource="@Games">
  <MultiSelectEvents TItem="GameFields" TValue="string[]"
    Filtering="@Filteringhandler"></MultiSelectEvents>
  <MultiSelectFieldSettings Text="Text" Value="ID"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
public class GameFields
{
public string ID { get; set; }
public string Text { get; set; }
}
private List<GameFields> Games = new List<GameFields>() {
new GameFields(){ ID= "Game1", Text= "American Football" },
new GameFields(){ ID= "Game2", Text= "Badminton" },
new GameFields(){ ID= "Game3", Text= "Basketball" },
new GameFields(){ ID= "Game4", Text= "Cricket" },
};
private void Filteringhandler(FilteringEventArgs args)
{
// Here you can customize your code
}
}

```


OnActionBegin

OnActionBegin event triggers before fetching data from the remote server.

ASPX-CS

```

@using Syncfusion.Blazor.DropDowns
@using Syncfusion.Blazor.Data
<SfMultiSelect TValue="string" TItem="OrderDetails"
Query="@RemoteDataQuery">
<SfDataManager
Url="https://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/Orders"
CrossDomain="true"
Adaptor="Syncfusion.Blazor.Adaptors.ODataAdaptor"></SfDataManager>
<MultiSelectEvents TValue="string" TItem="OrderDetails"
OnActionBegin="@OnActionBeginhandler"></MultiSelectEvents>
<MultiSelectFieldSettings Text="CustomerID"
Value="CustomerID"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
public Query RemoteDataQuery = new Query().Select(new List<string> {
"CustomerID" }).Take(6).RequiresCount();
public class OrderDetails
{
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; }
}
private void OnActionBeginhandler(ActionBeginEventArgs args)
{
// Here you can customize your code
}
}

```

OnActionFailure

OnActionFailure event triggers when the data fetch request from the remote server fails.

ASPX-CS

```

@using Syncfusion.Blazor.DropDowns
@using Syncfusion.Blazor.Data
<SfMultiSelect TValue="string" TItem="OrderDetails"
Query="@RemoteDataQuery">
<SfDataManager
Url="https://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/Orders"
CrossDomain="true"
Adaptor="Syncfusion.Blazor.Adaptors.ODataAdaptor"></SfDataManager>
<MultiSelectEvents TValue="string" TItem="OrderDetails"
OnActionFailure="@OnActionFailurehandler"></MultiSelectEvents>

```

```

<MultiSelectFieldSettings Text="CustomerID"
Value="CustomerID"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
public Query RemoteDataQuery = new Query().Select(new List<string> {
"CustomerID" }).Take(6).RequiresCount();
public class OrderDetails
{
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; }
}
private void OnActionFailurehandler(Exception args)
{
// Here you can customize your code
}
}

```

OnValueSelect

OnValueSelect event triggers when a user selects an item in the popup using the mouse/tap or keyboard navigation.

ASPX-CS

```

@using Syncfusion.Blazor.DropDowns
<SfMultiSelect TItem="GameFields" TValue="string[]" AllowFiltering="true"
DataSource="@Games">
<MultiSelectEvents TItem="GameFields" TValue="string[]"
OnValueSelect="@OnValueSelecthandler"></MultiSelectEvents>
<MultiSelectFieldSettings Text="Text" Value="ID"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
public class GameFields
{
public string ID { get; set; }
public string Text { get; set; }
}
private List<GameFields> Games = new List<GameFields>() {
new GameFields(){ ID= "Game1", Text= "American Football" },
new GameFields(){ ID= "Game2", Text= "Badminton" },
new GameFields(){ ID= "Game3", Text= "Basketball" },
new GameFields(){ ID= "Game4", Text= "Cricket" },
};
private void OnValueSelecthandler(SelectEventArgs<GameFields> args)
{
// Here you can customize your code
}
}

```

Opened

Opened event triggers when the popup opens.

ASPX-CS

```

@using Syncfusion.Blazor.DropDowns
<SfMultiSelect TItem="GameFields" TValue="string[]" AllowFiltering="true"
DataSource="@Games">
  <MultiSelectEvents TItem="GameFields" TValue="string[]"
  Opened="@Openedhandler"></MultiSelectEvents>
  <MultiSelectFieldSettings Text="Text" Value="ID"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
public class GameFields
{
public string ID { get; set; }
public string Text { get; set; }
}
private List<GameFields> Games = new List<GameFields>() {
new GameFields(){ ID= "Game1", Text= "American Football" },
new GameFields(){ ID= "Game2", Text= "Badminton" },
new GameFields(){ ID= "Game3", Text= "Basketball" },
new GameFields(){ ID= "Game4", Text= "Cricket" },
};
private void Openedhandler(PopupEventArgs args)
{
// Here you can customize your code
}
}

```

ChipSelected

ChipSelected event triggers when the chip is selected.

ASPX-CS

```

@using Syncfusion.Blazor.DropDowns
<SfMultiSelect TItem="GameFields" TValue="string[]" AllowFiltering="true"
DataSource="@Games">
  <MultiSelectEvents TItem="GameFields" TValue="string[]"
  ChipSelected="@ChipSelectedhandler"></MultiSelectEvents>
  <MultiSelectFieldSettings Text="Text" Value="ID"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
public class GameFields
{
public string ID { get; set; }
public string Text { get; set; }
}
private List<GameFields> Games = new List<GameFields>() {
new GameFields(){ ID= "Game1", Text= "American Football" },
new GameFields(){ ID= "Game2", Text= "Badminton" },
new GameFields(){ ID= "Game3", Text= "Basketball" },
new GameFields(){ ID= "Game4", Text= "Cricket" },
};
}

```

```
private void ChipSelectedhandler(ChipSelectedEventArgs<GameFields> args)
{
    // Here you can customize your code
}
}
```

Cleared

Cleared event triggers after clearing all items using the clear icon.

ASPX-CS

```
@using Syncfusion.Blazor.DropDowns
<SfMultiSelect TItem="GameFields" TValue="string[]" AllowFiltering="true"
DataSource="@Games">
    <MultiSelectEvents TItem="GameFields" TValue="string[]"
Cleared="@Clearedhandler"></MultiSelectEvents>
    <MultiSelectFieldSettings Text="Text" Value="ID"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
    public class GameFields
    {
        public string ID { get; set; }
        public string Text { get; set; }
    }
    private List<GameFields> Games = new List<GameFields>() {
        new GameFields(){ ID= "Game1", Text= "American Football" },
        new GameFields(){ ID= "Game2", Text= "Badminton" },
        new GameFields(){ ID= "Game3", Text= "Basketball" },
        new GameFields(){ ID= "Game4", Text= "Cricket" },
    };
    private void Clearedhandler(MouseEventArgs args)
    {
        // Here you can customize your code
    }
}
```

OnChipTag

OnChipTag event triggers before setting the selected item as chip in the component.

ASPX-CS

```
@using Syncfusion.Blazor.DropDowns
<SfMultiSelect TItem="GameFields" TValue="string[]" AllowFiltering="true"
DataSource="@Games">
    <MultiSelectEvents TItem="GameFields" TValue="string[]"
Cleared="@Clearedhandler"></MultiSelectEvents>
    <MultiSelectFieldSettings Text="Text" Value="ID"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
    public class GameFields
    {
        public string ID { get; set; }
        public string Text { get; set; }
    }
    private List<GameFields> Games = new List<GameFields>() {
```

```

new GameFields(){ ID= "Game1", Text= "American Football" },
new GameFields(){ ID= "Game2", Text= "Badminton" },
new GameFields(){ ID= "Game3", Text= "Basketball" },
new GameFields(){ ID= "Game4", Text= "Cricket" },
};
private void Clearedhandler(MouseEventArgs args)
{
    // Here you can customize your code
}
}

```

OnValueRemove

OnValueRemove event triggers before the selected item is removed from the widget.

ASPX-CS

```

@using Syncfusion.Blazor.DropDowns
<SfMultiSelect TItem="GameFields" TValue="string[]" AllowFiltering="true"
DataSource="@Games">
    <MultiSelectEvents TItem="GameFields" TValue="string[]"
OnValueRemove="@OnValueRemovehandler"></MultiSelectEvents>
    <MultiSelectFieldSettings Text="Text" Value="ID"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
public class GameFields
{
    public string ID { get; set; }
    public string Text { get; set; }
}
private List<GameFields> Games = new List<GameFields>() {
new GameFields(){ ID= "Game1", Text= "American Football" },
new GameFields(){ ID= "Game2", Text= "Badminton" },
new GameFields(){ ID= "Game3", Text= "Basketball" },
new GameFields(){ ID= "Game4", Text= "Cricket" },
};
private void OnValueRemovehandler(RemoveEventArgs<GameFields> args)
{
    // Here you can customize your code
}
}

```

ValueRemoved

ValueRemoved event triggers after the selected item is removed from the widget.

ASPX-CS

```

@using Syncfusion.Blazor.DropDowns
<SfMultiSelect TItem="GameFields" TValue="string[]" AllowFiltering="true"
DataSource="@Games">
    <MultiSelectEvents TItem="GameFields" TValue="string[]"
ValueRemoved="@ValueRemovedhandler"></MultiSelectEvents>
    <MultiSelectFieldSettings Text="Text" Value="ID"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
public class GameFields

```

```
{
public string ID { get; set; }
public string Text { get; set; }
}
private List<GameFields> Games = new List<GameFields>() {
new GameFields(){ ID= "Game1", Text= "American Football" },
new GameFields(){ ID= "Game2", Text= "Badminton" },
new GameFields(){ ID= "Game3", Text= "Basketball" },
new GameFields(){ ID= "Game4", Text= "Cricket" },
};
private void ValueRemovedhandler(RemoveEventArgs<GameFields> args)
{
// Here you can customize your code
}
}
```

CustomValueSpecifier

CustomValueSpecifier event triggers when the CustomValue is selected.

ASPX-CS

```
@using Syncfusion.Blazor.DropDowns
<SfMultiSelect TItem="GameFields" TValue="string[]" AllowCustomValue="true"
DataSource="@Games">
<MultiSelectEvents TItem="GameFields" TValue="string[]"
CustomValueSpecifier="@CustomValueSpecifierhandler"></MultiSelectEvents>
<MultiSelectFieldSettings Text="Text" Value="ID"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
public class GameFields
{
public string ID { get; set; }
public string Text { get; set; }
}
private List<GameFields> Games = new List<GameFields>() {
new GameFields(){ ID= "Game1", Text= "American Football" },
new GameFields(){ ID= "Game2", Text= "Badminton" },
new GameFields(){ ID= "Game3", Text= "Basketball" },
new GameFields(){ ID= "Game4", Text= "Cricket" },
};
private void CustomValueSpecifierhandler(CustomValueEventArgs<GameFields>
args)
{
// Here you can customize your code
}
}
```

SelectedAll

SelectedAll event triggers after the select all process is completed.

ASPX-CS

```
@using Syncfusion.Blazor.DropDowns
<SfMultiSelect TItem="GameFields" TValue="string[]"
Mode="@VisualMode.CheckBox" ShowSelectAll="true" DataSource="@Games">
```

```

<MultiSelectEvents TItem="GameFields" TValue="string[]"
SelectedAll="@SelectedAllhandler"></MultiSelectEvents>
<MultiSelectFieldSettings Text="Text" Value="ID"></MultiSelectFieldSettings>
</SfMultiSelect>
@code {
public class GameFields
{
public string ID { get; set; }
public string Text { get; set; }
}
private List<GameFields> Games = new List<GameFields>() {
new GameFields(){ ID= "Game1", Text= "American Football" },
new GameFields(){ ID= "Game2", Text= "Badminton" },
new GameFields(){ ID= "Game3", Text= "Basketball" },
new GameFields(){ ID= "Game4", Text= "Cricket" },
};
private void SelectedAllhandler(SelectAllEventArgs<GameFields> args)
{
// Here you can customize your code
}
}

```

MultiSelect is limited with these events and new events will be added in the future based on the user requests. If the event you are looking for is not on the list, then please request [here](#).

How To

Blazor MultiSelect Dropdown List Options with Tooltip

You can achieve this behavior by using tooltip component. When the mouse is hovered over the DropDownList option, a tooltip appears with information about the hovered list item.

The following code demonstrates how to display a tooltip when hovering over the DropDownList option.

ASPX-CS

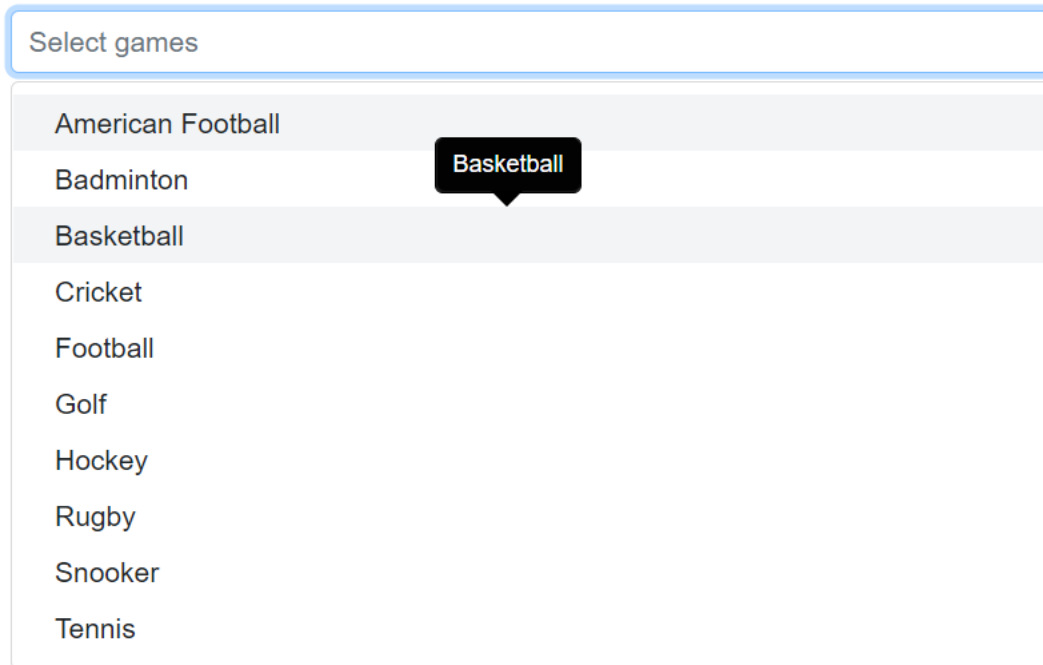
```

@using Syncfusion.Blazor.DropDowns;
@using Syncfusion.Blazor.Popups;
<SfTooltip @ref="TooltipObj" ID="Tooltip" Target=".e-list-item
.name[title]">
</SfTooltip>
<SfMultiSelect TItem="GameFields" TValue="string[]" Placeholder="Select
games" DataSource="@Games" HideSelectedItem="false">
<MultiSelectFieldSettings Text="Text" Value="ID"></MultiSelectFieldSettings>
<MultiSelectTemplates TItem="GameFields">
<ItemTemplate>
<div class="name" title="@((context as GameFields).Text)"> @((context as
GameFields).Text) </div>
</ItemTemplate>
</MultiSelectTemplates>
<MultiSelectEvents TValue="string[]" TItem="GameFields" Opened="OnOpen"
OnClose="OnClose"></MultiSelectEvents>
</SfMultiSelect>
@code {
SfTooltip TooltipObj;
public Boolean isOpen { get; set; } = false;
public class GameFields
{

```

```
public string ID { get; set; }
public string Text { get; set; }
}
private List<GameFields> Games = new List<GameFields>() {
new GameFields(){ ID= "Game1", Text= "American Football" },
new GameFields(){ ID= "Game2", Text= "Badminton" },
new GameFields(){ ID= "Game3", Text= "Basketball" },
new GameFields(){ ID= "Game4", Text= "Cricket" },
new GameFields(){ ID= "Game5", Text= "Football" },
new GameFields(){ ID= "Game6", Text= "Golf" },
new GameFields(){ ID= "Game7", Text= "Hockey" },
new GameFields(){ ID= "Game8", Text= "Rugby"},
new GameFields(){ ID= "Game9", Text= "Snooker" },
new GameFields(){ ID= "Game10", Text= "Tennis"},
};
public void OnOpen(PopupEventArgs args)
{
isOpen = true;
}
public void OnClose(PopupEventArgs args)
{
TooltipObj.CloseAsync();
}
protected override async Task OnAfterRenderAsync(bool firstRender)
{
if (isOpen)
{
await TooltipObj.RefreshAsync();
}
}
}
```

The output will be as follows.



Numeric TextBox

Getting Started with Blazor Numeric TextBox Component

This section briefly explains about how to include a [Blazor NumericTextBox](#) Component in your Blazor Server-Side and Client-Side application. You can refer to our Getting Started with [Blazor Server-Side NumericTextBox](#) and [Blazor WebAssembly NumericTextBox](#) documentation pages for configuration specifications.

To get start quickly with Blazor NumericTextBox component, you can check on this video.

{% youtube

"youtube:https://www.youtube.com/watch?v=QJGjKRXurv8"%}

Importing Syncfusion Blazor component in the application

- Install `Syncfusion.Blazor.Inputs` NuGet package to the application by using the `NuGet Package Manager`.

Please ensure to check the `Include prerelease` option for our Beta release.

- You can add the client-side resources through [CDN](#) or from [NuGet](#) package in the **HEAD** element of the `~/wwwroot/index.html` page.

HTML

```
<head>
<link href="_content/Syncfusion.Blazor.Themes/bootstrap4.css"
rel="stylesheet" />
```

```
<!-- <link  
href="https://cdn.syncfusion.com/blazor/{{version}}/styles/{{theme}}.css"  
rel="stylesheet" /> -->  
</head>
```

For Internet Explorer 11 kindly refer the polyfills. Refer the [documentation](#) for more information.

HTML

```
<head>  
<link href="_content/Syncfusion.Blazor.Themes/bootstrap4.css"  
rel="stylesheet" />  
<script  
src="https://github.com/Daddoon/Blazor.Polyfill/releases/download/3.0.1/blaz  
or.polyfill.min.js"></script>  
</head>
```

Adding component package to the application

Open `~/_Imports.razor` file and import the `Syncfusion.Blazor.Inputs` package.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
```

Add `SyncfusionBlazor` service in `Program.cs`

Open the `Program.cs` file and add services required by Syncfusion components using `builder.Services.AddSyncfusionBlazor()` method.

CSHARP

```
using Syncfusion.Blazor;  
namespace BlazorApplication  
{  
    public class Program  
    {  
        public static async Task Main(string[] args)  
        {  
            ....  
            ....  
            builder.Services.AddSyncfusionBlazor();  
            await builder.Build().RunAsync();  
        }  
    }  
}
```

To enable custom client side resource loading from CRG or CDN. You need to disable resource loading by `AddSyncfusionBlazor(true)` and load the scripts in the **HEAD** element of the `~/wwwroot/index.html` page.

HTML

```
<head>  
<script src="https://cdn.syncfusion.com/blazor/{{ site.blazorversion  
}}/syncfusion-blazor.min.js"></script>
```

```
</head>
```

Adding NumericTextBox component to the application

To initialize the NumericTextBox component add the below code to your `Index.razor` view page which is present under `~/Pages` folder.

ASPX-CS

```
<SfNumericTextBox TValue="int?" Value=10></SfNumericTextBox>
```

Run the application

After successful compilation of your application, press `F5` to run the application.

The output will be as follows.



Range validation

You can set the minimum and maximum range of values in the NumericTextBox using the [Min](#) and [Max](#) properties, so the numeric value should be in the min and max range.

The following example demonstrates range validation.

ASPX-CS

```
<SfNumericTextBox TValue="int?" Value=5 Max=100 Min=1  
Step=5></SfNumericTextBox>
```

The output will be as follows.



Formatting the value

Users can set the format of the NumericTextBox component using the [Format](#) property. The value will be displayed in the specified format, when the component is in focused out state.

The following example demonstrates format the value by using currency format value `c2`.

ASPX-CS

```
<SfNumericTextBox TValue="int?" Value=10 Format="c2"></SfNumericTextBox>
```

The output will be as follows.



Precision of numbers

You can restrict the number of decimals to be entered in the NumericTextBox by using the [Decimals](#) and [ValidateDecimalOnType](#) properties. So, you cannot enter the number whose precision is greater than the mentioned decimals.

If `ValidateDecimalOnType` is false, number of decimals will not be restricted. Else, number of decimals will be restricted while typing in the `NumericTextBox`.

ASPX-CS

```
<SfNumericTextBox TValue="double?" Value=10 ValidateDecimalOnType=true
Decimals=3 Format="n3" Placeholder="ValidateDecimalOnType enabled"
FloatLabelType="@FloatLabelType.Auto"></SfNumericTextBox>
<SfNumericTextBox TValue="double?" Value=10 Decimals=3 Format="n3"
Placeholder="ValidateDecimalOnType disabled"
FloatLabelType="@FloatLabelType.Auto"></SfNumericTextBox>
```

The output will be as follows.



You can also explore our [Blazor NumericTextBox example](#) that shows how to configure the numeric textbox in Blazor.

See Also

- [Getting Started with Syncfusion Blazor for Client-Side in .NET Core CLI](#)
- [Getting Started with Syncfusion Blazor for Server-side in Visual Studio](#)
- [Getting Started with Syncfusion Blazor for Server-Side in .NET Core CLI](#)

Data Binding in Blazor Numeric TextBox Component

This section briefly explains how to bind the value to the `NumericTextBox` component in the following different ways:

- One-way data binding
- Two-way data binding
- Dynamic value binding

One-way binding

You can bind the value to the `NumericTextBox` component directly for `Value` property as mentioned in the following code example. In one-way binding, you have to pass property or variable name along with `@` (For Ex: "`@Name`").

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfNumericTextBox TValue="int?" Value=@NumericValue></SfNumericTextBox>
<button @onclick="@UpdateValue">Update Value</button>
@code {
public int? NumericValue { get; set; } = 5;
public void UpdateValue()
{
}
```

```
NumericValue = 20;  
}  
}
```

Two-way data binding

Two-way binding can be achieved by using `bind-Value` attribute and it supports string, int, Enum, DateTime, and bool types. If component value has been changed, it will affect the all places where we bind the variable for the `bind-value` attribute.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs  
<p>NumericTextBox value is: @NumericValue</p>  
<SfNumericTextBox TValue="int?" @bind-  
Value="@NumericValue"></SfNumericTextBox>  
@code {  
    public int? NumericValue { get; set; } = 10;  
}
```

Dynamic value binding

You can change the property value dynamically by manually calling the `StateHasChanged()` method inside public event of **Blazor NumericTextBox component** only. This method notifies the component that its state has changed and queues a re-render.

There is no need to call this method for native events since it's called after any lifecycle method has been called and can also be invoked manually to trigger a re-render. Please refer the below mentioned code example.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs  
<p>NumericTextBox value is: @NumericValue</p>  
<SfNumericTextBox TValue="int?" Value="@NumericValue">  
<NumericTextBoxEvents TValue="int?" ValueChange="OnChange">  
</NumericTextBoxEvents>  
</SfNumericTextBox>  
@code {  
    public int? NumericValue { get; set; } = 12;  
    public void OnChange(Syncfusion.Blazor.Inputs.ChangeEventArgs<int?> args)  
    {  
        NumericValue = (int)args.Value;  
        StateHasChanged();  
    }  
}
```

Number Formats in Blazor Numeric TextBox Component

You can format the value of NumericTextBox using the `Format` property. The value will be displayed in the specified format when the component is in focused out state. The format string supports both the standard numeric format string and custom numeric format string.

Standard formats

From the standard numeric format, you can use the numeric related format specifiers such as **n**, **p**, and **c** in the **NumericTextBox** component. By using these format specifiers, you can achieve the percentage and currency textbox behavior also.

The following example demonstrates percentage and currency formats.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfNumericTextBox Value=0.5 Min=0 Max=1 Step=0.01 Format="p2"
Placeholder="Percentage format"
FloatLabelType="@FloatLabelType.Auto"></SfNumericTextBox>
<SfNumericTextBox TValue="int?" Value=10 Format="c2" Placeholder="Currency
format" FloatLabelType="@FloatLabelType.Auto"></SfNumericTextBox>
```

The output will be as follows.

Percentage format
50.00%

Currency format
\$10.00

Custom formats

From the custom numeric format string, you can provide any custom format by combining one or more custom specifiers.

The following examples demonstrate format the value by using currency format string **#** and **0**.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfNumericTextBox TValue="int?" Value=10 Format="###.##" Placeholder="Custom
format string #" FloatLabelType="@FloatLabelType.Always"></SfNumericTextBox>
<SfNumericTextBox TValue="int?" Value=10 Format="000.00" Placeholder="Custom
format string 0" FloatLabelType="@FloatLabelType.Always"></SfNumericTextBox>
```

The output will be as follows.

Custom format string #
10

Custom format string 0
010.00

Globalization in Blazor Numeric TextBox Component

Blazor server side

Add **UseRequestLocalization** middle-ware in **Configure** method in **Startup.cs** file to get browser Culture Info.

Refer the following code to add configuration in Startup.cs file

CSHARP

```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Localization;
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            app.UseRequestLocalization();
            ....
            ....
        }
    }
}
```

The **Localization** library allows you to localize default text content. The Numeric TextBox component has static text that can be changed to other cultures (Arabic, Deutsch, French, etc.).

In the following examples, demonstrate how to enable **Localization** for Numeric TextBox in server side Blazor samples. Here, we have used Resource file to translate the static text.

The Resource file is an XML file which contains the strings(key and value pairs) that you want to translate into different language. You can also refer Localization [link](#) to know more about how to configure and use localization in the ASP.NET Core application framework.

- Open the **Startup.cs** file and add the below configuration in the **ConfigureServices** function as follows.

CSHARP

```
using Syncfusion.Blazor;
using System.Globalization;
using Microsoft.AspNetCore.Localization;
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void ConfigureServices(IServiceCollection services)
        {
            ....
            ....
            services.AddSyncfusionBlazor();
            services.AddLocalization(options => options.ResourcesPath = "Resources");
            services.Configure<RequestLocalizationOptions>(options =>
            {
                // define the list of cultures your app will support
                var supportedCultures = new List<CultureInfo>()
```

```

{
    new CultureInfo("de")
};
// set the default culture
options.DefaultRequestCulture = new RequestCulture("de");
options.SupportedCultures = supportedCultures;
options.SupportedUICultures = supportedCultures;
options.RequestCultureProviders = new List<IRequestCultureProvider>() {
    new QueryStringRequestCultureProvider() // Here, You can also use other
    localization provider
};
});
services.AddSingleton(typeof(ISyncfusionStringLocalizer),
    typeof(SampleLocalizer));
}
}
}

```

- Then, write a **class** by inheriting **ISyncfusionStringLocalizer** interface and override the **Manager** property to get the resource file details from the application end.

CSHARP

```

using Syncfusion.Blazor;
namespace blazorInputs
{
    public class SampleLocalizer : ISyncfusionStringLocalizer
    {
        public string Get(string key)
        {
            return this.Manager.GetString(key);
        }
        public System.Resources.ResourceManager Manager
        {
            get
            {
                return blazorInputs.Resources.SyncfusionBlazorLocale.ResourceManager;
            }
        }
    }
}

```

- Add **.resx** file to Resource folder and enter the key value (Locale Keywords) in the **Name** column and the translated string in the Value column as follows.

Name	Value (in Deutsch culture)
---	---
NumericTextBox_DecrementTitle	Wert verringern
NumericTextBox_IncrementTitle	Inkrementieren Sie den Wert

- Finally, Specify the culture for Numeric TextBox using `locale` property.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfNumericTextBox TValue="int?" Value=10 Locale="de"></SfNumericTextBox>
```

Blazor WebAssembly

Internationalization library provides support for formatting and parsing the number by using the official [Unicode CLDR](#) JSON data. The NumericTextBox comes with built-in internationalization support to adapt based on culture.

The following steps explain how to render the NumericTextBox in German culture ('de-DE') in Blazor Web Assembly application.

- Open the **program.cs** file and add the below configuration in the **Builder ConfigureServices** function as follows.

CSHARP

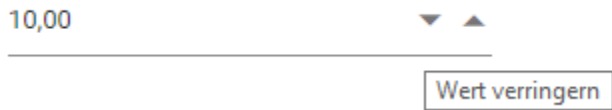
```
using Syncfusion.Blazor;
using Microsoft.AspNetCore.Builder;
namespace WebAssemblyLocale
{
    public class Program
    {
        public static async Task Main(string[] args)
        {
            ....
            ....
            builder.Services.Configure<RequestLocalizationOptions>(options =>
            {
                // Define the list of cultures your app will support
                var supportedCultures = new List<System.Globalization.CultureInfo>()
                {
                    new System.Globalization.CultureInfo("en-US"),
                    new System.Globalization.CultureInfo("de"),
                };
                // Set the default culture
                options.DefaultRequestCulture = new
                Microsoft.AspNetCore.Localization.RequestCulture("de");
                options.SupportedCultures = supportedCultures;
                options.SupportedUICultures = supportedCultures;
                options.RequestCultureProviders = new
                List<Microsoft.AspNetCore.Localization.IRequestCultureProvider>() {
                    new Microsoft.AspNetCore.Localization.QueryStringRequestCultureProvider()
                };
            });
            ....
            ....
        }
    }
}
```

- Download the required locale packages to render the Blazor NumericTextBox component with specified locale.
- To download the locale definition of Blazor components, use this [link](#).
- After downloading the blazor-locale package, copy the blazor-locale folder with required local definition file into wwwroot folder.
- By default, the blazor-locale package contains the localized text for static text present in components like button text, placeholder, tooltip, and more.
- Set the culture by using the SetCulture method.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
@inject HttpClient Http
<SfNumericTextBox TValue="int?" Value=10 Locale="de"></SfNumericTextBox>
@code {
    [Inject]
    protected IJSRuntime JsRuntime { get; set; }
    protected override async Task OnInitializedAsync()
    {
        this.JsRuntime.Sf().LoadLocaleData(await Http.GetJsonAsync<object>("blazor-locale/src/de.json")).SetCulture("de");
    }
}
```

The output will be as follows.



Customize the localized text

- You can change the localized text of particular component by editing the wwwroot/blazor-locale/src/{locale name}.json file.
- In the following code, modified the localized text of increment title and decrement title in de culture. File - wwwroot/blazor-locale/src/de.json.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
@inject HttpClient Http
<SfNumericTextBox TValue="int?" Value=10 Locale="zh"
EnableRtl="true"></SfNumericTextBox>
@code {
    [Inject]
    protected IJSRuntime JsRuntime { get; set; }
    protected override async Task OnInitializedAsync()
    {
        this.JsRuntime.Sf().LoadLocaleData(await Http.GetJsonAsync<object>("blazor-locale/src/zh.json")).SetCulture("zh");
    }
}
```

```
}

```

The output will be as follows.



Accessibility in Blazor Numeric TextBox Component

The NumericTextBox characterized with complete ARIA Accessibility support that 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 NumericTextBox uses the **spin button** role and following ARIA properties to its element based on its state.

| Property | Functionality |

| --- | --- |

| aria-live | Indicates the priority of updates to a live region |

| aria-valuemin | Specifies the minimum allowable range of the NumericTextBox. |

| aria-valuemax | Specifies the maximum allowable range of the NumericTextBox. |

| aria-disabled | Indicates disabled state of the NumericTextBox. |

| aria-readonly | Indicates the read-only state of the NumericTextBox. |

| aria-valuenow | Specifies the current value of the NumericTextBox. |

| aria-invalid | Indicates that the user input is incorrect or not within acceptable ranges. |

| aria-label | Indicates a string value that labels the NumericTextBox. |

Keyboard interaction

Keyboard interaction of the NumericTextBox component has been designed based on [WAI-ARIA Practices](#) described for the NumericTextBox and it is an alternative to mouse actions to interact with the NumericTextBox.

The following table shows shortcut keys and its corresponding usage.

| Keyboard shortcuts | Actions |

| --- | --- |

| Arrow Down | Increments the value. |

| Arrow Up | Decrements the value. |

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfNumericTextBox TValue="int?" Value=10></SfNumericTextBox>

```

The output will be as follows.



Native Events in Blazor Numeric TextBox Component

This section explains the steps to include native events and pass data to event handler in the NumericTextBox component.

Bind native events to NumericTextBox

You can access any native event by using on `<event>` attribute with a component. The attribute's value is treated as an event handler.

In the following example, the KeyPressed method is called every time the key is pressed on input.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfNumericTextBox TValue="int?"
@onkeypress='@KeyPressed'></SfNumericTextBox>
@code {
public void KeyPressed() {
Console.WriteLine("Key Pressed!");
}
}
```

Also, you can rewrite the previous example code as follows using Lambda expressions.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfNumericTextBox TValue="int?" @onkeypress="@(() => Console.WriteLine("Key
Pressed!"))"></SfNumericTextBox>
```

Pass event data to event handler

Blazor provides set of argument types to map to native events. The list of event types and event arguments are:

- Focus Events - FocusEventArgs
- Mouse Events - MouseEventArgs
- Keyboard Events - KeyboardEventArgs
- Input Events - ChangeEventArgs/EventArgs
- Touch Events – TouchEventArgs
- Pointer Events – PointerEventArgs

In the following example, the KeyPressed method is called every time any key is pressed inside input. But the message will be printed when you press "n" key.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfNumericTextBox TValue="int?" @onkeypress='@(e =>
KeyPressed(e))'></SfNumericTextBox>
@code {
public void KeyPressed(KeyboardEventArgs args)
```

```
{
  if (args.Key == "5")
  {
    Console.WriteLine("5 was pressed");
  }
}
```

Using Lambda expression also, you can pass the event data to the event handler.

[List of Native events supported](#)

| List of Native events | | |

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

| onclick | onblur | onfocus | onfocusout |

| onmousemove | onmouseover | onmouseout | onmousedown | onmouseup |

| ondblclick | onkeydown | onkeyup | onkeypress |

| ontouchend | onfocusin | onmouseup | ontouchstart |

[Events in Blazor Numeric TextBox Component](#)

This section explains the list of events of the Numeric TextBox component which will be triggered for appropriate Numeric TextBox actions.

[Blur](#)

Blur event triggers when the NumericTextBox got focus out.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfNumericTextBox TValue="int?">
  <NumericTextBoxEvents TValue="int?"
  Blur="@BlurHandler"></NumericTextBoxEvents>
</SfNumericTextBox>
@code {
  private void BlurHandler(NumericBlurEventArgs<int?> args)
  {
    // Here you can customize your code
  }
}
```

[Created](#)

Created event triggers when the NumericTextBox component is created.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfNumericTextBox TValue="int?">
  <NumericTextBoxEvents TValue="int?"
  Created="@CreatedHandler"></NumericTextBoxEvents>
</SfNumericTextBox>
@code {
  private void CreatedHandler(Object args)
  {
  }
```

```
// Here you can customize your code
}
```

Destroyed

Destroyed event triggers when the NumericTextBox component is destroyed.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfNumericTextBox TValue="int?">
<NumericTextBoxEvents TValue="int?"
Destroyed="@DestroyedHandler"></NumericTextBoxEvents>
</SfNumericTextBox>
@code {
private void DestroyedHandler(Object args)
{
// Here you can customize your code
}
}
```

Focus

Focus event triggers when the NumericTextBox got focus in.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfNumericTextBox TValue="int?">
<NumericTextBoxEvents TValue="int?"
Focus="@FocusHandler"></NumericTextBoxEvents>
</SfNumericTextBox>
@code {
private void FocusHandler(NumericFocusEventArgs<int?> args)
{
// Here you can customize your code
}
}
```

ValueChange

ValueChange event triggers when the NumericTextBox got focus in.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfNumericTextBox TValue="int?">
<NumericTextBoxEvents TValue="int?"
ValueChange="@ValueChangeHandler"></NumericTextBoxEvents>
</SfNumericTextBox>
@code {
private void ValueChangeHandler(ChangeEventArgs<int?> args)
{
// Here you can customize your code
}
}
```

Numeric TextBox is limited with these events and new events will be added in the future based on the user requests. If the event you are looking for is not on the list, then please request [here](#).

How To

Customize the UI appearance of Blazor Numeric TextBox Component

You can change the appearance of the NumericTextBox by adding custom [CssClass](#) to the component and enabling styles. Refer to the following example to change the appearance of the NumericTextBox.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfNumericTextBox TValue="int?" Value=10 CssClass="e-style"
Placeholder="Enter value"
FloatLabelType="@FloatLabelType.Always"></SfNumericTextBox>
<style>
.e-numeric.e-style .e-control.e-numerictextbox {
color: royalblue;
font-size: xx-large;
border: 0px;
}
.e-input-group.e-style.e-control-wrapper:not(.e-success):not(.e-
warning):not(.e-error):not(.e-float-icon-left), .e-float-input.e-control-
wrapper:hover:not(.e-success):not(.e-warning):not(.e-error):not(.e-
disabled):not(.e-float-icon-left) {
border-color: royalblue;
}
.e-control-wrapper.e-numeric.e-float-input.e-style .e-spin-down {
color: royalblue;
}
.e-control-wrapper.e-numeric.e-float-input.e-style .e-float-line::before {
background: royalblue;
}
.e-control-wrapper.e-numeric.e-float-input.e-style .e-float-line::after {
background: royalblue;
}
.e-control-wrapper.e-numeric.e-float-input.e-style .e-spin-up {
color: royalblue;
}
.e-control-wrapper.e-numeric.e-float-input.e-style.e-input-group .e-float-
text.e-label-top {
color: royalblue;
font-size: medium;
}
</style>
```

The output will be as follows.



Customize the up and down arrow in Blazor Numeric TextBox Component

This section explains how to change or customize spin up and down icons. You can customize spin button icons using `e-spin-up` and `e-spin-down` classes of those buttons.

You can override the default icons of `e-spin-up` and `e-spin-down` classes using the following CSS code snippets.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfNumericTextBox TValue="int?" Value=10 CssClass="e-custom"></SfNumericTextBox>
<style>
.e-numeric.e-custom .e-input-group-icon.e-spin-up:before {
content: "\e823";
color: rgba(0, 0, 0, 0.54);
}
.e-numeric.e-custom .e-input-group-icon.e-spin-down:before {
content: "\e934";
color: rgba(0, 0, 0, 0.54);
}
</style>
```

The output will be as follows.



Customize step value and hide spin buttons in Blazor NumericTextBox

The spin buttons allows you to increase or decrease the value with the predefined [Step](#) value. The visibility of spin buttons can be set using the [ShowSpinButton](#) property.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfNumericTextBox TValue="int?" Value=10 Min=10 Max=100 Step=2
ShowSpinButton=false></SfNumericTextBox>
```

The output will be as follows.



Model Binding in Blazor Numeric TextBox Component

This section demonstrates the Strongly typed extension support in NumericTextBox. The view which bind with any model is called as strongly typed view. You can bind any class as model to view. You can access model properties on that view. You can use data associated with model to render components.

In this sample, first click the submit button to post the selected value in the MaskedTextBox. When posting the null value, validation error message will be shown below the NumericTextBox.

ASPX-CS

```
@using System.ComponentModel.DataAnnotations
@using Syncfusion.Blazor.Inputs
```



```

<EditForm Model="@User">
  <DataAnnotationsValidator />
  <div asp-validation-summary="All" class="text-danger"></div>
  <div class="form-group">
    <SfNumericTextBox Placeholder='Enter value' @bind-
    Value="@User.ID"></SfNumericTextBox>
    <ValidationMessage For="@(( ) => User.ID)" />
  </div>
  <button type="submit" class="btn btn-primary">Submit</button>
</EditForm>
@code {
public Customer User = new Customer();
public class Customer
{
  [Required(ErrorMessage = "Value is required")]
  public int? ID { get; set; }
}
}

```

The output will be as follows.

PDF Viewer

Getting Started

Features in Blazor PDF Viewer Component

The [PDF Viewer control for Blazor](#) allows you to view, print, and annotate the PDF files in your Blazor applications and its key features are

- Accurate and reliable rendering of PDF pages.
- Provides easy page navigation with:
 - Thumbnail page view
 - Bookmark panel
 - Hyperlink navigation
 - Table of content navigation
- Core interactions:
 - Zooming and panning
 - Text searching
 - Text selection and copy
 - Print PDF file
- Annotate PDF with different types of annotations such as:
 - Highlight, underline, and strikeout annotation
 - Shape annotation: Rectangle, circle, polygon, line, and arrow.

- Stamp annotation: Built-in and custom stamp
- Measurement annotation
- Free text annotation
- Add a comment or note for all type of annotations
- FormFilling
- Handwritten Signature

There is a separate PDF Viewer component for Blazor server-side and Blazor WebAssembly applications.

* The `SfPdfViewerServer` control is for Blazor server-side application. This control resides with Syncfusion.Blazor.PdfViewerServer.Windows NuGet package. This server-side control is highly recommended.

* The `SfPdfViewer` control is for Blazor WebAssembly application. This control requires server-side processing to render the PDF files through web service. It resides with Syncfusion.Blazor.PdfViewer NuGet package.

Prerequisites

- [Visual Studio 2019](#)
- [.NET Core SDK 3.0.103](#)
- [Blazor VS Extension](#)

You can refer to our [Blazor PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor PDF Viewer example](#) to understand how to explain core features of PDF Viewer.

Server-side application in Blazor PDF Viewer Component

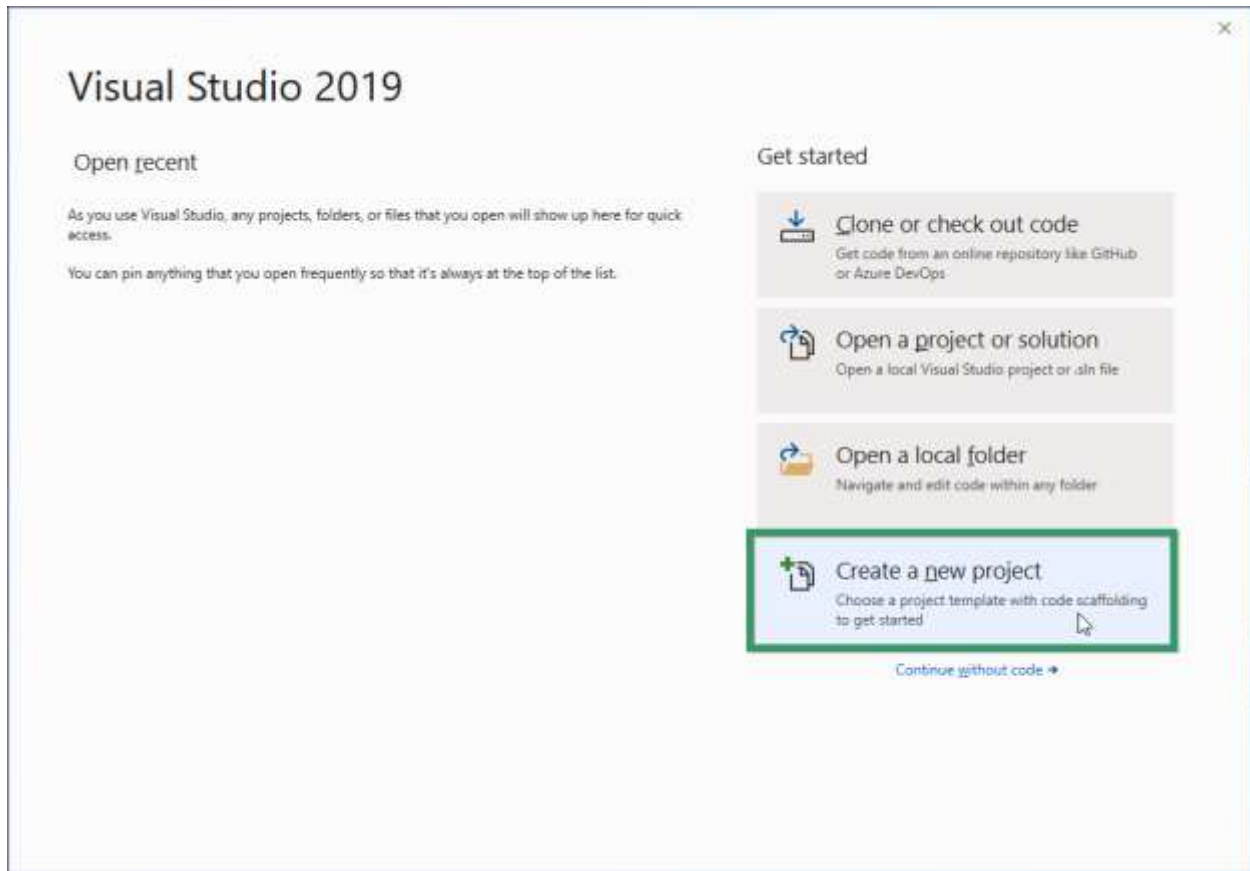
Note: There is a separate PDF Viewer component for Blazor server-side and Blazor WebAssembly applications.

* The `SfPdfViewerServer` control is for Blazor server-side application. This control resides with Syncfusion.Blazor.PdfViewerServer.Windows NuGet package. This server-side control is highly recommended.

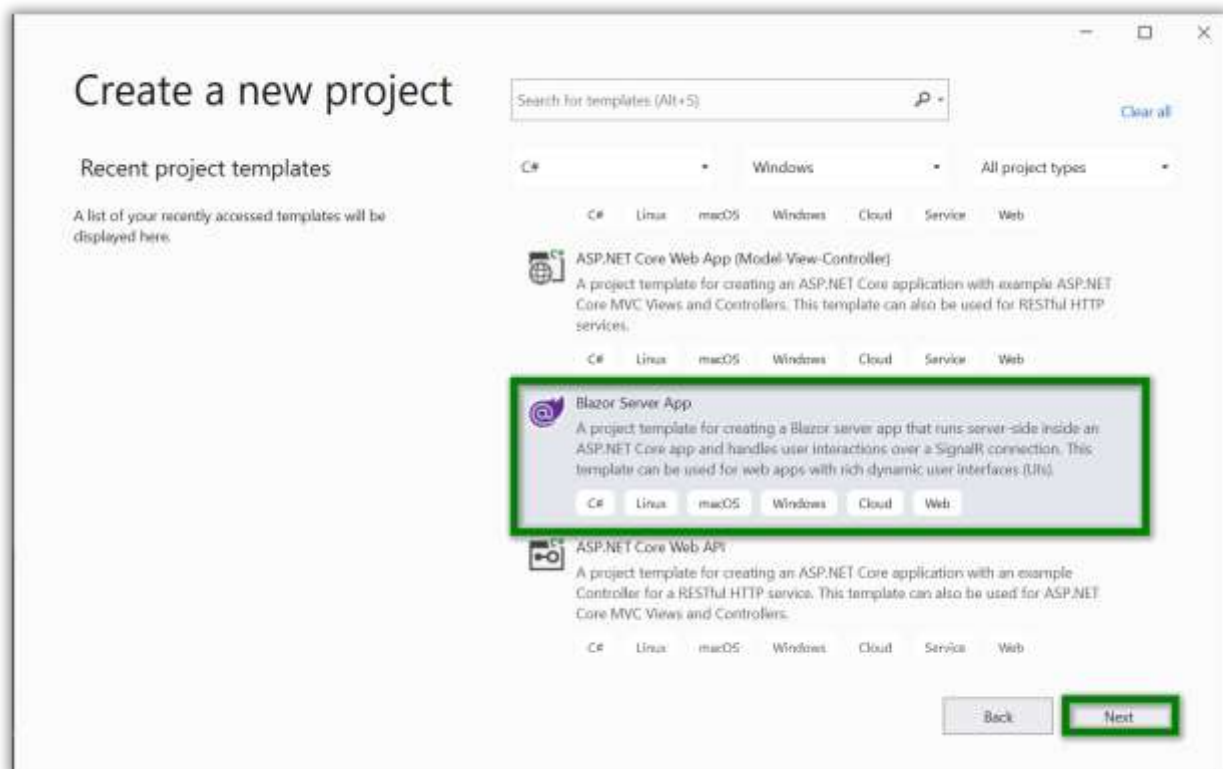
* The `SfPdfViewer` control is for Blazor WebAssembly application. This control requires server-side processing to render the PDF files through web service. It resides with Syncfusion.Blazor.PdfViewer NuGet package.

This section briefly explains how to include a PDF Viewer in your Blazor server-side application.

Step 1: Choose **Create a new project** from the Visual Studio dashboard. Click Next.



Step 2: Select **Blazor Server App** from the template, and then click **Next** button.



Step 3: Now, the project configuration window will popup. Click **Next** button to modify the additional information.

The screenshot shows a 'Configure your new project' window. At the top, it says 'Blazor Server App' with tabs for C#, Linux, macOS, Windows, Cloud, and Web. The 'C#' tab is selected. Below this, there are three input fields: 'Project name' with 'BlazorServer', 'Location' with 'C:\Users\MubeenKouserSyedShab\source\repos', and 'Solution name' with 'BlazorServer'. There is a checkbox labeled 'Place solution and project in the same directory' which is checked. At the bottom right, there are 'Back' and 'Next' buttons. The 'Next' button is highlighted with a green rectangular box.

Step 4: Select the target Framework **.NET 5.0** at the top of the Application based on your required target and then click **Create** button to create a new Blazor Server application.

The screenshot shows a window titled "Additional information" for a "Blazor Server App". At the top, there are tabs for "C#", "Linux", "macOS", "Windows", "Cloud", and "Web". Below the tabs, there are several configuration options:

- Target Framework:** A dropdown menu showing ".NET 5.0 (Current)".
- Authentication Type:** A dropdown menu showing "None".
- Configure for HTTPS:** A checkbox that is checked.
- Enable Docker:** A checkbox that is unchecked.
- Docker OS:** A dropdown menu showing "Windows".

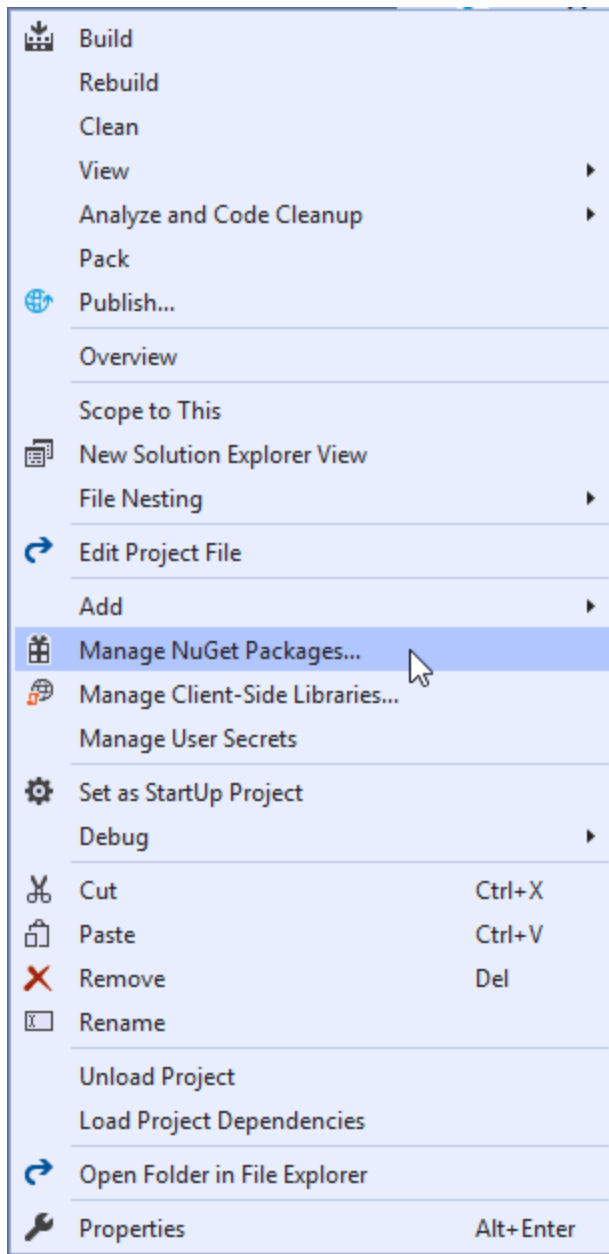
At the bottom right of the window, there are two buttons: "Back" and "Create". The "Create" button is highlighted with a green rectangular box.

Step 5: Installing Syncfusion Blazor packages in the application

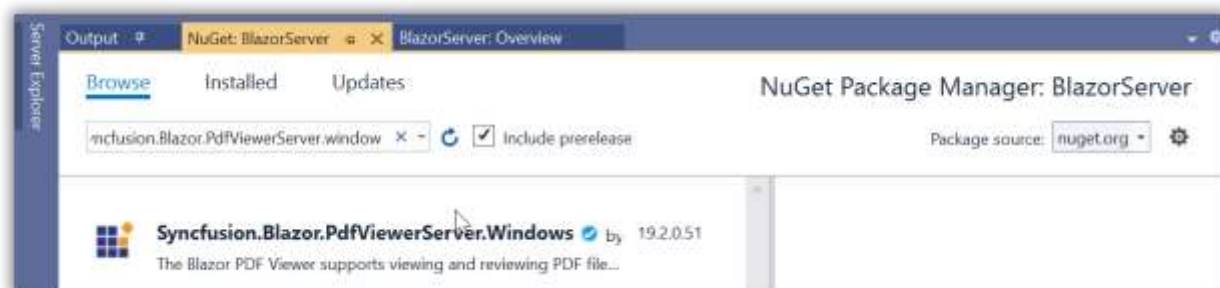
You can install the Syncfusion Blazor individual NuGet Packages in your application.

Starting with Volume 4, 2020 (v18.4.0.30) release, Syncfusion provides [individual NuGet packages](#) for our Syncfusion Blazor components. We highly recommend this new standard for your Blazor production applications. Refer to [this section](#) to know the benefits of the individual NuGet packages.

1. Install **Syncfusion.Blazor.PdfViewerServer.Windows** NuGet package to the new application using the **NuGet Package Manager**. For more details about available NuGet packages, refer to the [Individual NuGet Packages](#) documentation.
2. Right-click the project, and then select **Manage NuGet Packages**.



3. Search **Syncfusion.Blazor.PdfViewerServer.Windows** keyword in the Browse tab and install **Syncfusion.Blazor.PdfViewerServer.Windows** NuGet package in the application.



4. The `Syncfusion.Blazor.PdfViewerServer.Windows` package will be included in the newly created project once the installation process is completed.

If you are developing for Linux or Mac (OSX) operating system, use the following corresponding libraries as follows:

* For Linux, use [Syncfusion.Blazor.PdfViewerServer.Linux](#)

* For Mac (OSX), use [Syncfusion.Blazor.PdfViewerServer.OSX](#)

Step 6: Open `~/_Imports.razor` file and import the `Syncfusion.Blazor.PdfViewerServer` namespace.

CSHARP

```
@using Syncfusion.Blazor.PdfViewerServer
```

Step 7: Add the Syncfusion bootstrap4 theme in the `<head>` element of the `~/Pages/_Host.html` page.

HTML

```
<head>
....
....
<link href="_content/Syncfusion.Blazor.Themes/bootstrap4.css"
rel="stylesheet" />
</head>
```

Note: The same theme file can be referred through the CDN version by using `[https://cdn.syncfusion.com/blazor/{site.blazorversion}]/styles/bootstrap4.css`(`https://cdn.syncfusion.com/blazor/{site.blazorversion}]/styles/bootstrap4.css`).

For **Internet Explorer 11** kindly refer the polyfills. Refer the [documentation](#) for more information.

HTML

```
<head>
<link href="https://cdn.syncfusion.com/blazor/{site.blazorversion}]/styles/bootstrap4.css" rel="stylesheet" />
<script
src="https://github.com/Daddoon/Blazor.Polyfill/releases/download/3.0.1/blazor.polyfill.min.js"></script>
</head>
```

Step 8: Add SyncfusionBlazor service in `Startup.cs` file.

Open the `Startup.cs` file and add services required by Syncfusion components using `services.AddSyncfusionBlazor()` method. Add this method in the `ConfigureServices` function as follows.

CSHARP

```
using Syncfusion.Blazor;
namespace BlazorServer
{
public class Startup
{
```

```
public void ConfigureServices(IServiceCollection services)
{
    ....
    ....
    services.AddSyncfusionBlazor();
}
}
```

To enable custom WebAssembly resource loading from CRG or CDN. You need to disable resource loading by **AddSyncfusionBlazor(true)** and load the scripts in the HEAD element of the `~/Pages/_Host.cshtml` page.

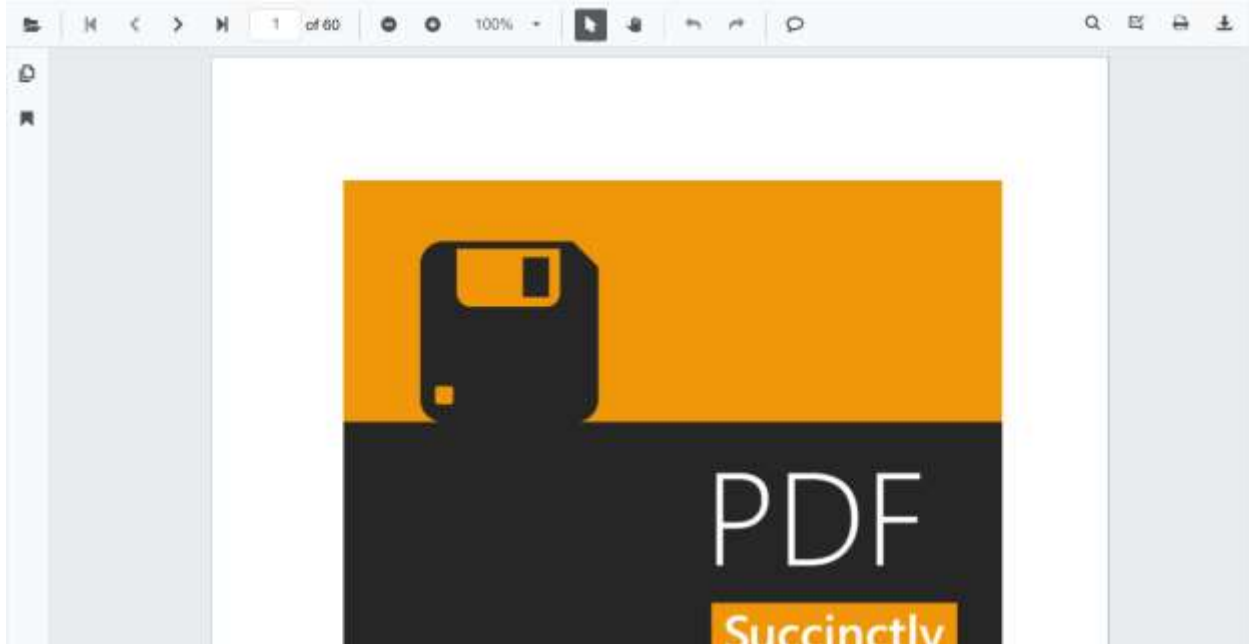
Step 9: Add the Syncfusion PDF Viewer component to the `~/Pages/Index.razor` page by using the `SfPdfViewerServer` tag. Also, you can load the PDF Viewer with a document from `wwwroot/Data` location. while initial rendering itself by specifying it in the **DocumentPath** property of the PDF Viewer component.

CSHARP

```
<SfPdfViewerServer DocumentPath="@DocumentPath" Height="500px"
Width="1060px" ></SfPdfViewerServer>
@code{
    private string DocumentPath { get; set; } =
        "wwwroot/Data/PDF_Succinctly.pdf";
}
```

If the DocumentPath property value is not provided, the PDF Viewer component will be rendered without loading the PDF document. The users can then use the open option from the toolbar to browse and open the PDF as required.

Step 10: Run the application, the PDF Viewer component will be rendered in the web browser as shown in the following screenshot.



Download the Server side application from [here](#)

You can refer to our [Blazor PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor PDF Viewer example](#) to understand how to explain core features of PDF Viewer.

WebAssembly application in Blazor PDF Viewer Component

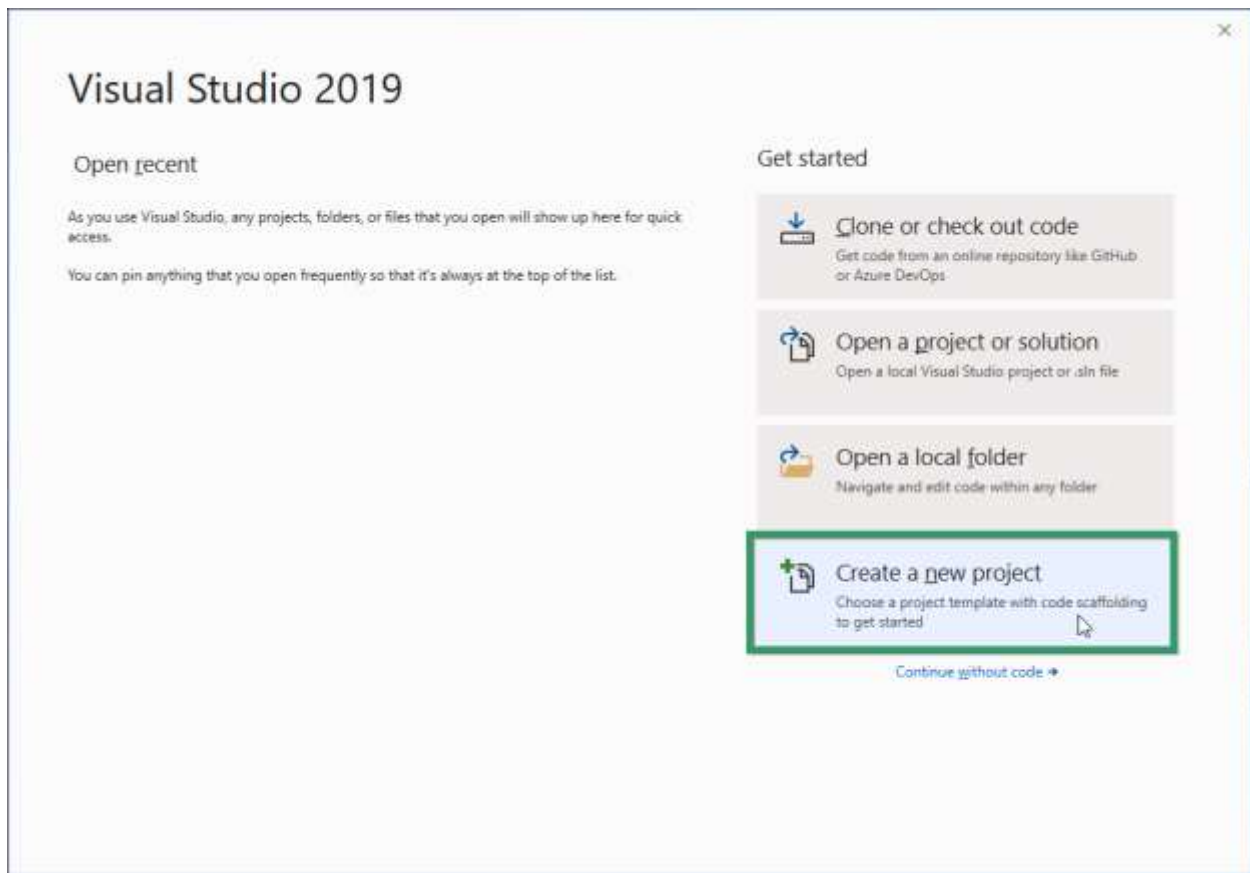
Note: There is a separate PDF Viewer component for Blazor server-side and Blazor WebAssembly applications.

* The `SfPdfViewerServer` control is for Blazor server-side application. This control resides with Syncfusion.Blazor.PdfViewerServer.Windows NuGet package. This server-side control is highly recommended.

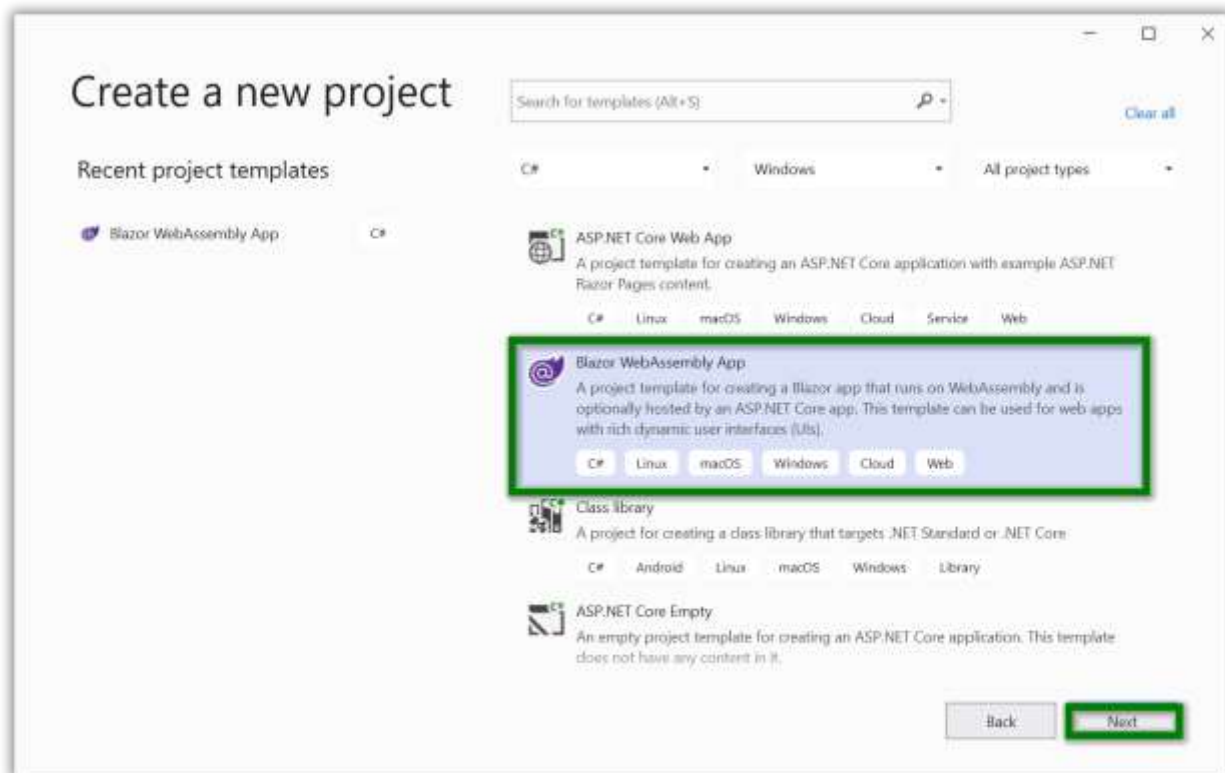
* The `SfPdfViewer` control is for Blazor WebAssembly application. This control requires server-side processing to render the PDF files through web service. It resides with Syncfusion.Blazor.PdfViewer NuGet package.

This section briefly explains how to include a PDF Viewer in your Blazor WebAssembly application.

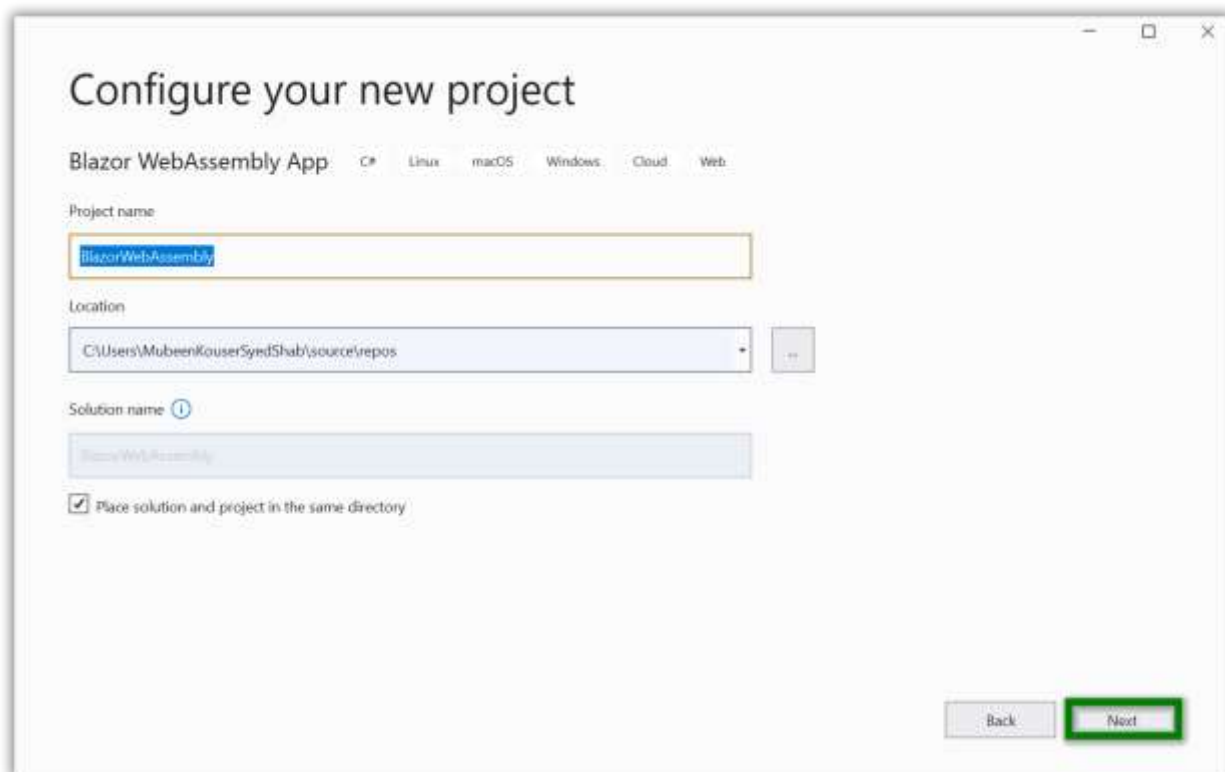
Step 1: Choose **Create a new project** from the Visual Studio dashboard. Click Next.



Step 2: Select **Blazor WebAssembly App** from the template, and then click **Next** button..



Step 3: Now, the project configuration window will popup. Click **Next** button to modify the additional information.



Step 4: Select the target Framework **.NET 5.0** at the top of the Application based on your required target and then click **Create** button to create a new Blazor WebAssembly application.

The screenshot shows a window titled "Additional information" for creating a "Blazor WebAssembly App". At the top, there are tabs for "C#", "Linux", "macOS", "Windows", "Cloud", and "Web". Below the tabs, the "Target Framework" is set to ".NET 5.0 (Current)". The "Authentication Type" is set to "None". There are three checkboxes: "Configure for HTTPS" (checked), "ASP.NET Core hosted", and "Progressive Web Application". At the bottom right, there are "Back" and "Create" buttons. The "Create" button is highlighted with a green border.

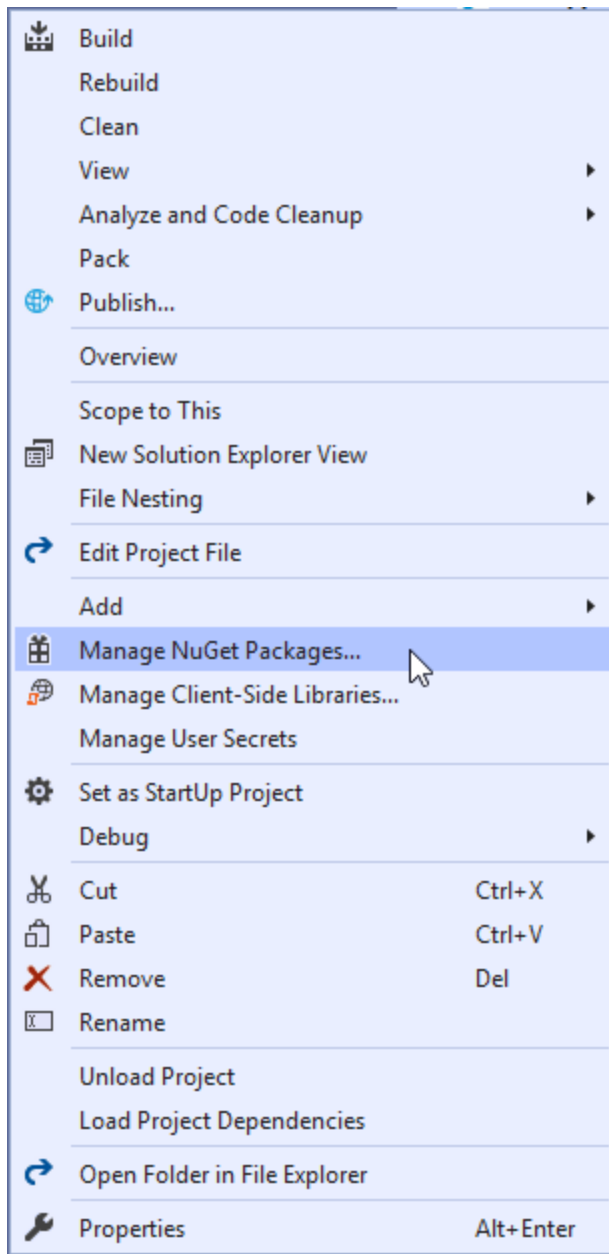
Step 5: Installing Syncfusion Blazor packages in the application

You can use any one of the below standard to install the Syncfusion Blazor library in your application.

- **Using Syncfusion Blazor individual NuGet Packages [New standard]**

Starting with Volume 4, 2020 (v18.4.0.30) release, Syncfusion provides [individual NuGet packages](#) for our Syncfusion Blazor components. We highly recommend this new standard for your Blazor production applications. Refer to [this section](#) to know the benefits of the individual NuGet packages.

1. Install **Syncfusion.Blazor.PdfViewer** NuGet package to the new application using the **NuGet Package Manager**. For more details about available NuGet packages, refer to the [Individual NuGet Packages](#) documentation.
2. Right-click the project, and then select **Manage NuGet Packages**.



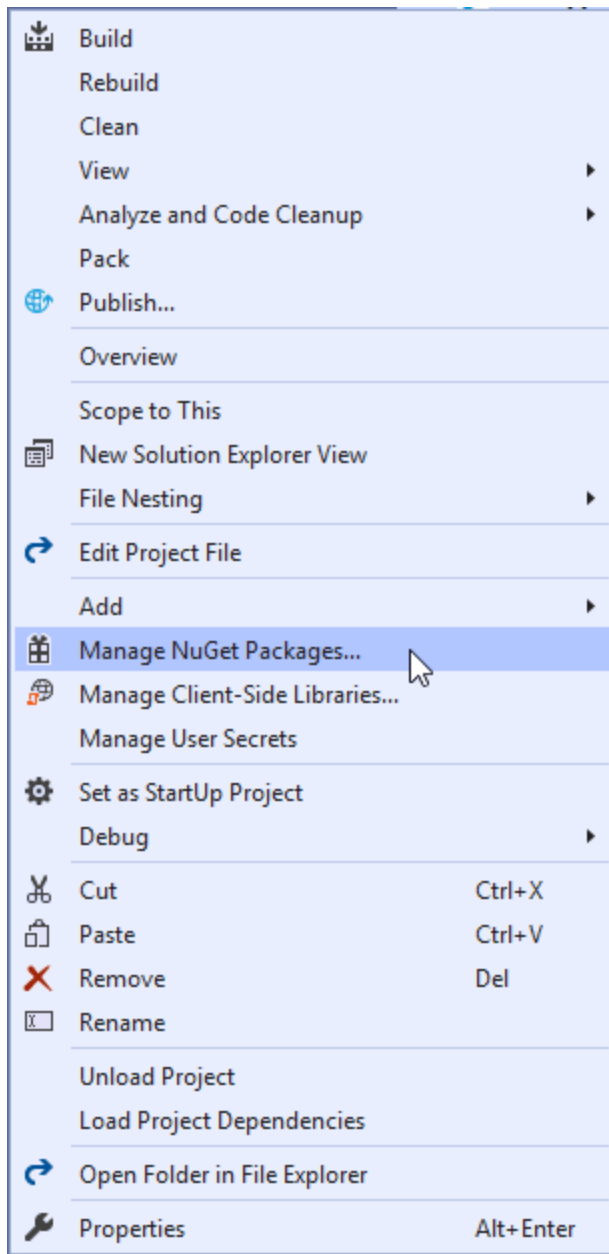
3. Search **Syncfusion.Blazor.PdfViewer** keyword in the Browse tab and install **Syncfusion.Blazor.PdfViewer** NuGet package in the application.



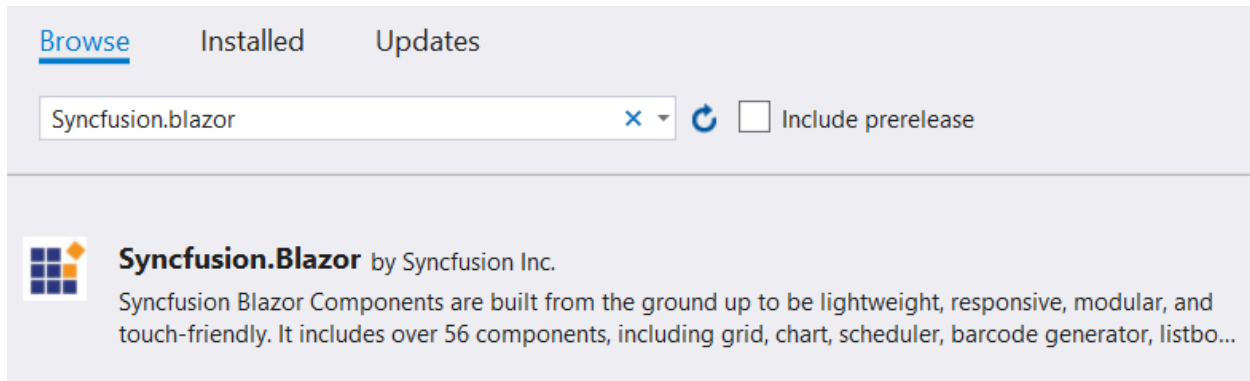
4. The Syncfusion Blazor PdfViewer package will be included in the newly created project once the installation process is completed.
 - **Using Syncfusion.Blazor NuGet Package [Old standard]**

Warning: If you prefer the above new standard (individual NuGet packages), then skip this section. Using both old and new standards in the same application will throw ambiguous compilation errors.

1. Now, install **Syncfusion.Blazor** NuGet package to the newly created application by using the **NuGet Package Manager**. Right-click the project and then select Manage NuGet Packages.



2. Search **Syncfusion.Blazor** keyword in the Browse tab and install **Syncfusion.Blazor** NuGet package in the application.



3. The Syncfusion Blazor package will be installed in the project once the installation process is completed.

Step 6: Open `~/_Imports.razor` file and import the `Syncfusion.Blazor.PdfViewer` when new standard is followed or import the `Syncfusion.Blazor` namespace when old standard is followed.

CSHARP

```
@using Syncfusion.Blazor.PdfViewer
```

(or)

CSHARP

```
@using Syncfusion.Blazor
```

Step 7: Add the Syncfusion bootstrap4 theme in the `<head>` element of the `~/wwwroot/index.html` page.

When `Syncfusion.Blazor.PdfViewer` is used, use the following code.

HTML

```
<head>
...
...
<link href="_content/Syncfusion.Blazor.Themes/bootstrap4.css"
rel="stylesheet" />
</head>
```

When `Syncfusion.Blazor` is used, use the following code.

HTML

```
<head>
...
...
<link href="_content/Syncfusion.Blazor/styles/bootstrap4.css"
rel="stylesheet" />
</head>
```


Note: The same theme file can be referred through the CDN version by using https://cdn.syncfusion.com/blazor/{{ site.blazorversion }}/styles/bootstrap4.css(https://cdn.syncfusion.com/blazor/{{ site.blazorversion }}/styles/bootstrap4.css).

For **Internet Explorer 11** kindly refer the polyfills. Refer the [documentation](#) for more information.

HTML

```
<head>
<link href="https://cdn.syncfusion.com/blazor/{{ site.blazorversion
}}/styles/bootstrap4.css" rel="stylesheet" />
<script
src="https://github.com/Daddoon/Blazor.Polyfill/releases/download/3.0.1/blaz
or.polyfill.min.js"></script>
</head>
```

Step 8: Open the ~/Program.cs file and register the Syncfusion Blazor Service.

CSHARP

```
using Syncfusion.Blazor;
namespace BlazorWebAssembly
{
    public class Program
    {
        public static async Task Main(string[] args)
        {
            ....
            ....
            builder.Services.AddSyncfusionBlazor();
            await builder.Build().RunAsync();
        }
    }
}
```

We can disable the dynamic script loading and refer to the scripts from the application end by using the IgnoreScriptIsolation parameter in AddSyncfusionBlazor() at the program.cs. For more details, please refer here for [how to refer custom/CDN resources](#).

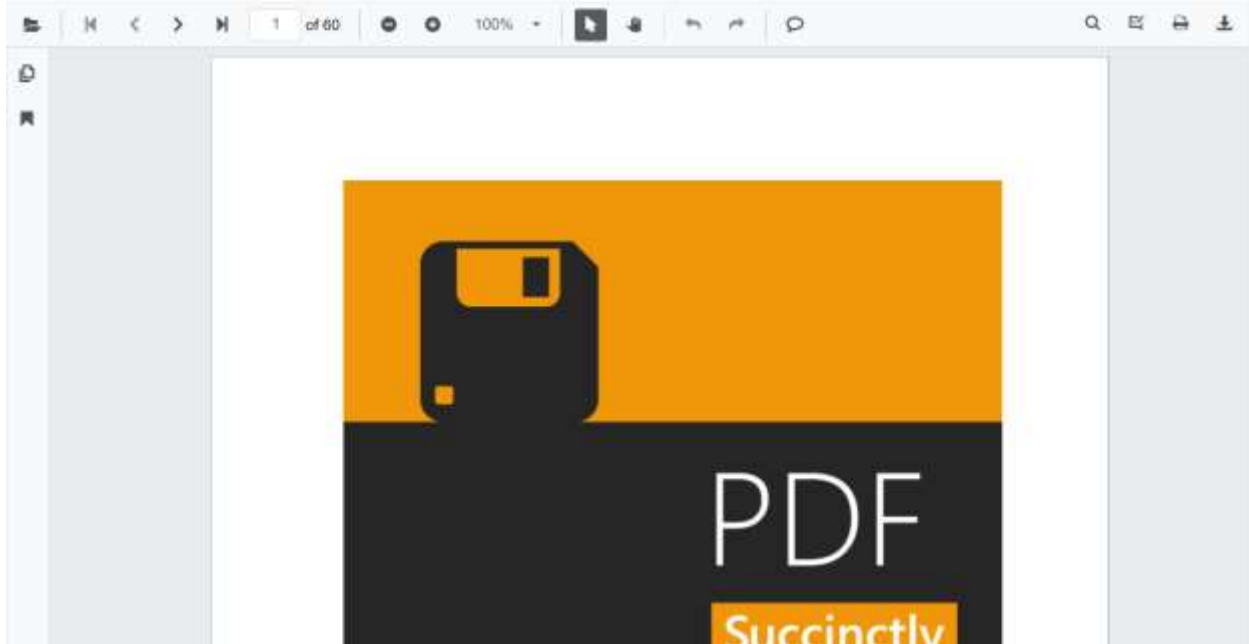
Step 9: Now, add the PDF Viewer (WebAssembly) component to the ~/Pages/Index.razor page.

CSHARP

```
@using Syncfusion.Blazor.PdfViewer
<SfPdfViewer DocumentPath="PDF_Succinctly.pdf"
ServiceUrl="https://ej2services.syncfusion.com/production/web-
services/api/pdfviewer" Height="500px" Width="1060px"></SfPdfViewer>
```

Since Syncfusion PDF Viewer (Blazor WebAssembly) control depends on server-side processing to render the PDF files, it is mandatory to create a web service as mentioned [here](#)

Step 10: Run the application, the PDF Viewer component will be rendered in the web browser as shown in the following screenshot.



Download the WebAssembly application from [here](#)

You can refer to our [Blazor PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor PDF Viewer example](#) to understand how to explain core features of PDF Viewer.

Open PDF files in PDF Viewer for Blazor from various storage location

You might need to open and view the PDF files from various location. In this section, you can find information about how to open PDF files from URL, Cloud, database, local file system, and as base64 string.

Opening a PDF from URL

If you have your PDF files in the web, you can open it in the viewer using URL. The following code example explains how to open PDF file from URL.

ASPX-CS

```
@using Syncfusion.Blazor.PdfViewerServer
<SfPdfViewerServer DocumentPath="@DocumentPath" Width="1060px"
Height="500px" />
@code {
    public string DocumentPath { get; set; }
    protected override void OnInitialized()
    {
        string Url =
            "https://s3.amazonaws.com/files2.syncfusion.com/dtsupport/directtrac/general
            /pd/HTTP_Succinctly-1719682472.pdf";
        System.Net.WebClient webClient = new System.Net.WebClient();
        byte[] byteArray = webClient.DownloadData(Url);
        DocumentPath = "data:application/pdf;base64," +
            Convert.ToBase64String(byteArray);
    }
}
```

Opening a PDF from Cloud

You can open the PDF file from Cloud storage.

The following code example shows how to open and load the PDF file stored in Azure Blob Storage.

ASPX-CS

```
@using Azure.Storage.Blobs
@using Azure.Storage.Blobs.Specialized
@using System.IO;
@using Syncfusion.Blazor.PdfViewerServer
<SfPdfViewerServer DocumentPath="@DocumentPath" Width="1060px"
Height="500px" />
@code {
public string DocumentPath { get; set; }
public void blobLoad(MouseEventArgs args)
{
//Connection String of Storage Account
string connectionString = "Here Place Your Connection string";
BlobServiceClient blobServiceClient = new
BlobServiceClient(connectionString);
//Container Name
string containerName = "pdf-file";
//File Name to be loaded into Syncfusion PDF Viewer
string fileName = "Python_Succinctly.pdf";
BlobContainerClient blobContainerClient =
blobServiceClient.GetBlobContainerClient(containerName);
BlockBlobClient blockBlobClient =
blobContainerClient.GetBlockBlobClient(fileName);
MemoryStream memoryStream = new MemoryStream();
blockBlobClient.DownloadTo(memoryStream);
DocumentPath = "data:application/pdf;base64," +
Convert.ToBase64String(memoryStream.ToArray());
}
}
```

Note: The **Azure.Storage.Blobs** NuGet package must be installed in your application to use the previous code example.

You can open the PDF file from Azure File Storage using the following code example.

ASPX-CS

```
@using Azure.Storage.Files.Shares
@using System.IO;
@using Syncfusion.Blazor.PdfViewerServer
<SfPdfViewerServer DocumentPath="@DocumentPath" Width="1060px"
Height="500px" />
@code {
public string DocumentPath { get; set; } =
"wwwroot/data/PDF_Succinctly.pdf";
protected override void OnInitialized()
{
//Connection String of Storage Account
string connectionString = "Here Place Your Connection string";
string shareName = "pdfdocument";
//File Name to be loaded into Syncfusion PDF Viewer
```

```
string filePath = "Hive_Succinctly.pdf";
ShareFileClient shareFileClient = new ShareFileClient(connectionString,
shareName, filePath);
Stream stream = shareFileClient.OpenRead();
byte[] bytes;
using (var memoryStream = new MemoryStream())
{
    stream.CopyTo(memoryStream);
    bytes = memoryStream.ToArray();
}
DocumentPath = "data:application/pdf;base64," +
Convert.ToBase64String(bytes);
}
```

Note: The **Azure.Storage.Files.Shares** NuGet package must be installed in your application to use the previous code example.

Opening a PDF from database

The following code example shows how to open the PDF file in viewer from SQL Server database.

ASPX-CS

```
@using Syncfusion.Blazor.PdfViewerServer
<SfPdfViewerServer DocumentPath="@DocumentPath" Width="1060px"
Height="500px" />
@code {
    public string DocumentPath { get; set; }
    protected override void OnInitialized()
    {
        string documentID = "PDF Succinctly";
        string connectionString = "Data
Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=C:\\\\Database.mdf;";
        System.Data.SqlClient.SqlConnection connection = new
        System.Data.SqlClient.SqlConnection(connectionString);
        //Searches for the PDF document from the database
        string query = "select Data from Table where DocumentName = '" + documentID
        + "'";
        System.Data.SqlClient.SqlCommand command = new
        System.Data.SqlClient.SqlCommand(query, connection);
        connection.Open();
        System.Data.SqlClient.SqlDataReader read = command.ExecuteReader();
        read.Read();
        //Reads the PDF document data as byte array from the database
        byte[] byteArray = (byte[])read["Data"];
        DocumentPath = "data:application/pdf;base64," +
        Convert.ToBase64String(byteArray);
    }
}
```

Note: The **System.Data.SqlClient** package must be installed in your application to use the previous code example. You need to modify the connectionString variable in the previous code example as per the connection string of your database.

Opening a PDF from file system

There is an UI option in built-in toolbar to open the PDF file from local file system. If you want to achieve the same functionality when design your own toolbar, you can use the following code example to load and open the PDF file. In this sample, the Syncfusion's Uploader control is used for Blazor.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
@using Syncfusion.Blazor.PdfViewerServer
<SfUploader ID="UploadFiles" AllowedExtensions=".pdf,.PDF">
  <UploaderEvents FileSelected="onsuccess"></UploaderEvents>
</SfUploader>
<SfPdfViewerServer @ref="@Viewer" Width="1060px" Height="500px" />
@code {
  SfPdfViewerServer Viewer;
  public void onsuccess(SelectedEventArgs action)
  {
    string filePath = action.FilesData[0].RawFile.ToString();
    Viewer.Load(filePath, null);
  }
}
```

Opening a PDF from base64 data

The following code snippet explains how the PDF file can be loaded in PDF Viewer as base64 string.

ASPX-CS

```
@using Syncfusion.Blazor.PdfViewerServer
<SfPdfViewerServer ID="pdfviewer" DocumentPath="@DocumentPath"
  Width="1060px" Height="500px"/>
@code {
  static byte[] byteArray =
  System.IO.File.ReadAllBytes("wwwroot/data/PDF_Succinctly.pdf");
  static string base64String = Convert.ToBase64String(byteArray);
  public string DocumentPath { get; set; } = "data:application/pdf;base64," +
  base64String;
}
```

Opening a PDF from stream

You can load a PDF file from stream in PDF Viewer by converting the stream into a base64 string. The following code sample explains how the PDF file can be loaded in PDF Viewer from stream.

CSHARP

```
@using Syncfusion.Blazor.PdfViewerServer
@using System.IO;
<SfPdfViewerServer ID="pdfviewer" DocumentPath="@DocumentPath"
  Width="1060px" Height="500px"/>
@code{
  public string DocumentPath { get; set; }
  protected override void OnInitialized()
  {
    string filePath = "wwwroot/data/PDF_Succinctly.pdf";
    FileStream fileStream = new FileStream(filePath, System.IO.FileMode.Open,
    System.IO.FileAccess.Read);
```

```

MemoryStream stream = new MemoryStream();
fileStream.CopyTo(stream);
byte[] byteArray = stream.ToArray();
string base64String = Convert.ToBase64String(byteArray);
DocumentPath = "data:application/pdf;base64," + base64String;
}
}

```

Note: You can refer to our [Blazor PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor PDF Viewer example](#) to understand how to explain core features of PDF Viewer.

Saving PDF file in Blazor PDF Viewer Component

After editing the PDF file with various annotation tools, you will need to save the updated PDF to the server, database, or local file system.

Save PDF file to Server

You might need to save the PDF file back to the server. The following code example shows how to save the updated PDF file to server.

ASPX-CS

```

@using Syncfusion.Blazor.PdfViewerServer
@using Syncfusion.Blazor.Buttons
@using System.IO
<SfButton OnClick="OnClick">Save</SfButton>
<SfPdfViewerServer DocumentPath="@DocumentPath" @ref="viewer" Width="1060px"
Height="500px"></SfPdfViewerServer>
@code{
SfPdfViewerServer viewer;
public async void OnClick(MouseEventArgs args)
{
byte[] data = await viewer.GetDocument();
//PDF document file stream
Stream stream = new MemoryStream(data);
using (var fileStream = new
FileStream(@"wwwroot\Data\PDF_Succinctly_Updated.pdf", FileMode.Create,
FileAccess.Write))
{
//Saving the new file in root path of application
stream.CopyTo(fileStream);
fileStream.Close();
}
stream.Close();
}
public string DocumentPath { get; set; } =
"wwwroot/Data/PDF_Succinctly.pdf";
}

```

Save PDF file to Database

If you have plenty of PDF files stored in database and you want to save the updated PDF file back to the database, use the following code example.

ASPX-CS

```

@using Syncfusion.Blazor.PdfViewerServer
@using Syncfusion.Blazor.Buttons
@using System.Data.SqlClient
<SfButton OnClick="OnClick">Save</SfButton>
<SfPdfViewerServer DocumentPath="@DocumentPath" @ref="viewer" Width="1060px"
Height="500px">
</SfPdfViewerServer>
@code{
SfPdfViewerServer viewer;
public async void OnClick(MouseEventArgs args)
{
string DocumentName = "PDF_Succinctly";
byte[] data = await viewer.GetDocument();
string connectionString = @"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=D:\database.mdf;";
string queryStmt = "Update PDFFiles SET Content = @Content where
DocumentName = '" + DocumentName + "'";
using (SqlConnection con = new SqlConnection(connectionString))
{
using (SqlCommand cmd = new SqlCommand(queryStmt, con))
{
SqlParameter param = cmd.Parameters.Add("@Content",
System.Data.SqlDbType.VarBinary);
param.Value = data;
con.Open();
cmd.ExecuteNonQuery();
con.Close();
}
}
}
private string DocumentPath { get; set; } =
"wwwroot/Data/PDF_Succinctly.pdf";
}

```

Download PDF file as a copy

In the built-in toolbar, you have an option to download the updated PDF to the local file system, you can use it to download the PDF file.

If you want to achieve the same behavior through your custom UI, use the following code example.

ASPX-CS

```

@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.PdfViewerServer
<SfButton @onclick="OnClick">Download</SfButton>
<SfPdfViewerServer @ref="@viewer" Height="500px" Width="1060px"
DocumentPath="@DocumentPath" />
@code{
SfPdfViewerServer viewer;
public void OnClick(MouseEventArgs args)
{
viewer.Download();
}
public string DocumentPath { get; set; } =
"wwwroot/data/PDF_Succinctly.pdf";
}

```

You can refer to our [Blazor PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor PDF Viewer example](#) to understand how to explain core features of PDF Viewer.

Toolbar Customization in Blazor PDF Viewer Component

The PDF Viewer comes with a powerful built-in toolbar with the following important options;

- Open PDF file
- Page navigation
- Magnification
- Pan tool
- Text selection tool
- Text search
- Print
- Download
- Undo and redo
- Various annotation tools
- Bookmark panel
- Thumbnail panel



The following section explains, how to show or hide the toolbar and toolbar item.

Show or hide toolbar

At times, you might need to create your own toolbar. In that case, you need to hide the built-in toolbar. For that customization purpose, the PDF Viewer control provides an option to show or hide the main (top) toolbar either by using the `EnableToolbar` property or `ShowToolbar` method.

The following code snippet explains how to show or hide toolbar using the `EnableToolbar` property.

ASPX-CS

```
@using Syncfusion.Blazor.PdfViewerServer
<SfPdfViewerServer EnableToolbar="false" Width="1060px" Height="500px">
</SfPdfViewerServer>
```

The following code snippet explains how to show or hide toolbar using the ShowToolbar method.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.PdfViewerServer
<SfButton @onclick="OnClick">Hide Toolbar</SfButton>
<SfPdfViewerServer Width="1060px" Height="500px" @ref="@pdfViewer"
DocumentPath="@DocumentPath">
</SfPdfViewerServer>
@code{
SfPdfViewerServer pdfViewer;
public string DocumentPath { get; set; } =
"wwwroot/data/PDF_Succinctly.pdf";
public void OnClick(MouseEventArgs args)
{
pdfViewer.ShowToolbar(false);
}
}
```

Show or hide navigation toolbar

Navigation toolbar is the side bar, which contains the options to expand and collapse the bookmark panel and page thumbnail panel. This navigation toolbar visibility can be toggled either by using the EnableNavigationToolbar property or ShowNavigationToolbar method.

The following code snippet explains how to show or hide navigation toolbar using the EnableNavigationToolbar property.

ASPX-CS

```
@using Syncfusion.Blazor.PdfViewerServer
<SfPdfViewerServer EnableNavigationToolbar="false" Width="1060px"
Height="500px">
</SfPdfViewerServer>
```

The following code snippet explains how to show or hide navigation toolbar using the ShowNavigationToolbar method.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.PdfViewerServer
<SfButton @onclick="OnClick">Hide Navigation Toolbar</SfButton>
<SfPdfViewerServer EnableNavigationToolbar="true" Width="1060px"
Height="500px" @ref="@pdfViewer" DocumentPath="@DocumentPath">
</SfPdfViewerServer>
@code{
SfPdfViewerServer pdfViewer;
```

```
public string DocumentPath { get; set; } =
"wwwroot/data/PDF_Succinctly.pdf";
public void OnClick(MouseEventArgs args)
{
pdfViewer.ShowNavigationToolbar(false);
}
}
```

Show or hide the toolbar item

You can show or hide the toolbar items using the “PdfViewerToolbarSettings” class. The following code snippet explains how to show only the desired options in the toolbar. The resultant PDF Viewer’s toolbar will have these options - Open file, magnification tools, comment tool, and download option.

ASPX-CS

```
@using Syncfusion.Blazor.PdfViewerServer
@using Syncfusion.Blazor.PdfViewer
<SfPdfViewerServer Width="1060px" Height="500px"
DocumentPath="@DocumentPath" ToolbarSettings="@ToolbarSettings">
</SfPdfViewerServer>
@code{
public string DocumentPath { get; set; } =
"wwwroot/data/PDF_Succinctly.pdf";
public PdfViewerToolbarSettings ToolbarSettings = new
PdfViewerToolbarSettings() {
ToolbarItems = new List<ToolbarItem>()
{
ToolbarItem.PageNavigationTool,
ToolbarItem.MagnificationTool,
ToolbarItem.CommentTool,
ToolbarItem.SelectionTool,
ToolbarItem.PanTool,
ToolbarItem.UndoRedoTool,
ToolbarItem.CommentTool,
ToolbarItem.AnnotationEditTool,
ToolbarItem.SearchOption,
ToolbarItem.PrintOption,
ToolbarItem.DownloadOption
}
};
}
```

You can refer to our [Blazor PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor PDF Viewer example](#) to understand how to explain core features of PDF Viewer.

Navigation in Blazor PDF Viewer Component

You can navigate between pages in Syncfusion PDF Viewer in the following ways:

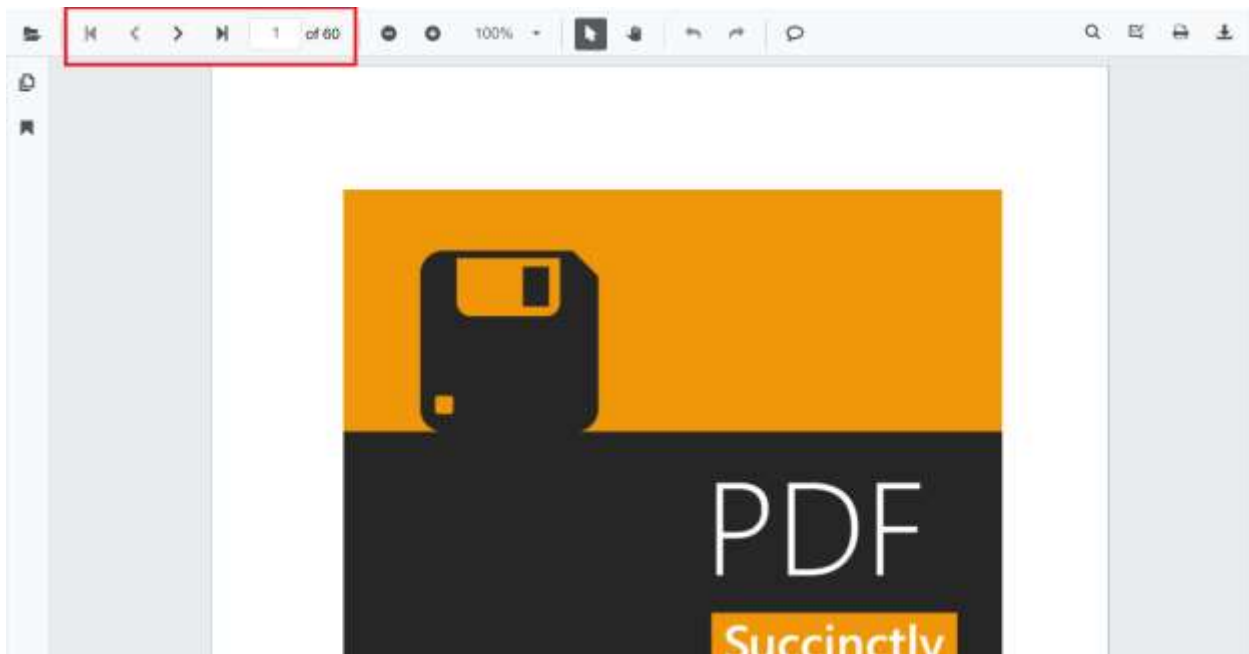
- Scroll through the pages.
- Click Go to pages in the built-in toolbar.
- Click the desired bookmark in bookmark pane.
- Click the desired page in thumbnail pane.

- Click hyperlink and table of contents.

Page navigation

The built-in toolbar of PDF Viewer contains the following page navigation tools:

- **First Page:** Navigates you to the first page in the document.
- **Last Page:** Navigates you to the last page in the document.
- **Next Page:** Scrolls forwards through pages, one page at a time.
- **Go To:** Allows you to quickly jump to the desired page number.
- **Previous Page:** Scrolls backwards through pages, one page at a time.



You can enable or disable the page navigation option in PDF Viewer default toolbar by setting the `EnableNavigation` property.

ASPX-CS

```
@using Syncfusion.Blazor.PdfViewerServer
<SfPdfViewerServer Width="1060px" Height="500px"
DocumentPath="@DocumentPath" EnableNavigation="false" />
@code{
public string DocumentPath { get; set; } =
"wwwroot/data/PDF_Succinctly.pdf";
}
```

Also, you can programmatically perform page navigation as follows.

ASPX-CS

```
@using Syncfusion.Blazor.PdfViewerServer
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.Inputs
<div style="display:inline-block">
```

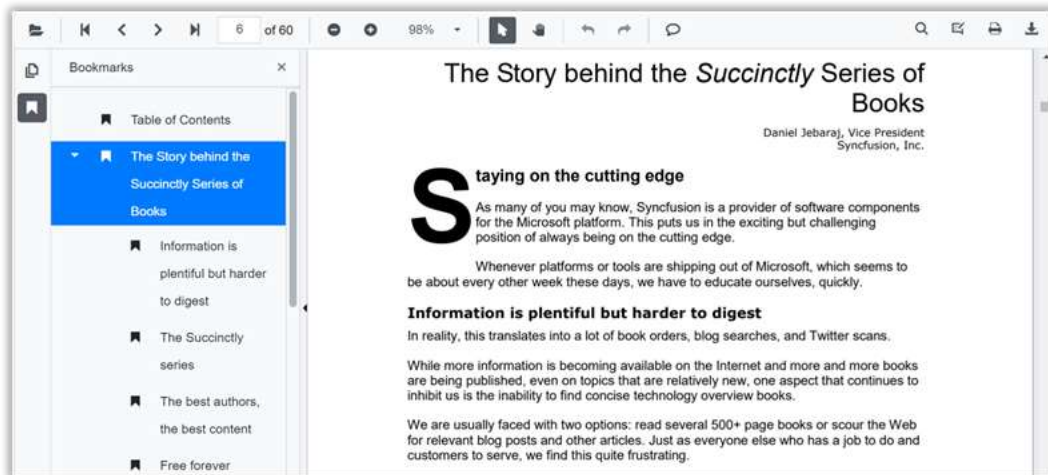
```

<SfButton OnClick="OnFirstPageClick">Go To First Page</SfButton>
</div>
<div style="display:inline-block">
<SfButton OnClick="OnLastPageClick">Go To Last Page</SfButton>
</div>
<div style="display:inline-block">
<SfButton OnClick="OnNextPageClick">Go To Next Page</SfButton>
</div>
<div style="display:inline-block">
<SfTextBox @ref="@TextBox"></SfTextBox>
</div>
<div style="display:inline-block;">
<SfButton OnClick="OnPageClick">Go To Page</SfButton>
</div>
<div style="display:inline-block">
<SfButton OnClick="OnPreviousPageClick">Go To Previous Page</SfButton>
</div>
<SfPdfViewerServer Width="1060px" Height="500px"
DocumentPath="@DocumentPath" @ref="@Viewer" />
@code{
SfPdfViewerServer Viewer;
SfTextBox TextBox;
public string DocumentPath { get; set; } =
"wwwroot/data/PDF_Succinctly.pdf";
public void OnFirstPageClick(MouseEventArgs args)
{
Viewer.GoToFirstPage();
}
public void OnLastPageClick(MouseEventArgs args)
{
Viewer.GoToLastPage();
}
public void OnNextPageClick(MouseEventArgs args)
{
Viewer.GoToNextPage();
}
public void OnPageClick(MouseEventArgs args)
{
double pageIndex = double.Parse(TextBox.Value.ToString());
Viewer.GoToPage(pageIndex);
}
public void OnPreviousPageClick(MouseEventArgs args)
{
Viewer.GoToPreviousPage();
}
}

```

Bookmark navigation

The bookmarks saved in PDF files are loaded and listed in the bookmark pane (in the left navigation pane). The users can jump to areas of interest by clicking the desired bookmark easily.



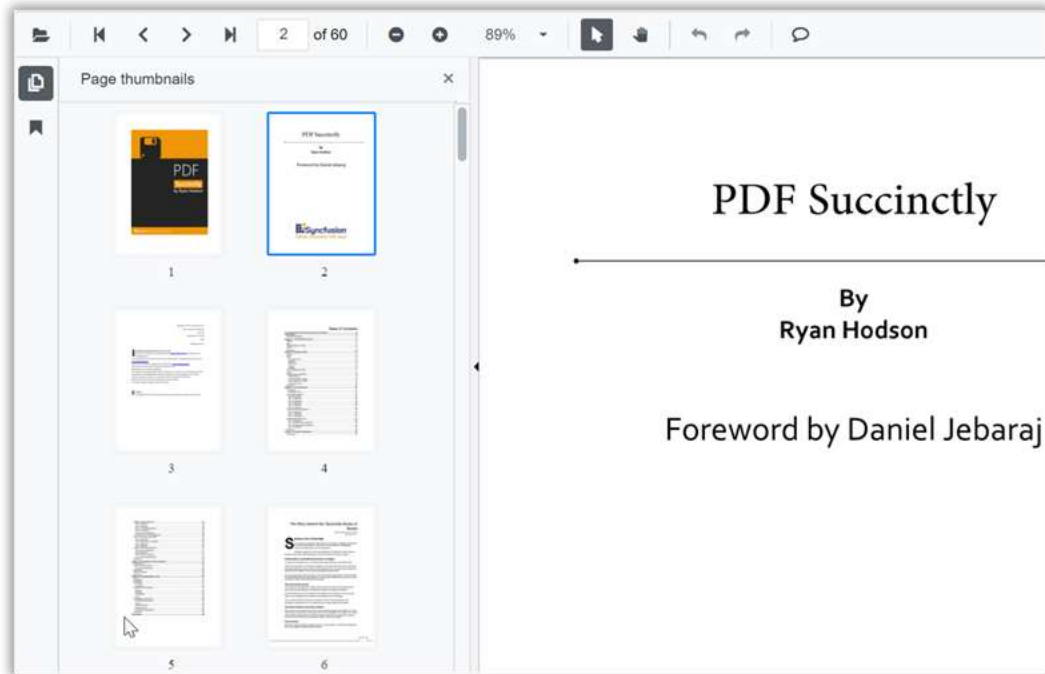
You can enable or disable the bookmark navigation pane by setting the `EnableBookmark` property.

ASPX-CS

```
@using Syncfusion.Blazor.PdfViewerServer
<SfPdfViewerServer Width="1060px" Height="500px"
DocumentPath="@DocumentPath" EnableBookmarkPanel="true"/>
@code{
public string DocumentPath { get; set; } =
"wwwroot/data/PDF_Succinctly.pdf";
}
```

Page thumbnail navigation

Page thumbnails is the miniature representation of actual pages in the PDF files. This feature displays thumbnails of the pages and represents a link to the respective pages. Clicking a page thumbnail will display the respective page in the document view.



You can enable or disable the thumbnail navigation pane by setting the `EnableThumbnail` property.

ASPX-CS

```
@using Syncfusion.Blazor.PdfViewerServer
<SfPdfViewerServer Width="1060px" Height="500px"
DocumentPath="@DocumentPath" EnableThumbnailPanel="true"/>
@code{
public string DocumentPath { get; set; } =
"wwwroot/data/PDF_Succinctly.pdf";
}
```

Hyperlink navigation

Hyperlink navigation features enables navigation to the URLs (website links) in a PDF file.

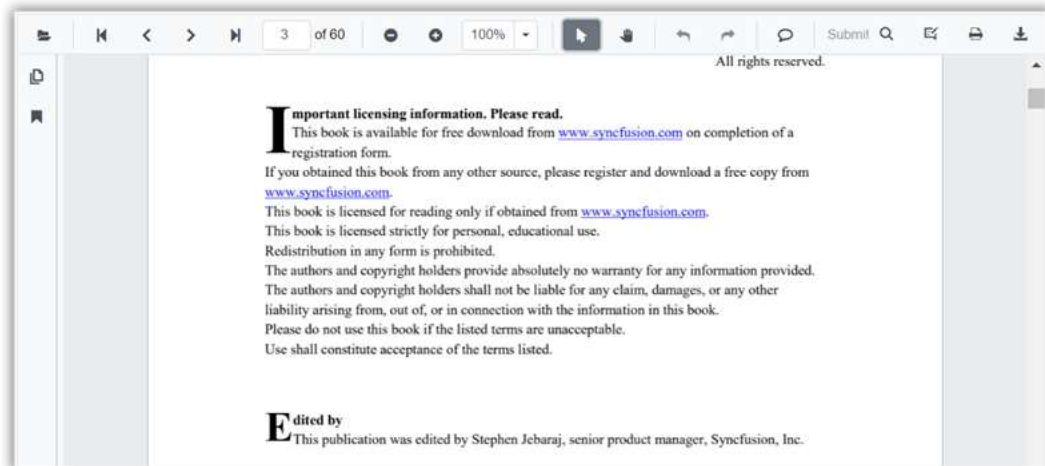
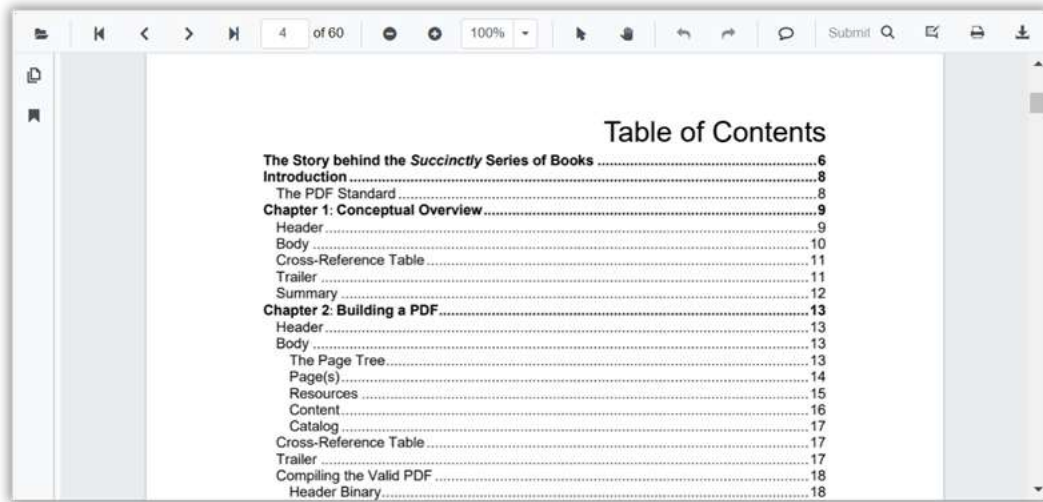


Table of content navigation

Table of contents navigation allows users to navigate to different parts of a PDF file that are listed in the table of contents section.



You can enable or disable both hyperlink and table of content navigation by setting the `EnableHyperlink` property.

ASPX-CS

```
@using Syncfusion.Blazor.PdfViewerServer
<SfPdfViewerServer Width="1060px" Height="500px"
DocumentPath="@DocumentPath" EnableHyperlink="true"/>
@code{
public string DocumentPath { get; set; } =
"wwwroot/data/PDF_Succinctly.pdf";
}
```

You can set the target attribute for a hyperlink in PDF Viewer using the `HyperlinkOpenState` property.

ASPX-CS

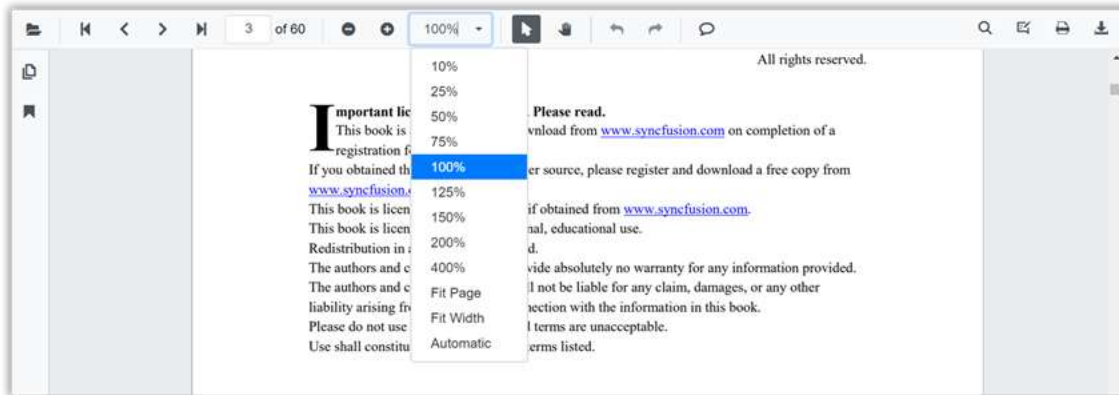
```
@using Syncfusion.Blazor.PdfViewer
@using Syncfusion.Blazor.PdfViewerServer
<SfPdfViewerServer Width="1060px" Height="500px"
DocumentPath="@DocumentPath" EnableHyperlink="true"
HyperlinkOpenState="LinkTarget.NewTab"/>
@code{
public string DocumentPath { get; set; } =
"wwwroot/data/PDF_Succinctly.pdf";
}
```

You can refer to our [Blazor PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor PDF Viewer example](#) to understand how to explain core features of PDF Viewer.

Magnification in Blazor PDF Viewer Component

The built-in toolbar of PDF Viewer contains the following zooming options:

- **Zoom In:** Increases the zoom value (document magnification) from the current value by preset levels.
- **Zoom Out:** Decreases the zoom value from the current value by preset levels.
- **Zoom To:** Magnifies the pages to the specified zoom value.
- **Fit Page:** Fits the page entirely in the available document view port size.
- **Fit Width:** Fits the page to the width of the view port size.



You can enable or disable the magnification option in PDF Viewer default toolbar by setting the `EnableMagnification` property.

ASPX-CS

```
@using Syncfusion.Blazor.PdfViewerServer
<SfPdfViewerServer Width="1060px" Height="500px"
DocumentPath="@DocumentPath" EnableMagnification="true"/>
@code{
public string DocumentPath { get; set; } =
"wwwroot/data/PDF_Succinctly.pdf";
}
```

Also, you can programmatically perform zooming operations as follows.

ASPX-CS

```
@using Syncfusion.Blazor.PdfViewerServer
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.Inputs
<div style="display:inline-block">
<SfButton OnClick="OnZoomInClick">Zoom In</SfButton>
</div>
<div style="display:inline-block">
<SfButton OnClick="OnZoomOutClick">Zoom Out</SfButton>
</div>
<div style="display:inline-block">
<SfTextBox @ref="@TextBox"></SfTextBox>
</div>
<div style="display:inline-block">
<SfButton OnClick="OnZoomClick">Zoom</SfButton>
</div>
<div style="display:inline-block;">
<SfButton OnClick="OnFitPageClick">Fit To Page</SfButton>
```



```

</div>
<div style="display:inline-block">
<SfButton OnClick="OnFitWidthClick">Fit To Width</SfButton>
</div>
<SfPdfViewerServer Width="1060px" Height="500px"
DocumentPath="@DocumentPath" @ref="@Viewer" />
@code{
SfPdfViewerServer Viewer;
SfTextBox TextBox;
public string DocumentPath { get; set; } =
"wwwroot/data/PDF_Succinctly.pdf";
public void OnZoomInClick(MouseEventArgs args)
{
Viewer.ZoomIn();
}
public void OnZoomOutClick(MouseEventArgs args)
{
Viewer.ZoomOut();
}
public void OnFitPageClick(MouseEventArgs args)
{
Viewer.FitToPage();
}
public void OnZoomClick(MouseEventArgs args)
{
double zoomValue = double.Parse(TextBox.Value.ToString());
Viewer.ZoomTo(zoomValue);
}
public void OnFitWidthClick(MouseEventArgs args)
{
Viewer.FitToWidth();
}
}

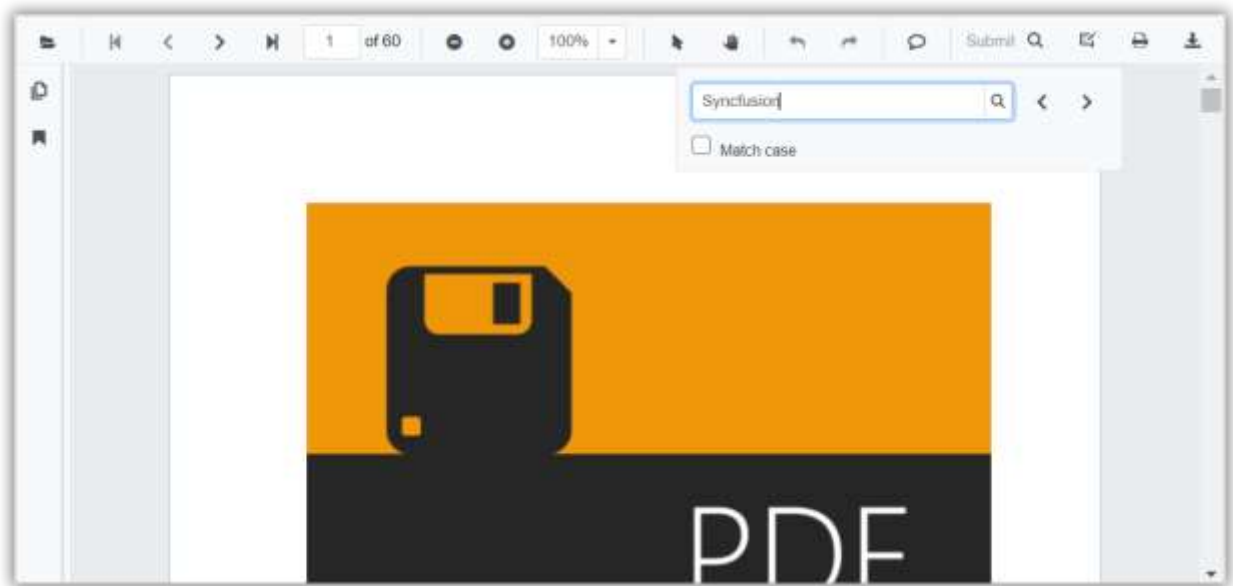
```

PDF Viewer can support zoom value ranges from 50% to 400%.

You can refer to our [Blazor PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor PDF Viewer example](#) to understand how to explain core features of PDF Viewer.

Text Search in Blazor PDF Viewer Component

You can find the specified text content in the PDF document using the built-in options provided with the toolbar. On initiating the search operation, the control searches for the specified text and highlights all the occurrences in the pages.



You can enable or disable the text search by setting the `EnableTextSearch` API.

ASPX-CS

```
@using Syncfusion.Blazor.PdfViewerServer
<SfPdfViewerServer Width="1060px" Height="500px"
DocumentPath="@DocumentPath" EnableTextSearch="true"/>
@code{
public string DocumentPath { get; set; } =
"wwwroot/data/PDF_Succinctly.pdf";
}
```

Also, you can programmatically perform search operation as given in the following code example.

ASPX-CS

```
@using Syncfusion.Blazor.PdfViewerServer
@using Syncfusion.Blazor.Buttons
<div style="display:inline-block">
<SfButton OnClick="OnSearchClick">Search Text</SfButton>
</div>
<div style="display:inline-block">
<SfButton OnClick="OnSearchNext">Search Next</SfButton>
</div>
<div style="display:inline-block">
<SfButton OnClick="OnSearchPrevious">Search Previous</SfButton>
</div>
<div style="display:inline-block">
<SfButton OnClick="OnCancelSearch">Cancel Search</SfButton>
</div>
<SfPdfViewerServer Width="1060px" Height="500px"
DocumentPath="@DocumentPath" @ref="@Viewer" />
@code{
SfPdfViewerServer Viewer;
public string DocumentPath { get; set; } =
"wwwroot/data/PDF_Succinctly.pdf";
}
```

```

public void OnSearchClick(MouseEventArgs args)
{
    //Here PDF is to be searched from the loaded document
    Viewer.SearchText("pdf", false);
}
public void OnSearchNext(MouseEventArgs args)
{
    Viewer.SearchNext();
}
public void OnSearchPrevious(MouseEventArgs args)
{
    Viewer.SearchPrevious();
}
public void OnCancelSearch(MouseEventArgs args)
{
    Viewer.CancelTextSearch();
}
}

```

You can refer to our [Blazor PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor PDF Viewer example](#) to understand how to explain core features of PDF Viewer.

Interaction mode in Blazor PDF Viewer Component

The built-in toolbar of PDF Viewer contains the following two interaction options:

- Selection mode
- Panning mode

Selection mode

In this mode, the text selection can be performed in the PDF document loaded in the PDF Viewer. It allows users to select and copy text from the PDF files. This is helpful for copying and sharing text content.

The panning and scrolling of the pages by touch cannot be performed in this mode.

You can enable or disable text selection by setting the `EnableTextSelection` property.

ASPX-CS

```

@using Syncfusion.Blazor.PdfViewerServer
<SfPdfViewerServer Width="1060px" Height="500px"
DocumentPath="@DocumentPath" EnableTextSelection="true"/>
@code{
    public string DocumentPath { get; set; } =
    "wwwroot/data/PDF_Succinctly.pdf";
}

```

Panning mode

In this mode, the panning and scrolling of the pages can be performed in the PDF document loaded in the PDF Viewer, but the text selection cannot be performed.

You can change the interaction mode of PDF Viewer using the `InteractionMode` property.

ASPX-CS

```

@using Syncfusion.Blazor.PdfViewer
@using Syncfusion.Blazor.PdfViewerServer
<SfPdfViewerServer Width="1060px" Height="500px"
DocumentPath="@DocumentPath" InteractionMode="InteractionMode.Pan"/>
@code{
public string DocumentPath { get; set; } =
"wwwroot/data/PDF_Succinctly.pdf";
}

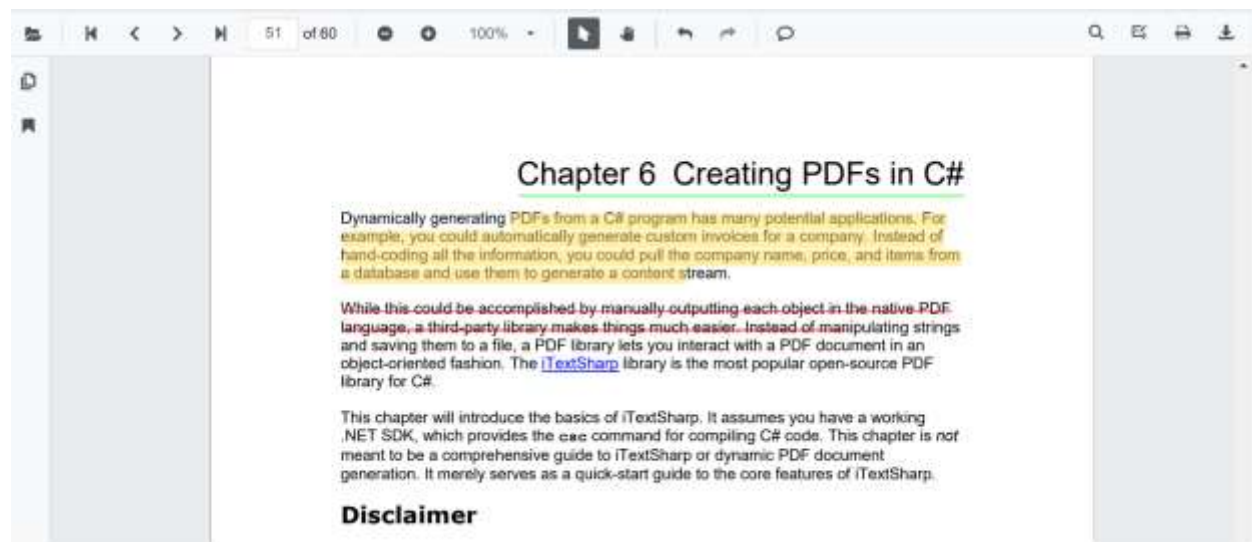
```

You can refer to our [Blazor PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor PDF Viewer example](#) to understand how to explain core features of PDF Viewer.

Annotations

Text markup annotations in Blazor PDF Viewer Component

The PDF Viewer control provides the options to add, edit, and delete text markup annotations such as highlight, underline, and strikethrough annotations in the PDF document.



Adding text markup annotation to the PDF Document

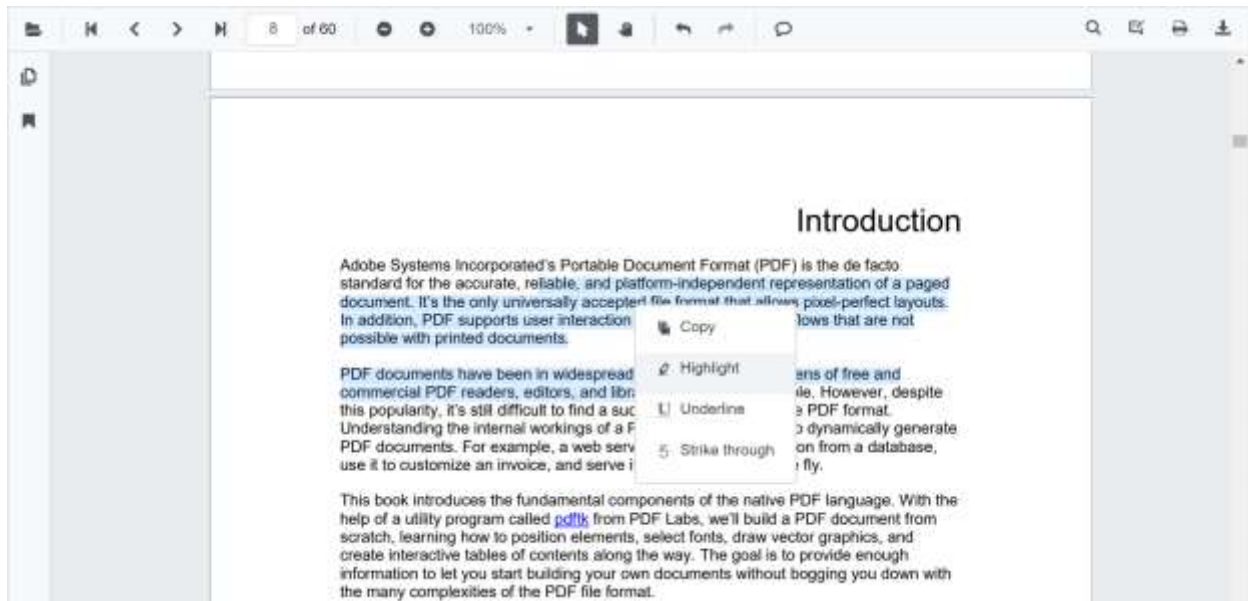
Text Markup annotations can be added to the PDF document using the annotation toolbar and by using context menu.

Highlight a text

There are two ways to highlight a text in the PDF document.

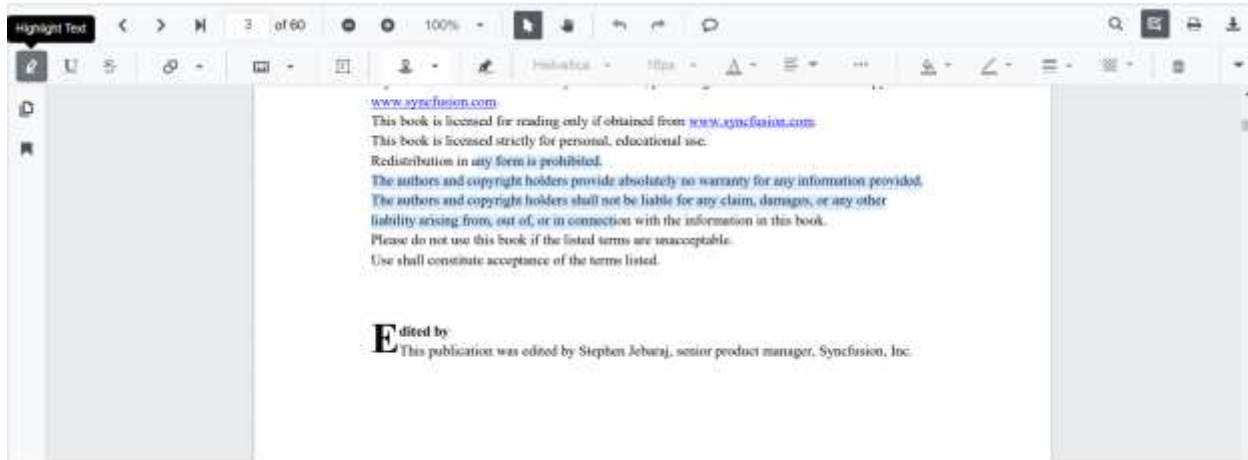
Using the context menu.

- Select a text in the PDF document and right-click it
- Select **Highlight** option in the context menu that appears



Using the annotation toolbar.

- Click the **Edit Annotation** button in the PDF Viewer toolbar. A toolbar appears below it.
- Select the **Highlight** button in the annotation toolbar. It enables the highlight mode.
- Select the text and the highlight annotation will be added.
- You can also select the text and apply the highlight annotation using the **Highlight** button



In the pan mode, if the highlight mode is entered, the PDF Viewer control will switch to text select mode to enable the text selection for highlighting the text.

Refer to the following code snippet to switch to highlight mode

ASPX-CS

```
@using Syncfusion.Blazor.PdfViewerServer
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.PdfViewer
<SfButton OnClick="OnClick">Click Here</SfButton>
<SfPdfViewerServer DocumentPath="@DocumentPath" @ref="viewer" Width="1060px"
Height="500px">
```

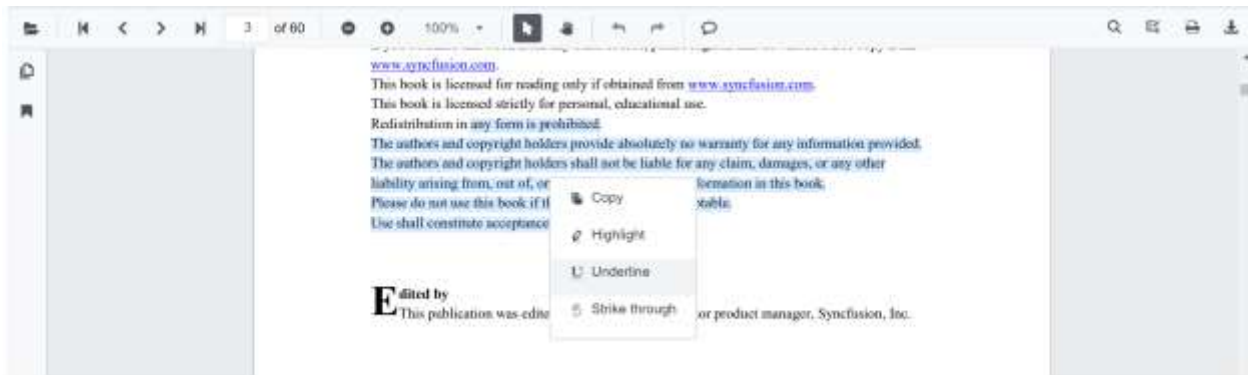
```
</SfPdfViewerServer>
@code{
SfPdfViewerServer viewer;
public void OnClick(MouseEventArgs args)
{
viewer.SetAnnotationMode(AnnotationType.Highlight);
}
private string DocumentPath { get; set; } =
"wwwroot/Data/PDF_Succinctly.pdf";
}
```

Underline a text

There are two ways to underline a text in the PDF document:

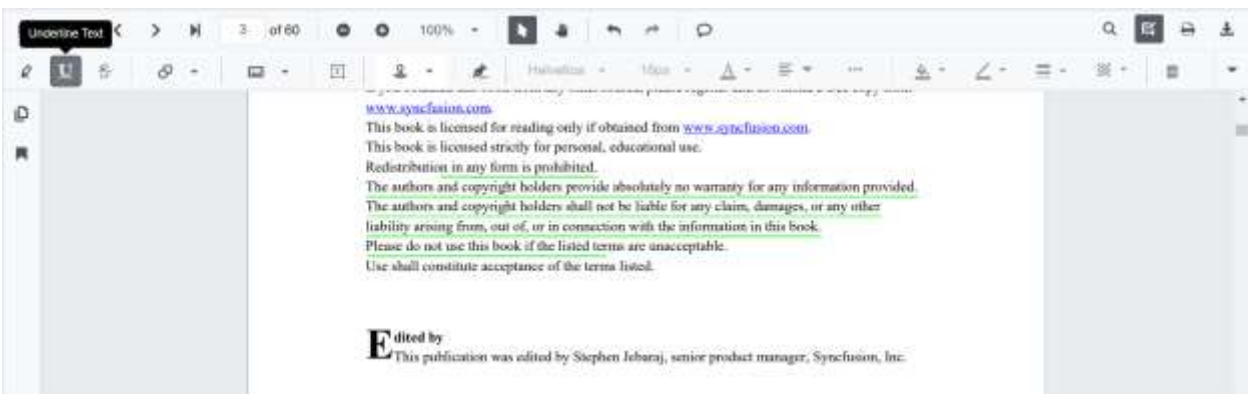
Using the context menu.

- Select a text in the PDF document and right-click it
- Select **Underline** option in the context menu that appears.



Using the annotation toolbar.

- Click the **Edit Annotation** button in the PDF Viewer toolbar. A toolbar appears below it.
- Select the **Underline** button in the annotation toolbar. It enables the underline mode.
- Select the text and the underline annotation will be added.
- You can also select the text and apply the underline annotation using the **Underline** button



In the pan mode, if the underline mode is entered, the PDF Viewer control will switch to text select mode to enable the text selection for underlining the text.

Refer to the following code snippet to switch to underline mode

ASPX-CS

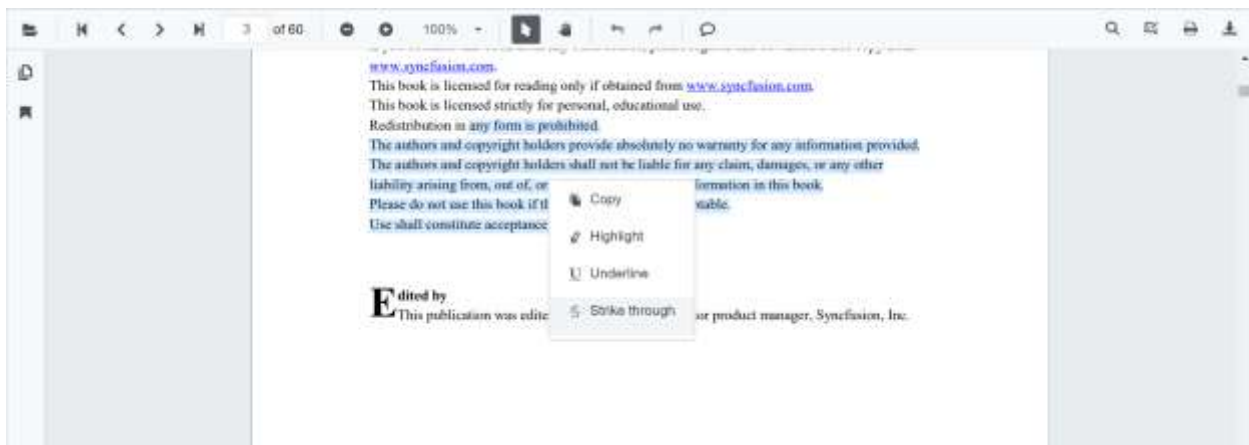
```
@using Syncfusion.Blazor.PdfViewerServer
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.PdfViewer
<SfButton OnClick="OnClick">Click Here</SfButton>
<SfPdfViewerServer DocumentPath="@DocumentPath" @ref="viewer" Width="1060px"
Height="500px">
</SfPdfViewerServer>
@code{
SfPdfViewerServer viewer;
public void OnClick(MouseEventArgs args)
{
viewer.SetAnnotationMode(AnnotationType.Underline);
}
private string DocumentPath { get; set; } =
"wwwroot/Data/PDF_Succinctly.pdf";
}
```

Strikethrough a text

There are two ways to strikethrough a text in the PDF document:

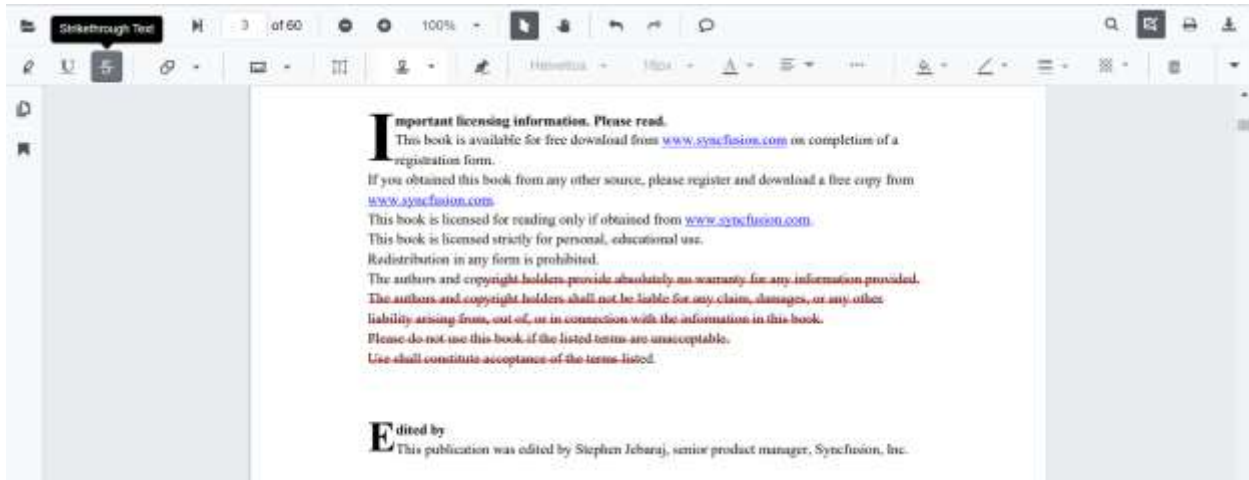
Using the context menu.

- Select a text in the PDF document and right-click it
- Select **strikethrough** option in the context menu that appears.



Using the annotation toolbar.

- Click the **Edit Annotation** button in the PDF Viewer toolbar. A toolbar appears below it.
- Select the **Strikethrough** button in the annotation toolbar. It enables the strikethrough mode.
- Select the text and the strikethrough annotation will be added.
- You can also select the text and apply the strikethrough annotation using the **Strikethrough** button



In the pan mode, if the strikethrough mode is entered, the PDF Viewer control will switch to text select mode to enable the text selection to strike through the text.

Refer to the following code snippet to switch to strikethrough mode

ASPX-CS

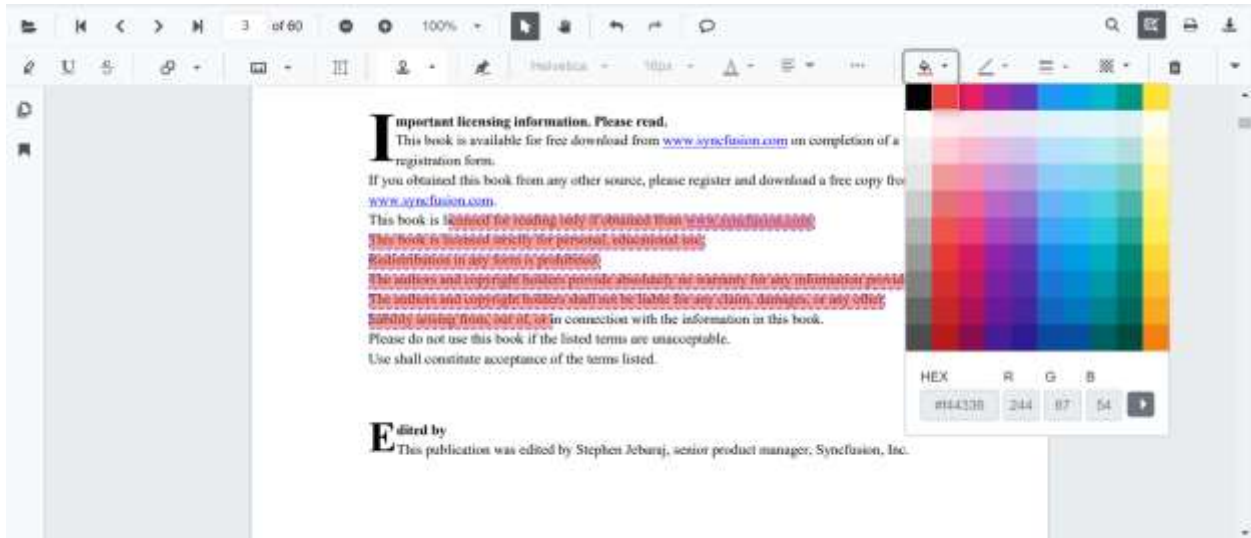
```
@using Syncfusion.Blazor.PdfViewerServer
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.PdfViewer
<SfButton OnClick="OnClick">Click Here</SfButton>
<SfPdfViewerServer DocumentPath="@DocumentPath" @ref="viewer" Width="1060px"
Height="500px">
</SfPdfViewerServer>
@code{
SfPdfViewerServer viewer;
public void OnClick(MouseEventArgs args)
{
viewer.SetAnnotationMode(AnnotationType.Strikethrough);
}
private string DocumentPath { get; set; } =
"wwwroot/Data/PDF_Succinctly.pdf";
}
```

Editing the properties of the text markup annotation

The color and the opacity of the text markup annotation can be edited using the Edit Color tool and the Edit Opacity tool in the annotation toolbar.

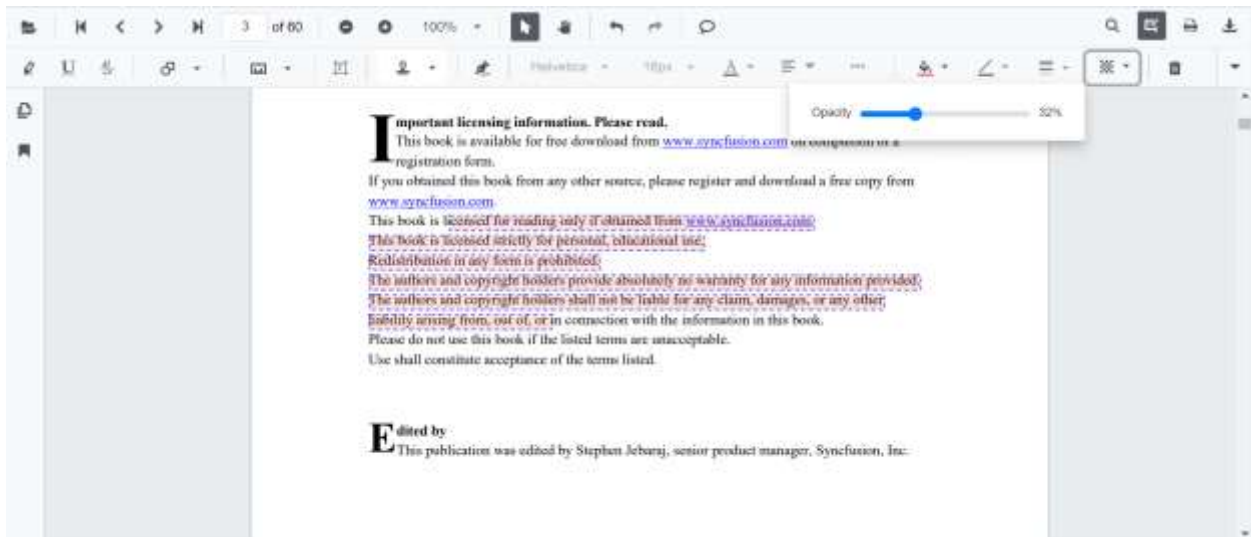
Editing color

The color of the annotation can be edited using the color palette provided in the Edit Color tool.



Editing opacity

The opacity of the annotation can be edited using the range slider provided in the Edit Opacity tool.



Text markup annotation settings

The properties of the text markup annotation can be set before creating the control using `highlightSettings`, `underlineSettings`, and `strikethroughSettings`.

After editing the default color and opacity using the Edit color tool and Edit opacity tool, they will be changed to the selected values.

Refer to the following code snippet to set the default annotation settings.

ASPX-CS

```
@using Syncfusion.Blazor
@using Syncfusion.Blazor.PdfViewerServer
<SfPdfViewerServer @ref="@viewer" DocumentPath="@DocumentPath"
HighlightSettings="@HighlightSettings"
UnderlineSettings="@UnderlineSettings"
StrikethroughSettings="@StrikethroughSettings">
</SfPdfViewerServer>
```

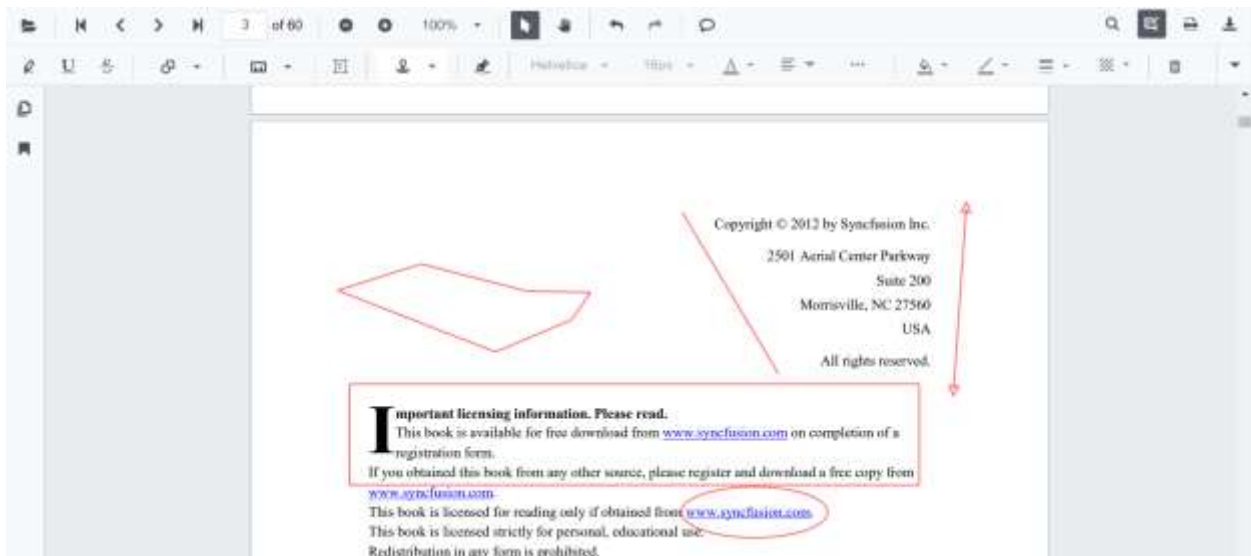
```
@code{
SfPdfViewerServer viewer;
private string DocumentPath { get; set; } =
"wwwroot/Data/PDF_Succinctly.pdf";
PdfViewerHighlightSettings HighlightSettings = new
PdfViewerHighlightSettings { Color = "green", Opacity = 0.6 };
PdfViewerUnderlineSettings UnderlineSettings = new
PdfViewerUnderlineSettings { Color = "blue", Opacity = 0.6 };
PdfViewerStrikethroughSettings StrikethroughSettings = new
PdfViewerStrikethroughSettings { Color = "orange", Opacity = 0.6 };
}
}
```

You can refer to our [Blazor PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor PDF Viewer example](#) to understand how to explain core features of PDF Viewer.

Shape annotations in Blazor PDF Viewer Component

The PDF Viewer control provides the options to add, edit and delete the shape annotations. The shape annotation types supported in the PDF Viewer control are:

- Line
- Arrow
- Rectangle
- Circle
- Polygon



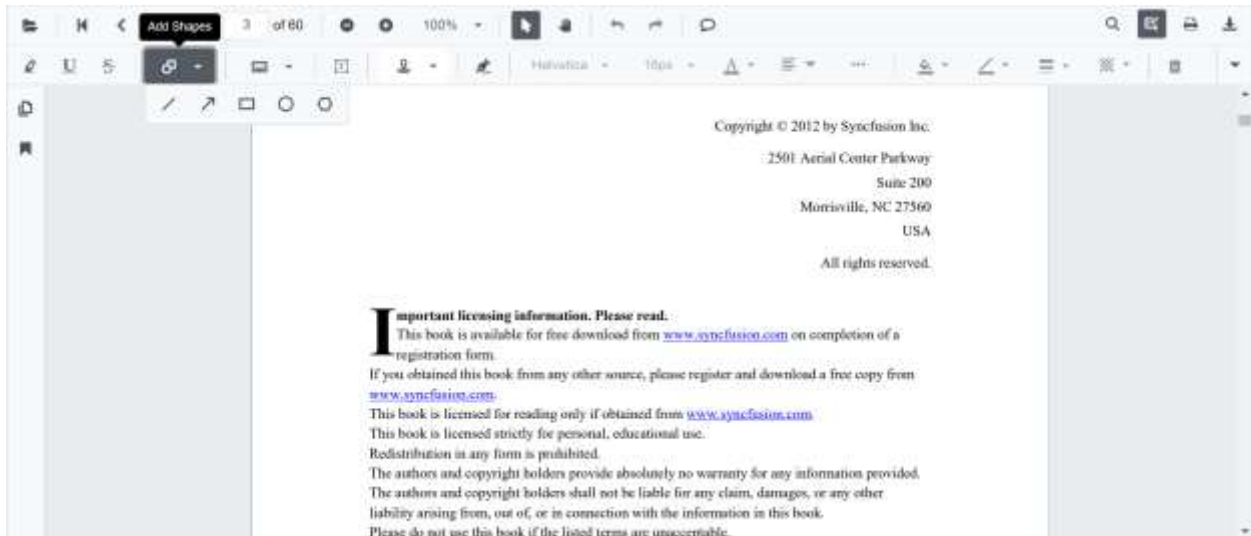
Adding a shape annotation to the PDF document

Shape annotations can be added to the PDF document using the annotation toolbar.

- Click the **Edit Annotation** button in the PDF Viewer toolbar. A toolbar appears below it.
- Click the **Shape Annotation** dropdown button. A dropdown pop-up will appear and shows the shape annotations to be added.

- Select the shape types to be added to the page in the dropdown pop-up. It enables the selected shape annotation mode.
- You can add the shapes over the pages of the PDF document.

In the pan mode, if the shape annotation mode is entered, the PDF Viewer control will switch to text select mode.



Refer to the following code snippet to switch to circle annotation mode.

ASPX-CS

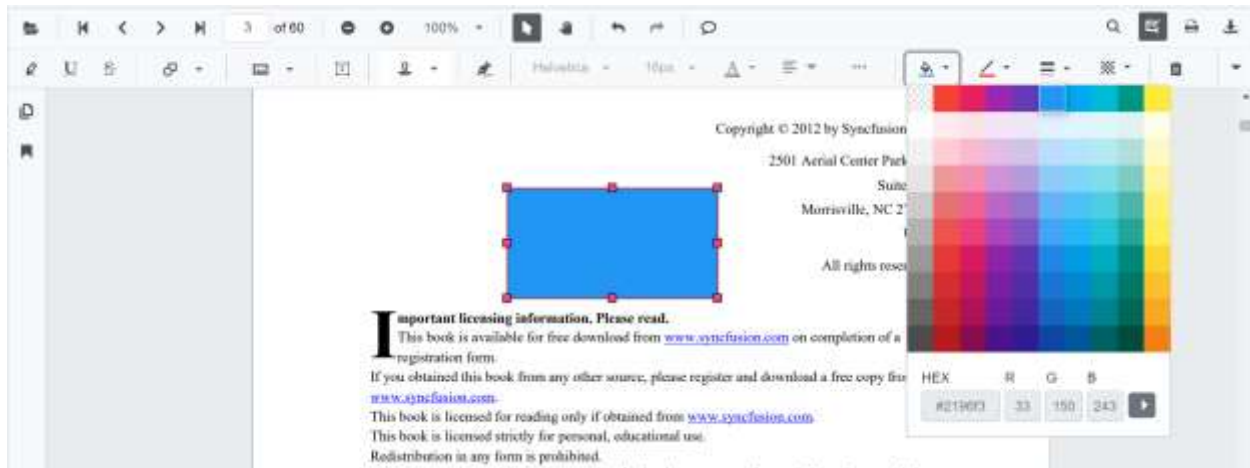
```
@using Syncfusion.Blazor.PdfViewerServer
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.PdfViewer
<SfButton OnClick="OnClick">Click Here</SfButton>
<SfPdfViewerServer DocumentPath="@DocumentPath" @ref="viewer" Width="1060px"
Height="500px">
</SfPdfViewerServer>
@code{
SfPdfViewerServer viewer;
public void OnClick(MouseEventArgs args)
{
viewer.SetAnnotationMode(AnnotationType.Circle);
}
private string DocumentPath { get; set; } =
"wwwroot/Data/PDF_Succinctly.pdf";
}
```

Editing the properties of the shape annotation

The fill color, stroke color, thickness and opacity of the shape annotation can be edited using the Edit color tool, Edit stroke color tool, Edit thickness tool and Edit opacity tool in the annotation toolbar.

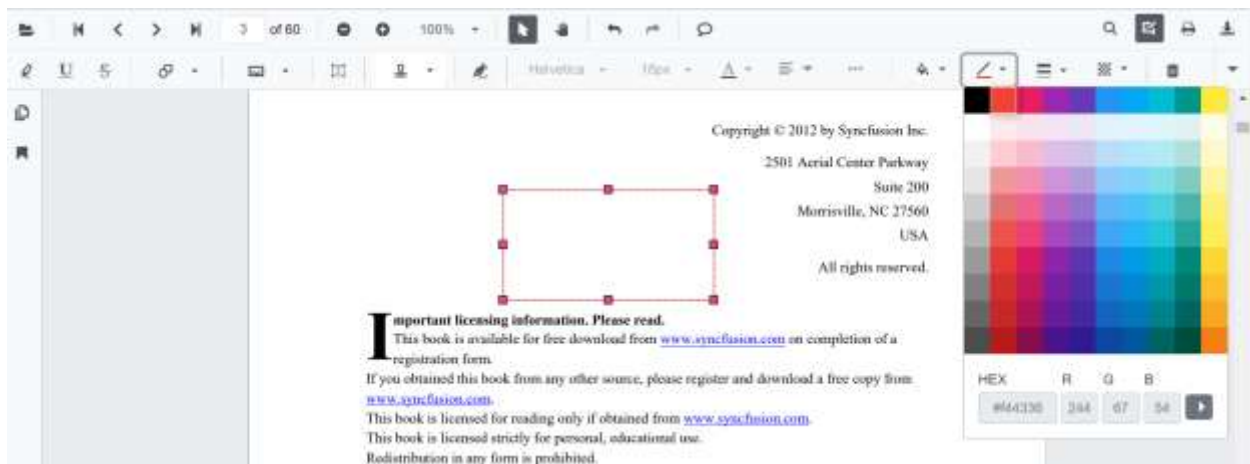
Editing fill color

The fill color of the annotation can be edited using the color palette provided in the Edit Color tool.



Editing stroke color

The stroke color of the annotation can be edited using the color palette provided in the Edit Stroke Color tool.



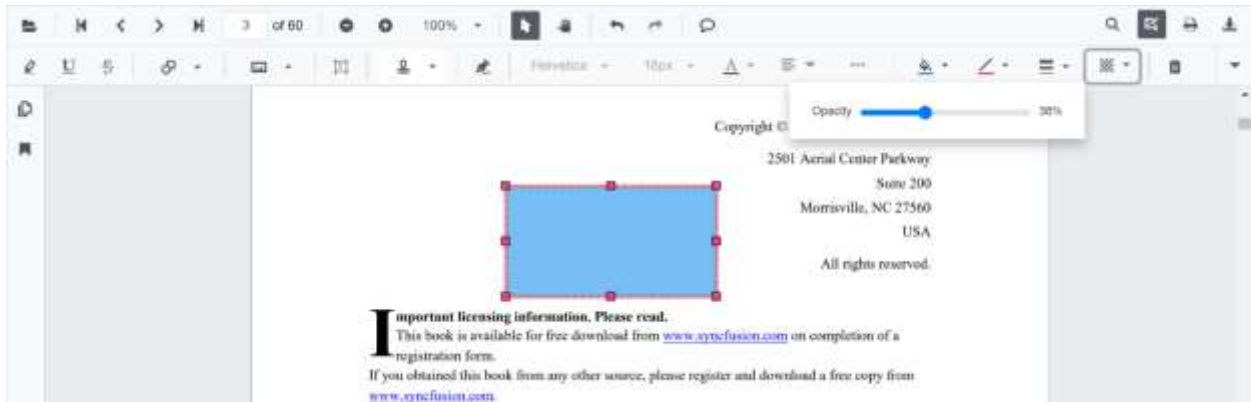
Editing thickness

The thickness of the border of the annotation can be edited using the range slider provided in the Edit Thickness tool.



Editing opacity

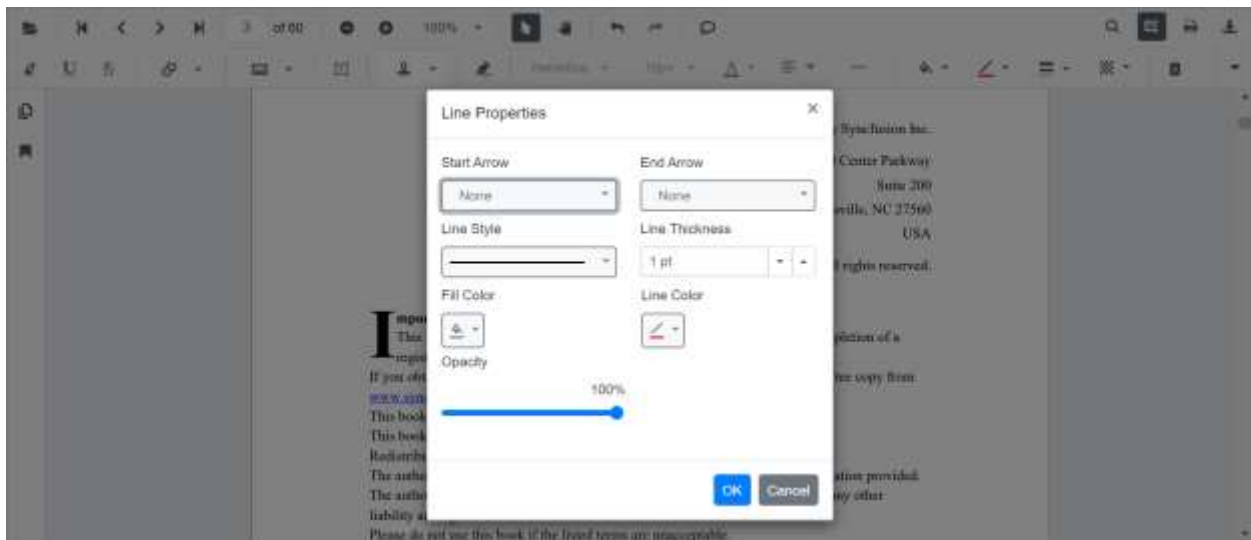
The opacity of the annotation can be edited using the range slider provided in the Edit Opacity tool.



Editing the line properties

The properties of the line shapes such as line and arrow annotations can be edited using the Line Properties window. It can be opened by selecting the Properties option in the context menu that appears on right clicking the line and arrow annotations.

Refer to the following code snippet to set the default annotation settings.



Setting default properties during control initialization

The properties of the shape annotations can be set before creating the control using LineSettings, ArrowSettings, RectangleSettings, CircleSettings, and PolygonSettings.

ASPX-CS

```
@using Syncfusion.Blazor
@using Syncfusion.Blazor.PdfViewerServer
<SfPdfViewerServer @ref="@viewer" DocumentPath="@DocumentPath"
LineSettings="@LineSettings" ArrowSettings="@ArrowSettings"
RectangleSettings="@RectangleSettings" CircleSettings="@CircleSettings"
PolygonSettings="@PolygonSettings">
</SfPdfViewerServer>
@code{
SfPdfViewerServer viewer;
private string DocumentPath { get; set; } =
"wwwroot/Data/PDF_Succinctly.pdf";
```

```

PdfViewerLineSettings LineSettings = new PdfViewerLineSettings {
FillColor="blue", Opacity = 0.6 ,StrokeColor="green"};
PdfViewerArrowSettings ArrowSettings = new PdfViewerArrowSettings {
FillColor="green", Opacity = 0.6 ,StrokeColor="blue"};
PdfViewerRectangleSettings RectangleSettings = new
PdfViewerRectangleSettings { FillColor="yellow", Opacity = 0.6
,StrokeColor="orange" };
PdfViewerCircleSettings CircleSettings = new PdfViewerCircleSettings {
FillColor = "orange", Opacity = 0.6, StrokeColor = "pink" };
PdfViewerPolygonSettings PolygonSettings = new PdfViewerPolygonSettings
{FillColor="pink", Opacity = 0.6 ,StrokeColor="yellow" };
}

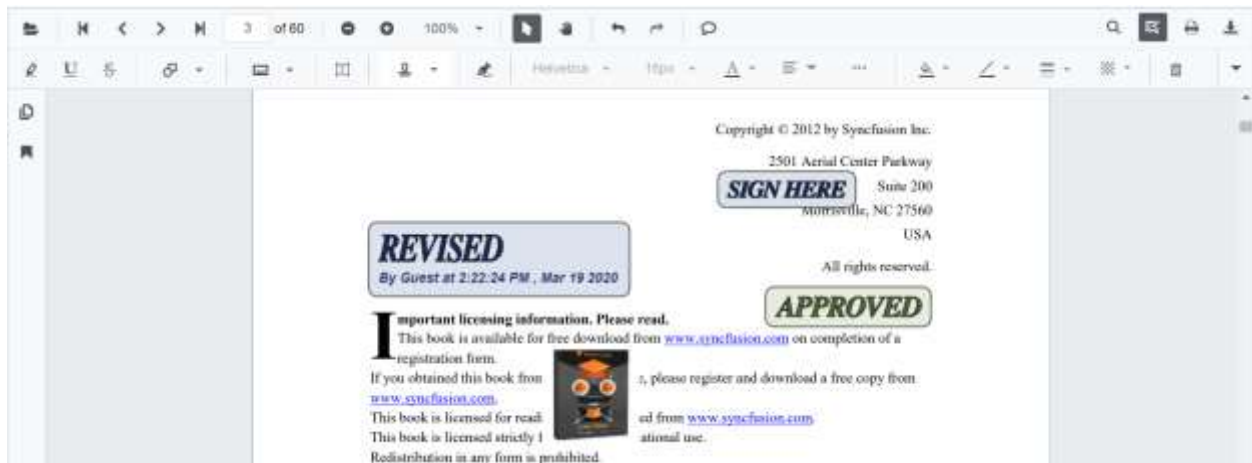
```

You can refer to our [Blazor PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor PDF Viewer example](#) to understand how to explain core features of PDF Viewer.

Stamp annotations in Blazor PDF Viewer Component

The PDF Viewer control provides options to add, edit, delete and rotate the following stamp annotation in the PDF documents:

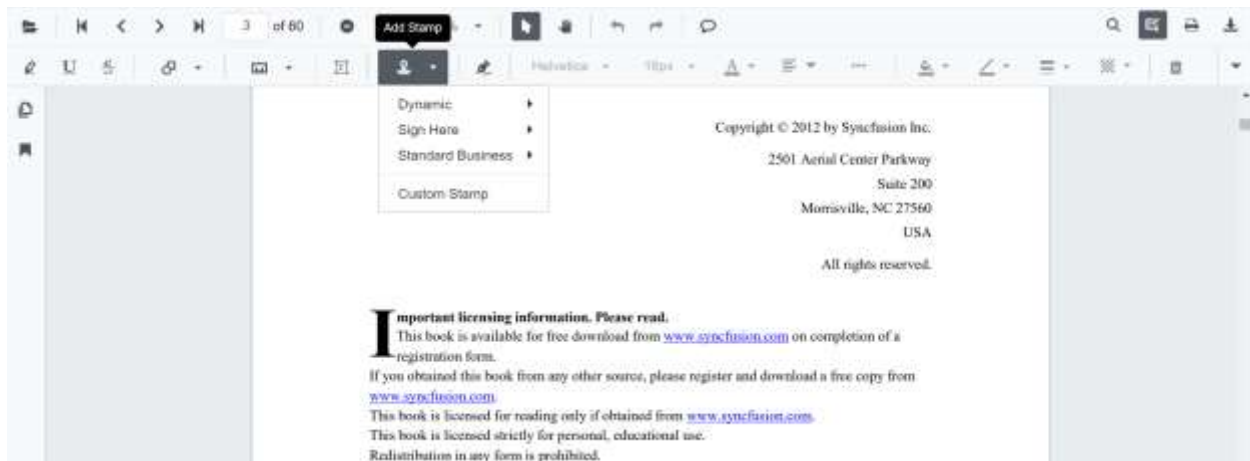
- Dynamic
- Sign Here
- Standard Business
- Custom Stamp



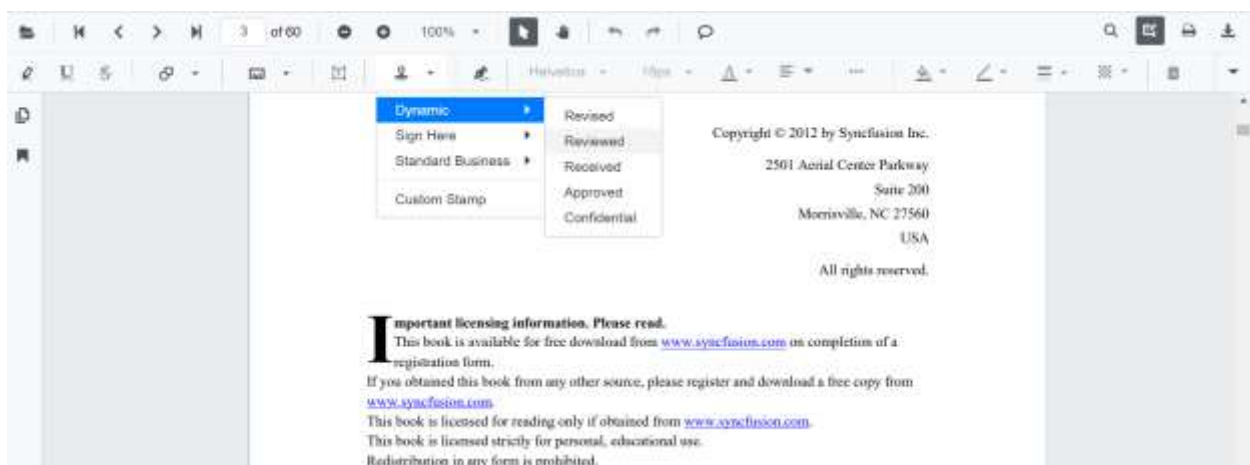
Adding stamp annotations to the PDF document

The stamp annotations can be added to the PDF document using the annotation toolbar.

- Click the **Edit Annotation** button in the PDF Viewer toolbar. A toolbar appears below it.
- Click the **Stamp Annotation** dropdown button. A dropdown pop-up will appear and shows the stamp annotations to be added.



- Select the annotation type to be added to the page in the pop-up.

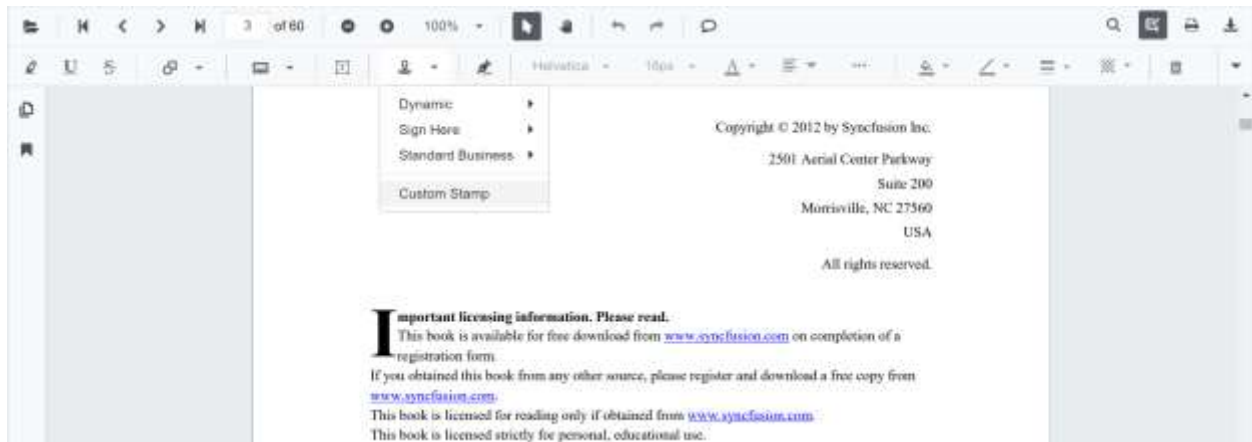


- You can add the annotation over the pages of the PDF document.

In the pan mode, if the stamp annotation mode is entered, the PDF Viewer control will switch to text select mode.

Adding custom stamp to the PDF document

- Click the **Edit Annotation** button in the PDF Viewer toolbar. A toolbar appears below it.
- Click the **Stamp Annotation** dropdown button. A dropdown pop-up will appear and shows the stamp annotations to be added.
- Click the Custom Stamp button.



- The file explorer dialog will appear, choose the image and then add the image in the PDF page.

The JPG and JPEG image format are only supported in custom stamp annotations.

Setting default properties during control initialization

The properties of the stamp annotation can be set before creating the control using StampSettings.

After editing the default opacity using the Edit Opacity tool, they will be changed to the selected values.

Refer to the following code snippet to set the default sticky note annotation settings.

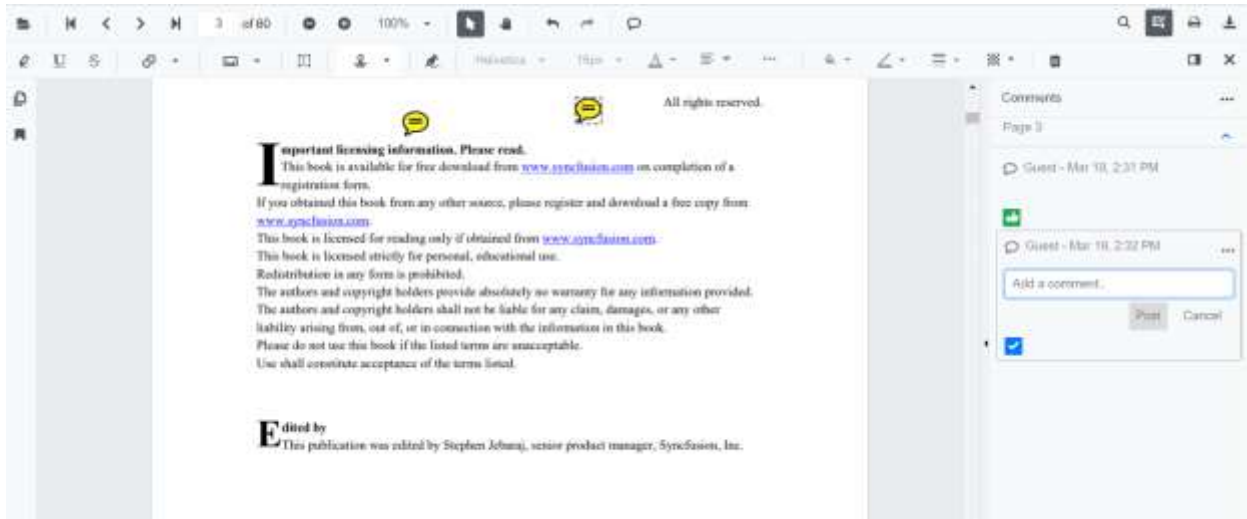
ASPX-CS

```
@using Syncfusion.Blazor
@using Syncfusion.Blazor.PdfViewerServer
<SfPdfViewerServer @ref="@viewer" DocumentPath="@DocumentPath"
StampSettings="@StampSettings" >
</SfPdfViewerServer>
@code{
SfPdfViewerServer viewer;
private string DocumentPath { get; set; } =
"wwwroot/Data/PDF_Succinctly.pdf";
PdfViewerStampSettings StampSettings = new PdfViewerStampSettings
{ Opacity=0.3, Author="Blazor" };
}
```

You can refer to our [Blazor PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor PDF Viewer example](#) to understand how to explain core features of PDF Viewer.

Sticky notes annotations in Blazor PDF Viewer Component

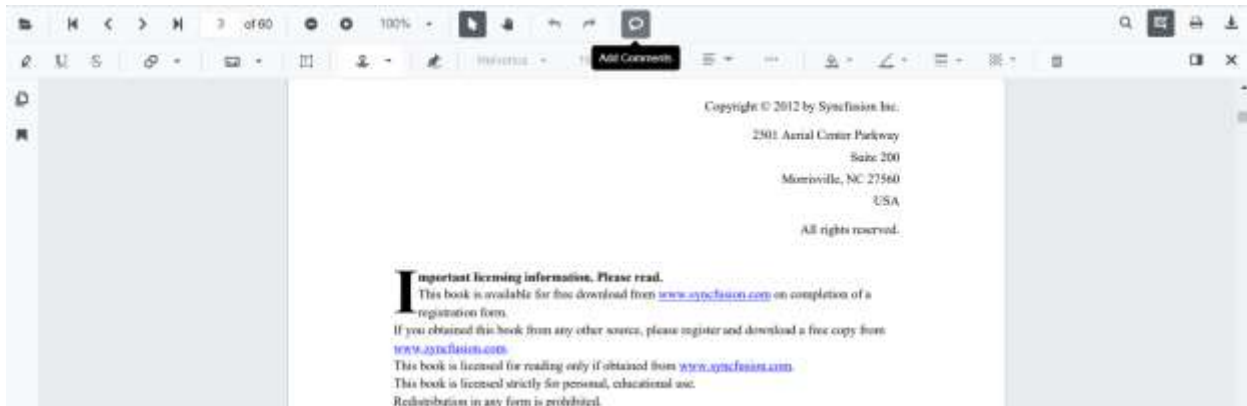
The PDF Viewer control provides the options to add, edit, and delete the sticky note annotations in the PDF document.



Adding a sticky note annotation to the PDF document

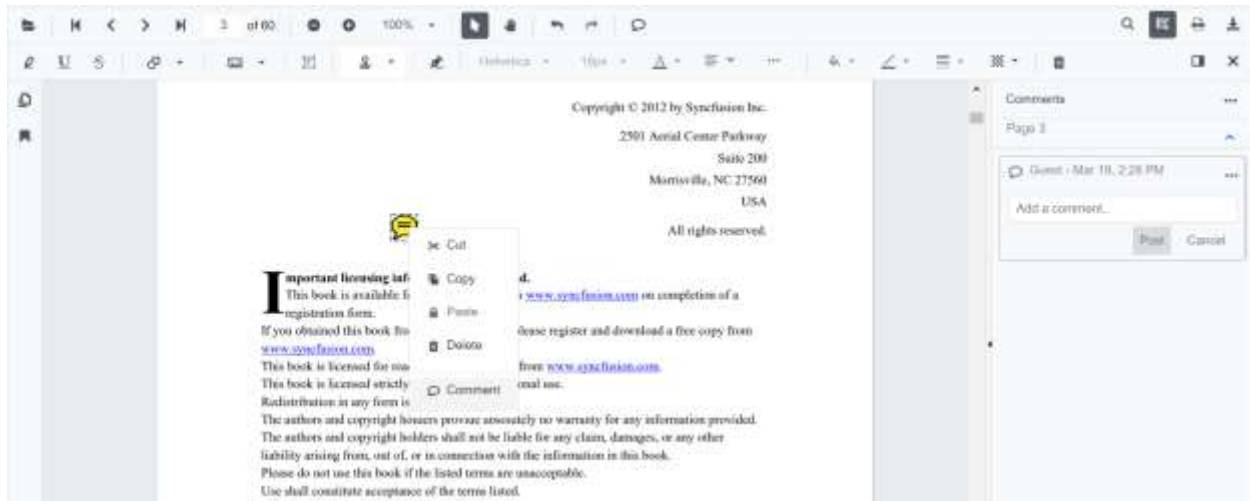
Sticky note annotations can be added to the PDF document using the annotation toolbar.

- Click the **Comments** button in the PDF Viewer toolbar. A toolbar appears below it.
- Click the position, where you want to add sticky note annotation in the PDF document.
- Sticky note annotation will be added in the clicked positions.



Annotation comments can be added to the PDF document using the comment panel.

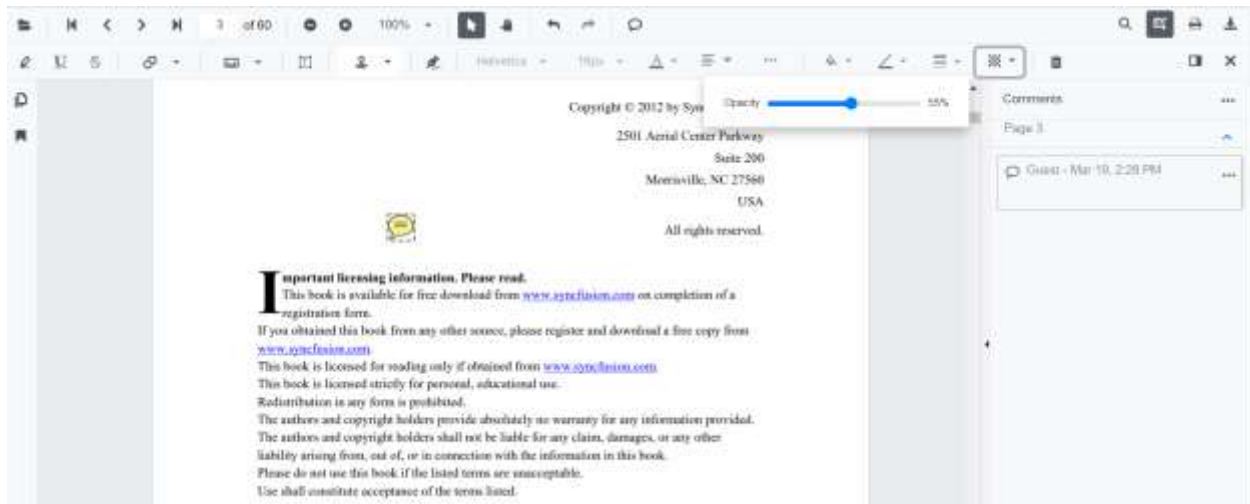
- Select a Sticky note annotation in the PDF document and right-click it.
- Select Comment option in the context menu that appears.
- Now, you can add Comments, Reply, and Status using Comment Panel.
- Now, you can add Comments, Reply, and Status using Comment Panel



Editing the properties of the sticky note annotation

Editing opacity

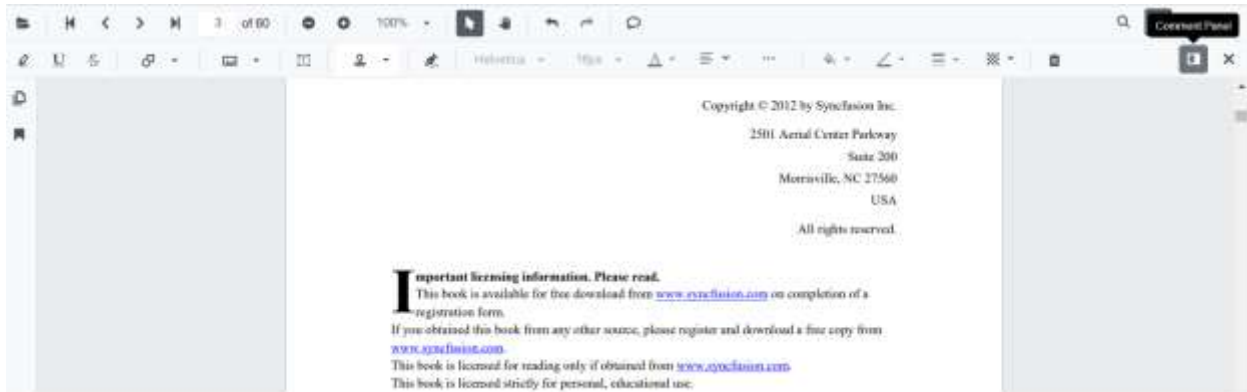
The opacity of the annotation can be edited using the range slider provided in the Edit Opacity tool.



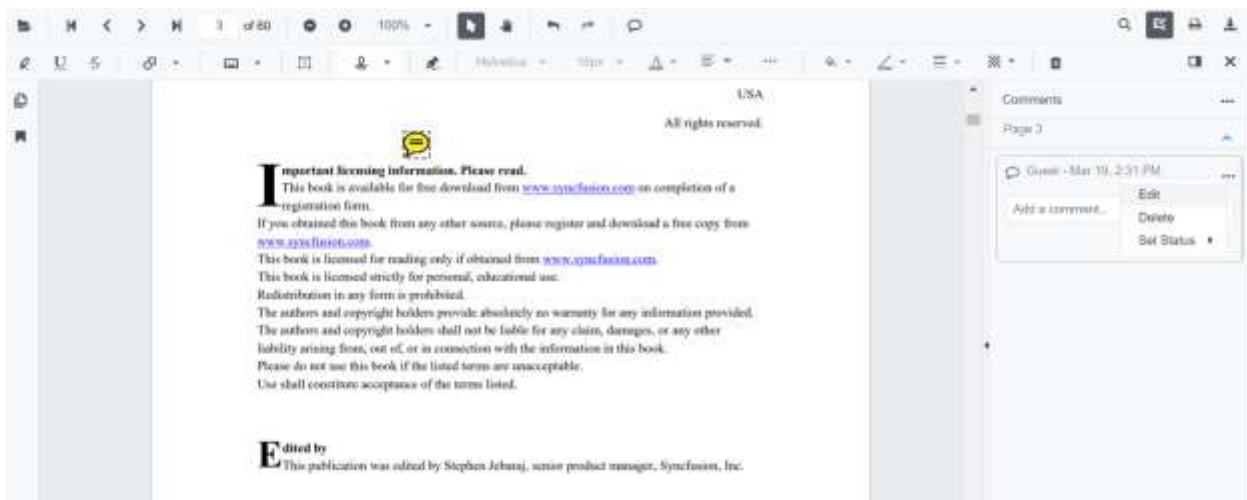
Editing comments

The comment, comment reply, and comment status of the annotation can be edited using the Comment Panel.

- Open the comment panel using the Comment Panel button showing in the annotation toolbar.



You can modify or delete the comments or comments replay and it's status using the menu option provided in the comment panel.



Setting default properties during control initialization

The properties of the sticky note annotation can be set before creating the control using StickyNoteSettings.

After editing the default opacity using the Edit Opacity tool, they will be changed to the selected values. Refer to the following code snippet to set the default sticky note annotation settings.

ASPX-CS

```
@using Syncfusion.Blazor
@using Syncfusion.Blazor.PdfViewerServer
<SfPdfViewerServer @ref="@viewer" DocumentPath="@DocumentPath"
StickyNotesSettings="@StickyNotesSettings" >
</SfPdfViewerServer>
@code{
SfPdfViewerServer viewer;
private string DocumentPath { get; set; } =
"wwwroot/Data/PDF_Succinctly.pdf";
PdfViewerStickyNotesSettings StickyNotesSettings = new
PdfViewerStickyNotesSettings {Author="Syncfusion"};
}
```

Disabling sticky note annotations

The PDF Viewer control provides an option to disable the sticky note annotations feature. The code snippet for disabling the feature is as follows.

ASPX-CS

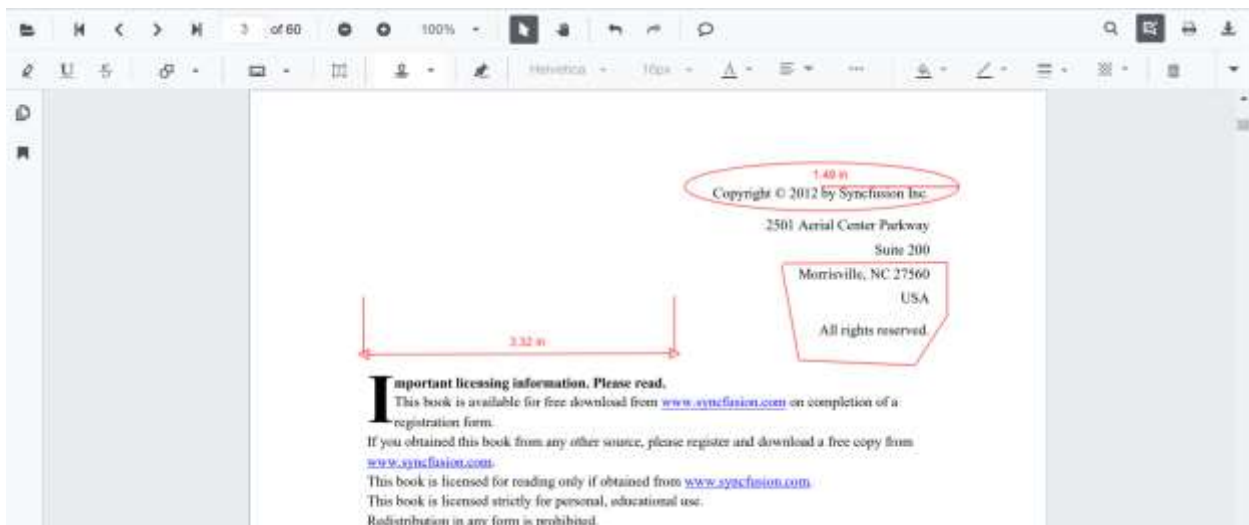
```
@using Syncfusion.Blazor.PdfViewerServer
@using Syncfusion.Blazor.PdfViewer
<SfPdfViewerServer DocumentPath="@DocumentPath"
EnableStickyNotesAnnotation=false>
</SfPdfViewerServer>
@code{
private string DocumentPath { get; set; } =
"wwwroot/Data/PDF_Succinctly.pdf";
}
```

You can refer to our [Blazor PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor PDF Viewer example](#) to understand how to explain core features of PDF Viewer.

Measurement annotations in Blazor PDF Viewer Component

The PDF Viewer provides the options to add measurement annotations. You can measure the page annotations with the help of measurement annotation. The supported measurement annotations in the PDF Viewer control are:

- Distance
- Perimeter
- Area
- Radius
- Volume



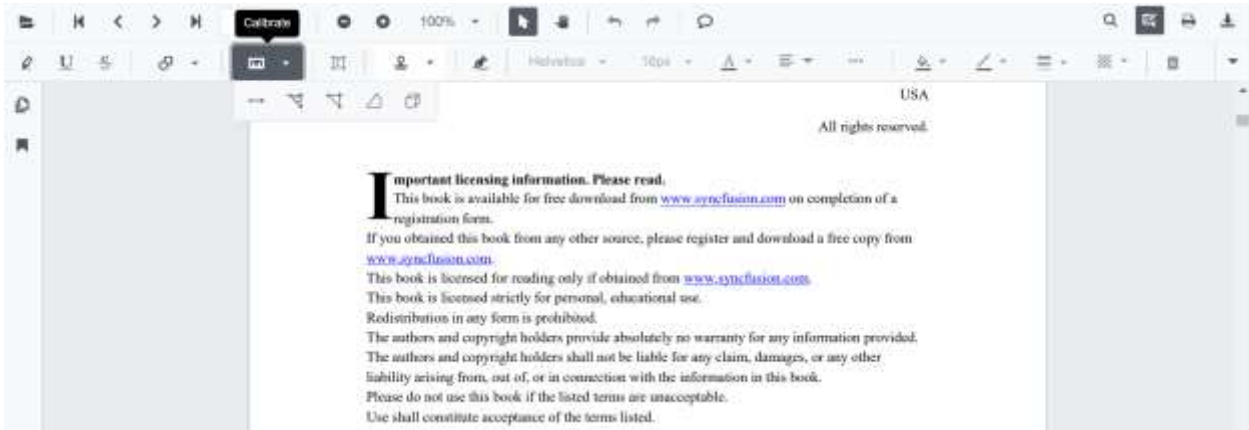
Adding measurement annotations to the PDF document

The measurement annotations can be added to the PDF document using the annotation toolbar.

- Click the **Edit Annotation** button in the PDF Viewer toolbar. A toolbar appears below it.

- Click the **measurement Annotation** dropdown button. A dropdown pop-up will appear and shows the measurement annotations to be added.
- Select the measurement type to be added to the page in the dropdown pop-up. It enables the selected measurement annotation mode.
- You can measure and add the annotation over the pages of the PDF document.

In the pan mode, if the measurement annotation mode is entered, the PDF Viewer control will switch to text select mode.



Refer to the following code snippet to switch to distance annotation mode.

ASPX-CS

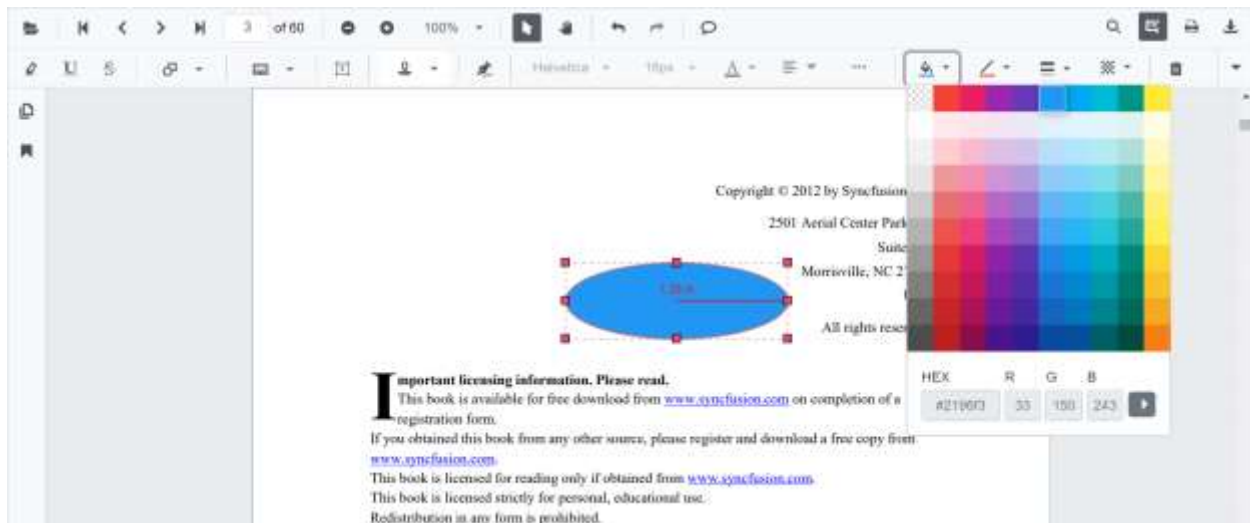
```
@using Syncfusion.Blazor.PdfViewerServer
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.PdfViewer
<SfButton OnClick="OnClick">Click Here</SfButton>
<SfPdfViewerServer DocumentPath="@DocumentPath" @ref="viewer" Width="1060px"
Height="500px">
</SfPdfViewerServer>
@code{
SfPdfViewerServer viewer;
public void OnClick(MouseEventArgs args)
{
viewer.SetAnnotationMode(AnnotationType.Distance);
}
private string DocumentPath { get; set; } =
"wwwroot/Data/PDF_Succinctly.pdf";
}
```

Editing the properties of measurement annotation

The fill color, stroke color, thickness, and opacity of the measurement annotation can be edited using the Edit Color tool, Edit Stroke Color tool, Edit Thickness tool, and Edit Opacity tool in the annotation toolbar.

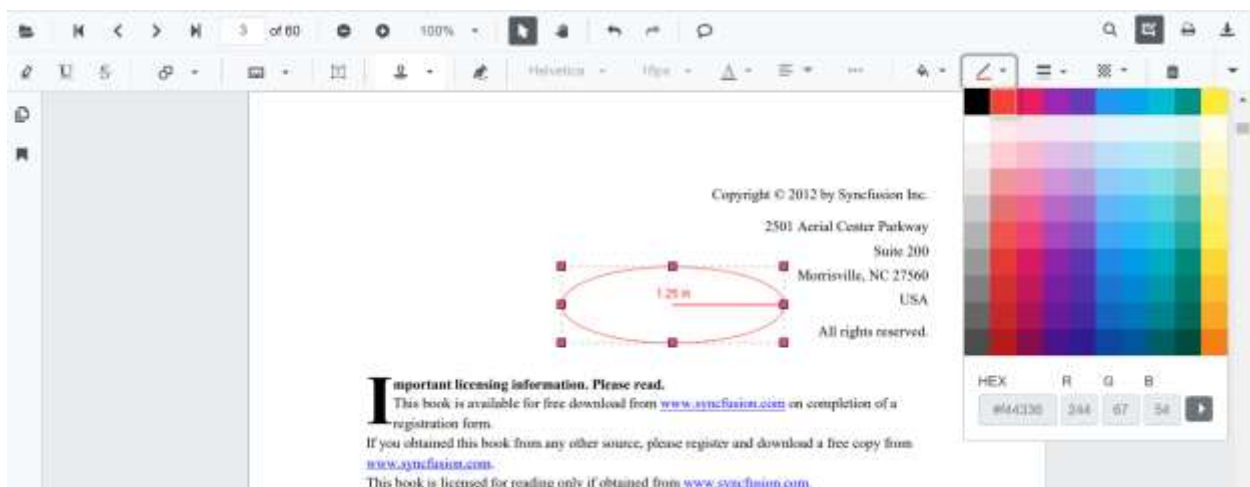
Editing fill color

The fill color of the annotation can be edited using the color palette provided in the Edit Color tool.



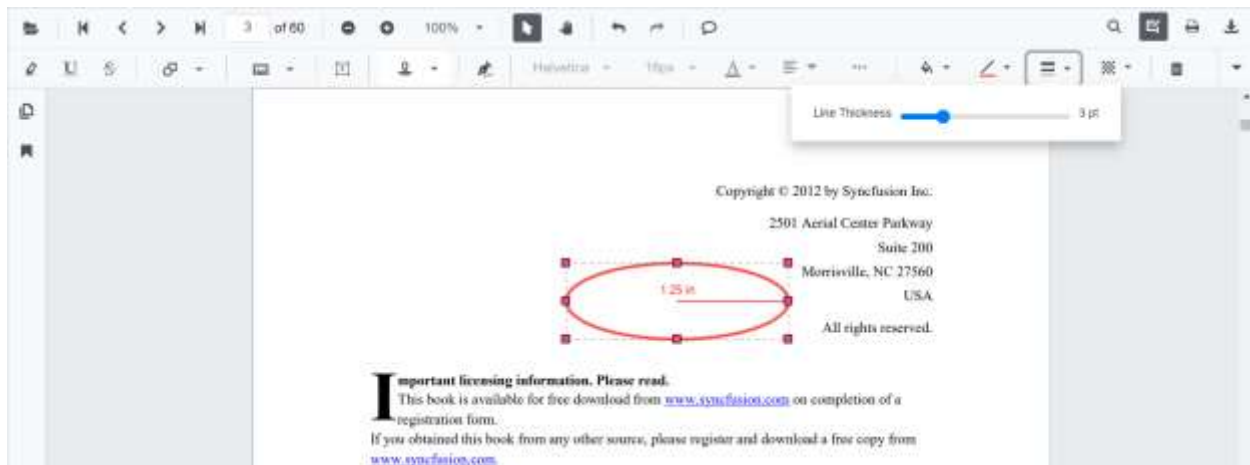
Editing stroke color

The stroke color of the annotation can be edited using the color palette provided in the Edit Stroke Color tool.



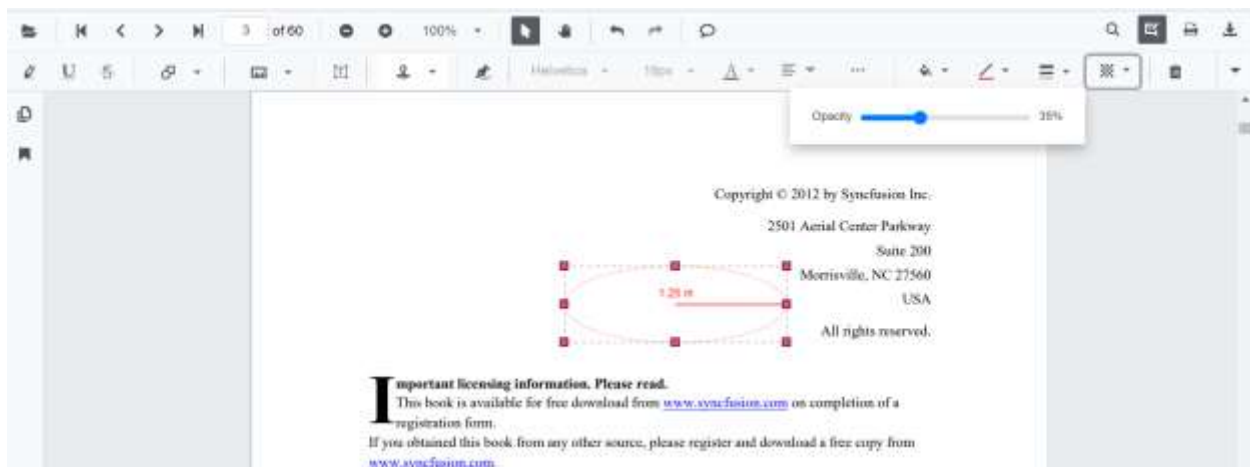
Editing thickness

The thickness of the border of the annotation can be edited using the range slider provided in the Edit thickness tool.



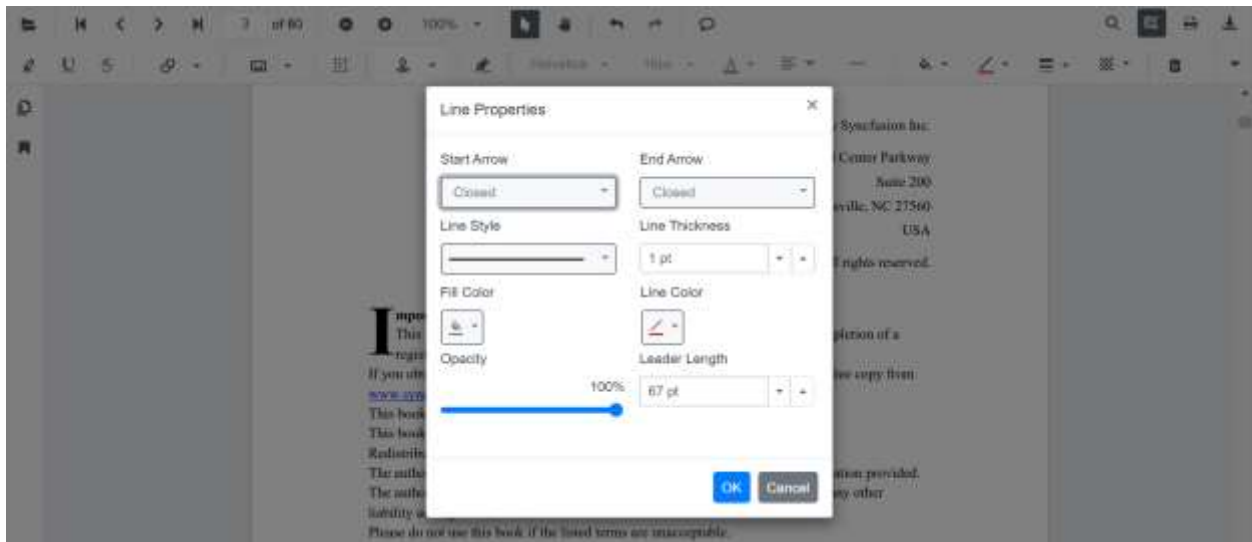
Editing opacity

The opacity of the annotation can be edited using the range slider provided in the Edit Opacity tool.



Editing the line properties

The properties of the line shapes such as distance and perimeter annotations can be edited using the Line properties window. It can be opened by selecting the Properties option in the context menu that appears on right-clicking the distance and perimeter annotations.



Setting default properties during control initialization

The properties of the shape annotations can be set before creating the control using `distanceSettings`, `perimeterSettings`, `areaSettings`, `radiusSettings` and `volumeSettings`.

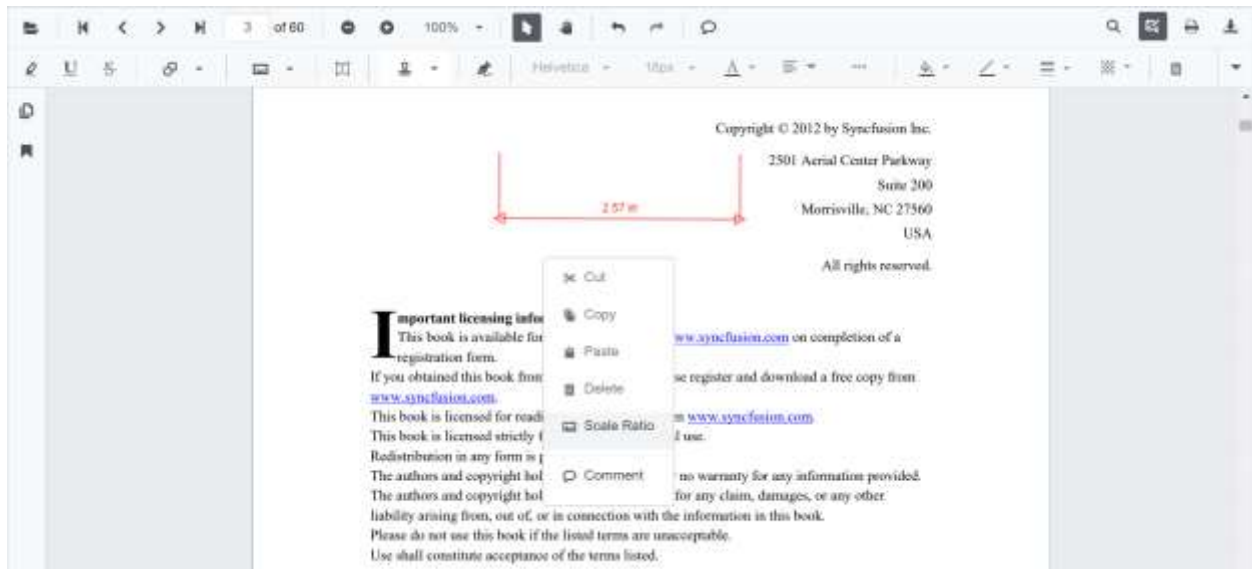
Refer to the following code snippet to set the default annotation settings.

ASPX-CS

```
@using Syncfusion.Blazor
@using Syncfusion.Blazor.PdfViewerServer
<SfPdfViewerServer @ref="@viewer" DocumentPath="@DocumentPath"
DistanceSettings="@DistanceSettings" PerimeterSettings="@PerimeterSettings"
AreaSettings="@AreaSettings" RadiusSettings="@RadiusSettings"
VolumeSettings="@VolumeSettings">
</SfPdfViewerServer>
@code{
SfPdfViewerServer viewer;
private string DocumentPath { get; set; } =
"wwwroot/Data/PDF_Succinctly.pdf";
PdfViewerDistanceSettings DistanceSettings = new PdfViewerDistanceSettings {
FillColor="blue", Opacity = 0.6
,StrokeColor="green",LineHeadEndStyle=LineHeadStyle.Closed,LineHeadStartStyle=LineHeadStyle.Round};
PdfViewerPerimeterSettings PerimeterSettings = new
PdfViewerPerimeterSettings { FillColor="green", Opacity = 0.6
,StrokeColor="blue"};
PdfViewerAreaSettings AreaSettings = new PdfViewerAreaSettings {
FillColor="yellow", Opacity = 0.6 ,StrokeColor="orange" };
PdfViewerVolumeSettings VolumeSettings = new PdfViewerVolumeSettings {
FillColor = "orange", Opacity = 0.6, StrokeColor = "pink" };
PdfViewerRadiusSettings RadiusSettings = new PdfViewerRadiusSettings
{FillColor="pink", Opacity = 0.6 ,StrokeColor="yellow" };
}
```

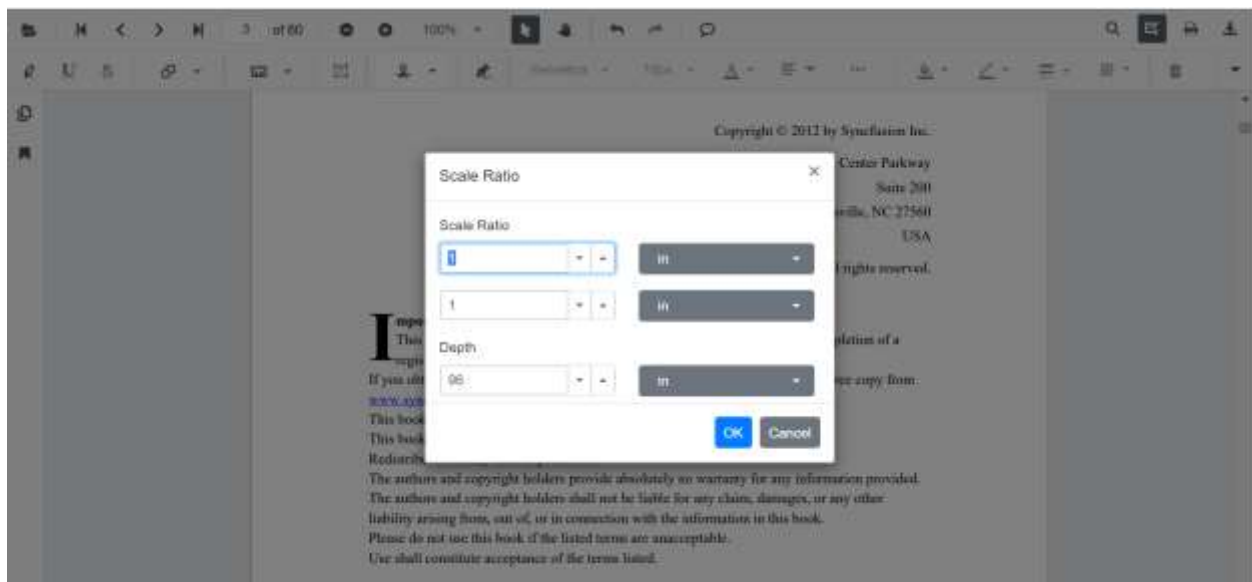
Editing scale ratio and unit of the measurement annotation

The scale ratio and unit of measurement can be modified using the scale ratio option provided in the context menu of the PDF Viewer control.



The Units of measurements support for the measurement annotations in the PDF Viewer are

- Inch
- Millimeter
- Centimeter
- Point
- Pica
- Feet



Setting default scale ratio settings during control initialization

The properties of scale ratio for measurement annotation can be set before creating the control using `ScaleRatioSettings` as shown in the following code snippet,

ASPX-CS

```
@using Syncfusion.Blazor
```

```
@using Syncfusion.Blazor.PdfViewerServer
<SfPdfViewerServer @ref="@viewer" DocumentPath="@DocumentPath"
MeasurementSettings="@MeasurementSettings" >
</SfPdfViewerServer>
@code{
SfPdfViewerServer viewer;
private string DocumentPath { get; set; } =
"wwwroot/Data/PDF_Succinctly.pdf";
PdfViewerMeasurementSettings MeasurementSettings = new
PdfViewerMeasurementSettings {
ScaleRatio=2,ConversionUnit=CalibrationUnit.Cm};
}
```

You can refer to our [Blazor PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor PDF Viewer example](#) to understand how to explain core features of PDF Viewer.

Free text annotations in Blazor PDF Viewer Component

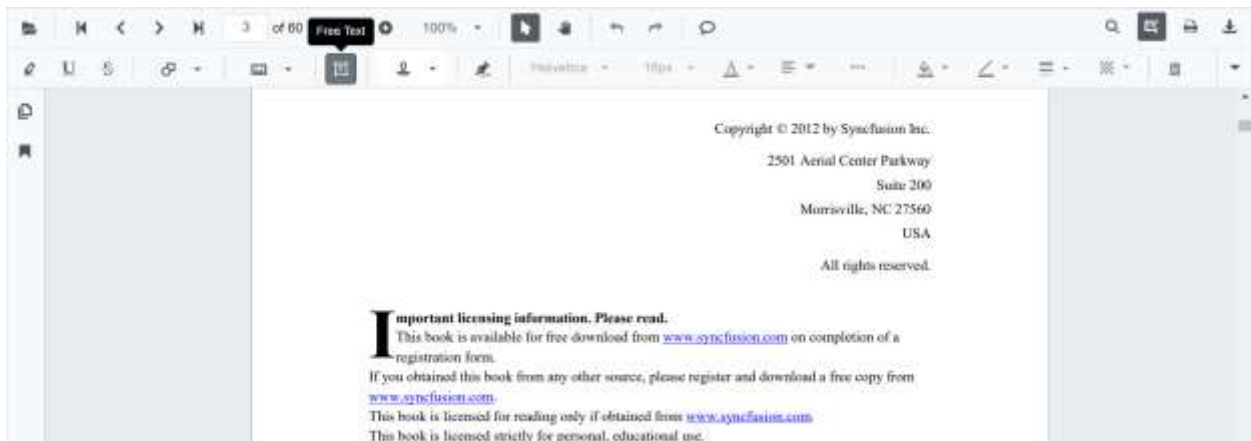
The PDF Viewer control provides the options to add, edit and delete the free text annotations.

Adding a free text annotation to the PDF document

The Free text annotations can be added to the PDF document using the annotation toolbar.

- Click the **Edit Annotation** button in the PDF Viewer toolbar. A toolbar appears below it.
- Select the **Free Text Annotation** button in the annotation toolbar. It enables the Free Text annotation mode.
- You can add the text over the pages of the PDF document.

In the pan mode, if the free text annotation mode is entered, the PDF Viewer control will switch to text select mode.



Refer to the following code snippet to switch to Free Text annotation mode.

ASPX-CS

```
@using Syncfusion.Blazor.PdfViewerServer
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.PdfViewer
<SfButton OnClick="OnClick">Click Here</SfButton>
```

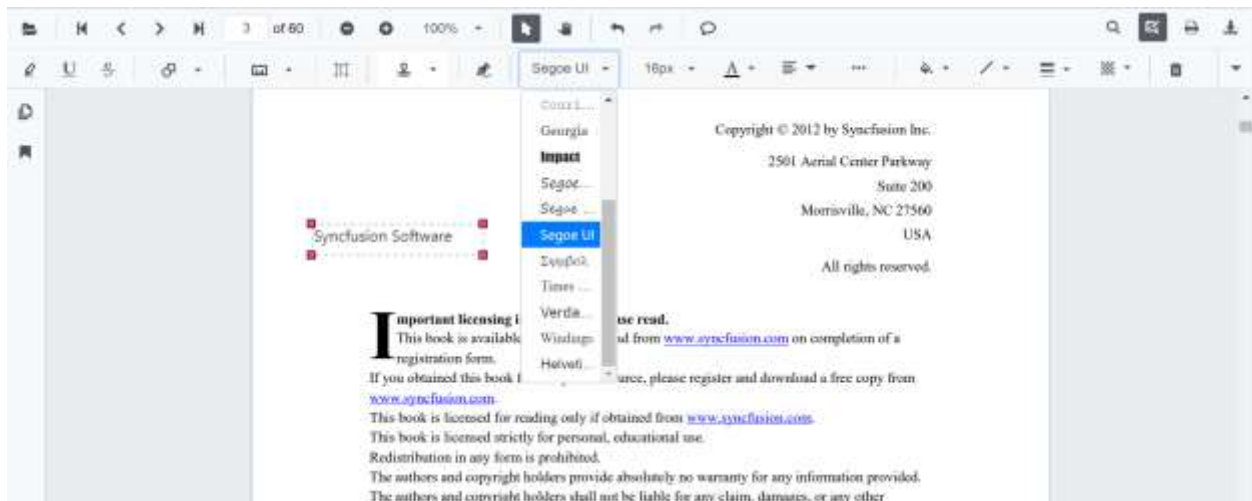
```
<SfPdfViewerServer DocumentPath="@DocumentPath" @ref="viewer" Width="1060px"
Height="500px">
</SfPdfViewerServer>
@code{
SfPdfViewerServer viewer;
public void OnClick(MouseEventArgs args)
{
viewer.SetAnnotationMode(AnnotationType.FreeText);
}
private string DocumentPath { get; set; } =
"wwwroot/Data/PDF_Succinctly.pdf";
}
```

Editing the properties of free text annotation

The font family, font size, font styles, font color, text alignment, fill color, the border stroke color, border thickness, and opacity of the free text annotation can be edited using the Font Family tool, Font Size tool, Font Color tool, Text Align tool, Font Style tool, Edit Color tool, Edit Stroke Color tool, Edit Thickness tool, and Edit Opacity tool in the annotation toolbar.

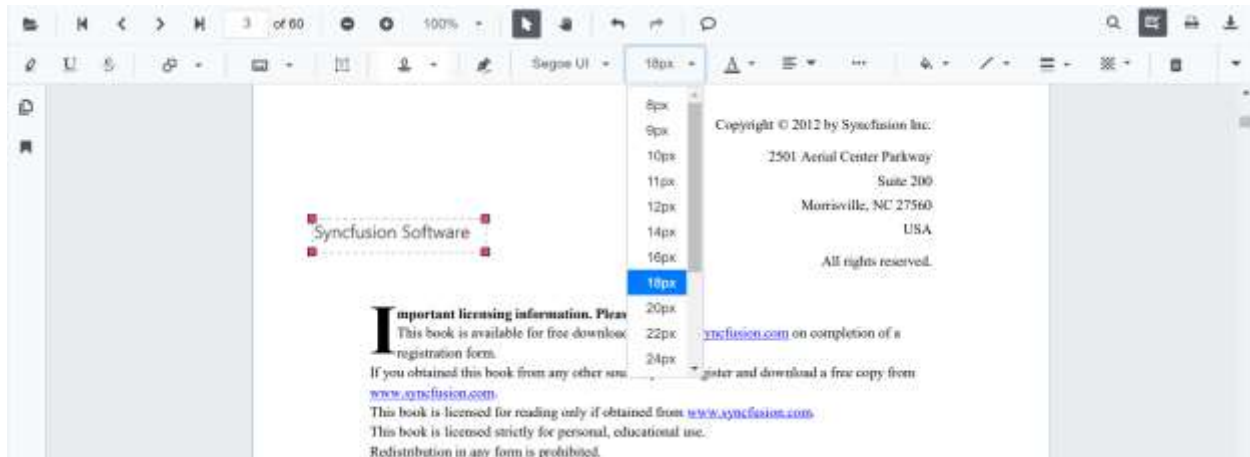
Editing font family

The font family of the annotation can be edited by selecting the desired font in the Font Family tool.



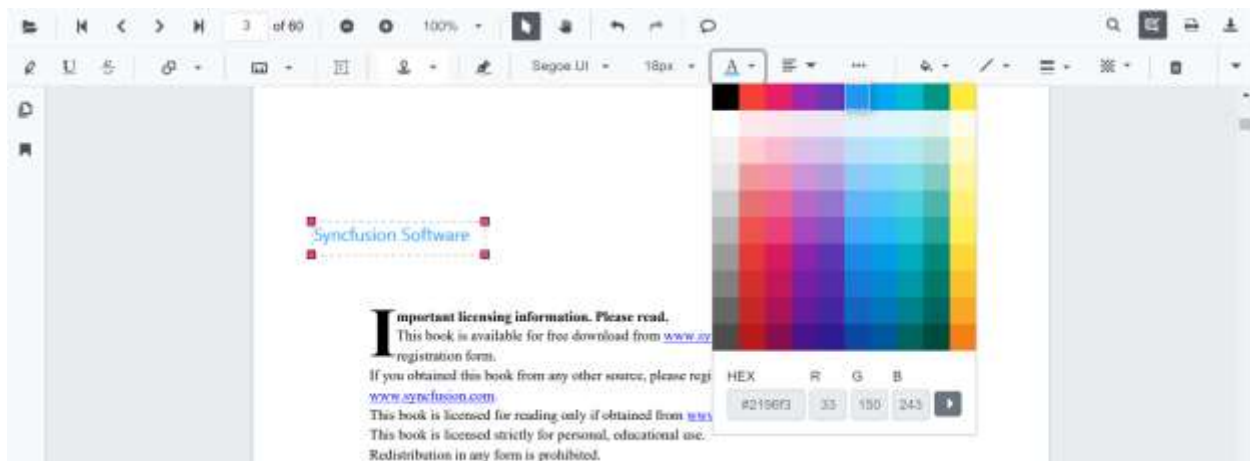
Editing font size

The font size of the annotation can be edited by selecting the desired size in the Font Size tool.



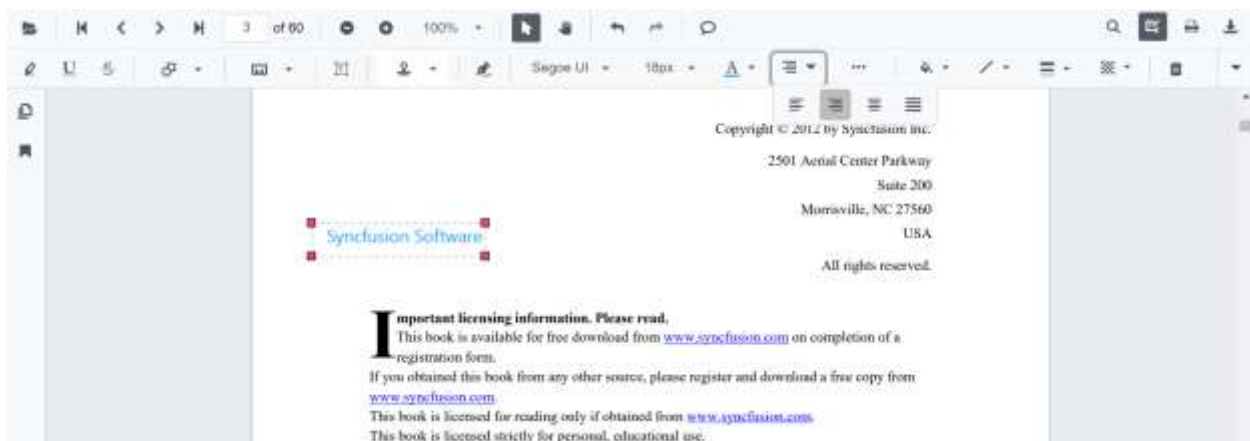
Editing font color

The font color of the annotation can be edited using the color palette provided in the Font Color tool.



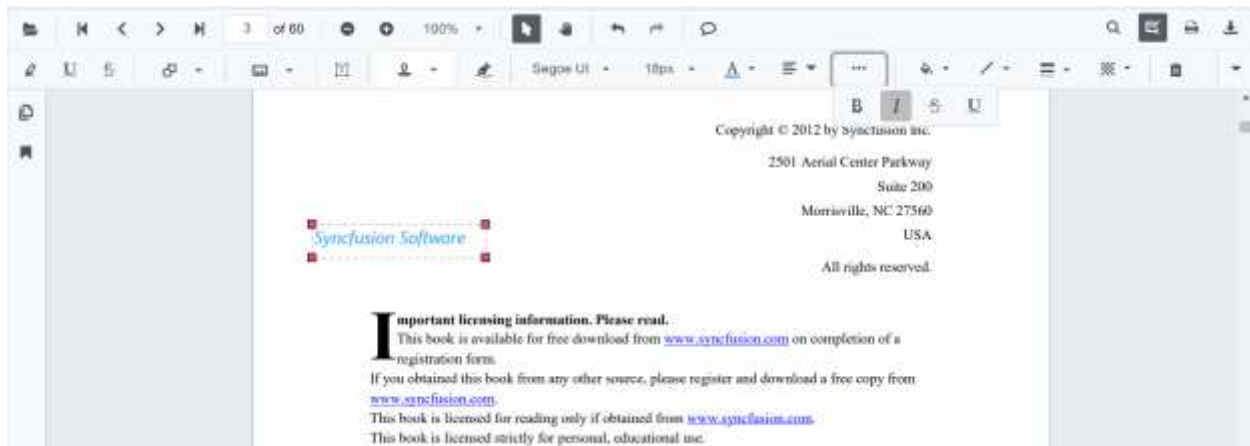
Editing the text alignment

The text in the annotation can be aligned by selecting the desired styles in the dropdown pop-up in the Text Align tool.



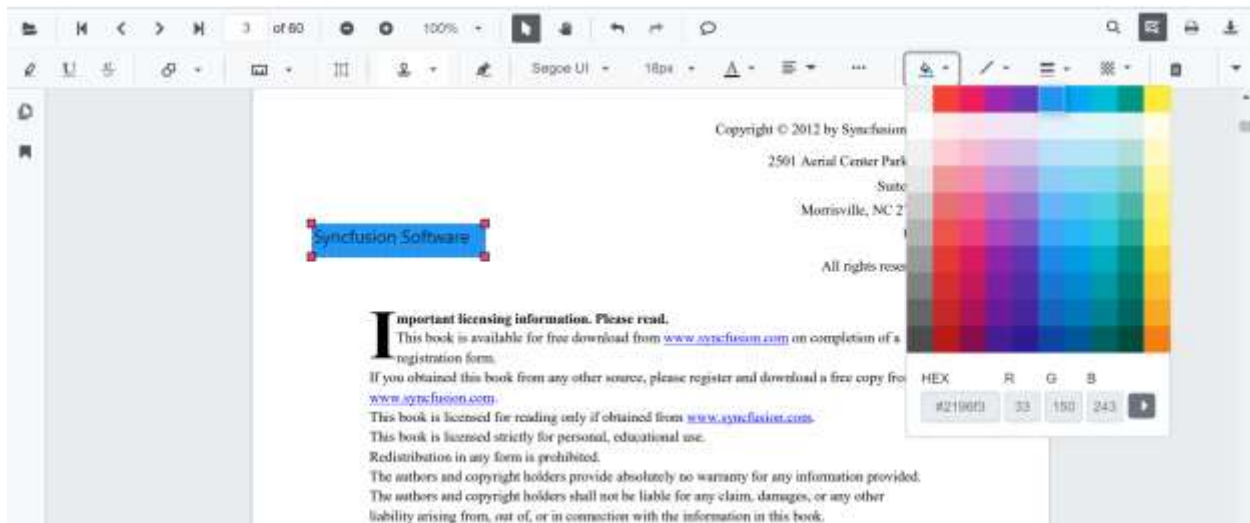
Editing text styles

The style of the text in the annotation can be edited by selecting the desired styles in the dropdown pop-up in the Font Style tool.



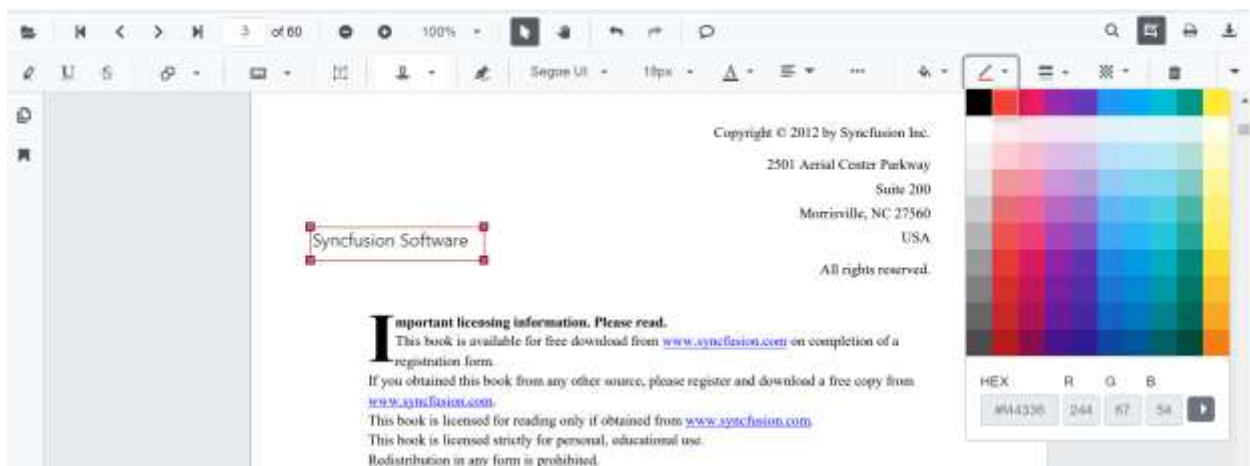
Editing fill color

The fill color of the annotation can be edited using the color palette provided in the Edit Color tool.



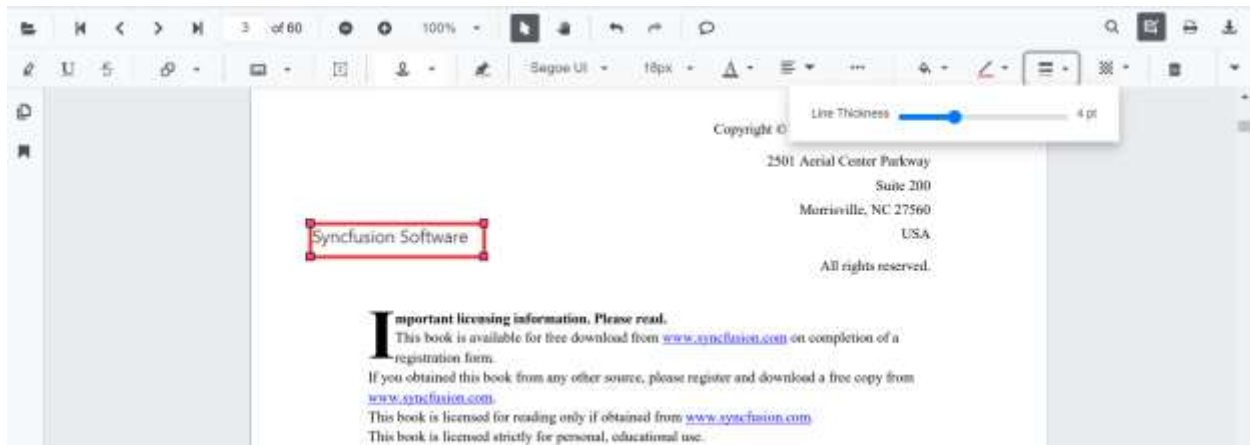
Editing stroke color

The stroke color of the annotation can be edited using the color palette provided in the Edit Stroke Color tool.



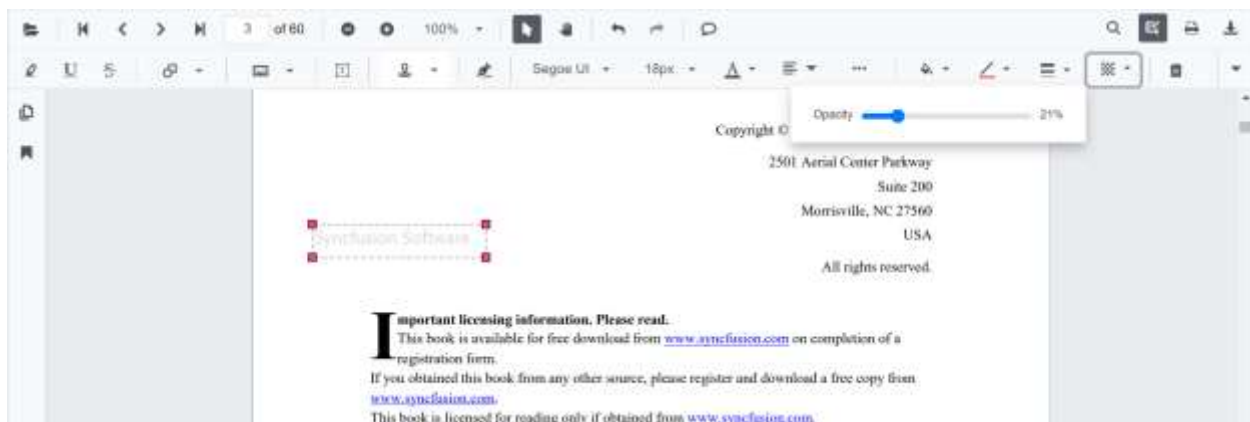
Editing thickness

The thickness of the border of the annotation can be edited using the range slider provided in the Edit Thickness tool.



Editing opacity

The opacity of the annotation can be edited using the range slider provided in the Edit Opacity tool.



Setting default properties during control initialization

The properties of the free text annotation can be set before creating the control using FreeTextSettings.

After editing the default values, they will be changed to the selected values.

Refer to the following code snippet to set the default free text annotation settings

ASPX-CS

```
@using Syncfusion.Blazor.PdfViewerServer
@using Syncfusion.Blazor.PdfViewer
<SfPdfViewerServer @ref="@viewer" DocumentPath="@DocumentPath"
FreeTextSettings="@FreeTextSettings">
</SfPdfViewerServer>
@code{
SfPdfViewerServer viewer;
private string DocumentPath { get; set; } =
"wwwroot/Data/PDF_Succinctly.pdf";
PdfViewerFreeTextSettings FreeTextSettings = new
PdfViewerFreeTextSettings{FillColor="green",BorderColor="blue",FontColor="yellow"};
```

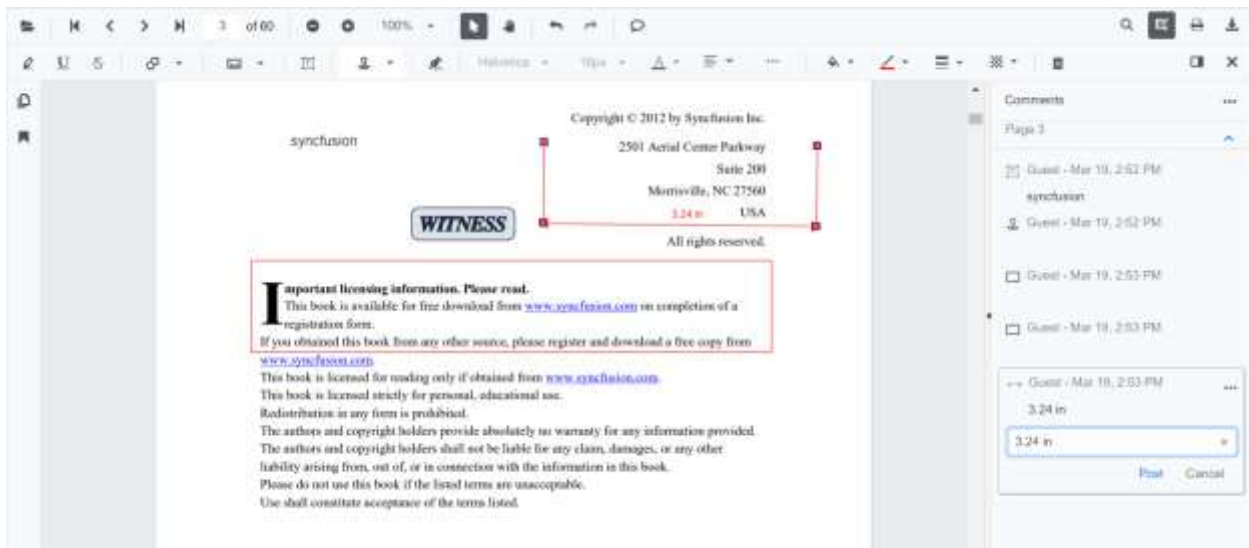

}

You can refer to our [Blazor PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor PDF Viewer example](#) to understand how to explain core features of PDF Viewer.

Comments in Blazor PDF Viewer Component

The PDF Viewer control provides options to add, edit, and delete the comments to the following annotation in the PDF documents:

- Shape annotation
- Stamp annotation
- Sticky note annotation
- Measurement annotation
- Text markup annotation
- Free text annotation



Adding a comment to the annotation

Annotation comment, comment replies, and status can be added to the PDF document using the comment panel.

Comment panel

Annotation comments can be added to the PDF using the comment panel. Comment panel can be opened by the following ways:

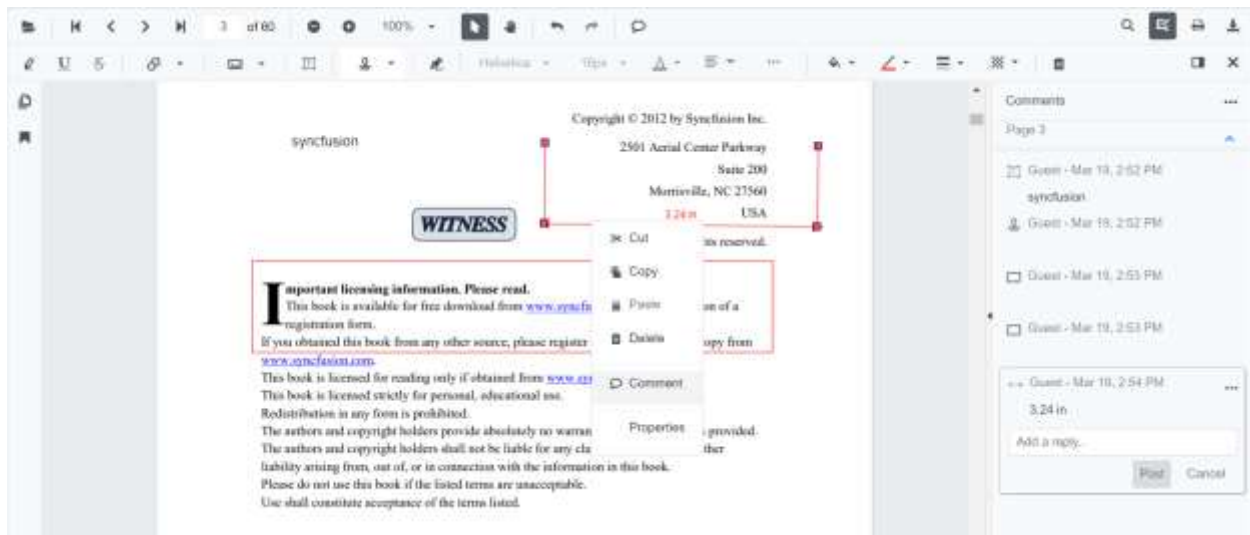
1. Using the annotation menu.
 - Click the Edit Annotation button in the PDF Viewer toolbar. A toolbar appears below it.
 - Click the Comment Panel button. A comment panel will appear
2. Using Context menu.
 - Select annotation in the PDF document and right-click it..
 - Select comment option in the context menu that appears.
3. Using Mouse click.

- Select annotation in the PDF document and double click it, a comment panel will appear.

If the comment panel is already in open state, you can select the annotations and add annotation comment using comment panel.

Adding comments

- Select annotation in the PDF document and click it.
- Selected annotation comment container is highlighted in the comment panel.
- Now, you can add comment and comment replies using comment panel.

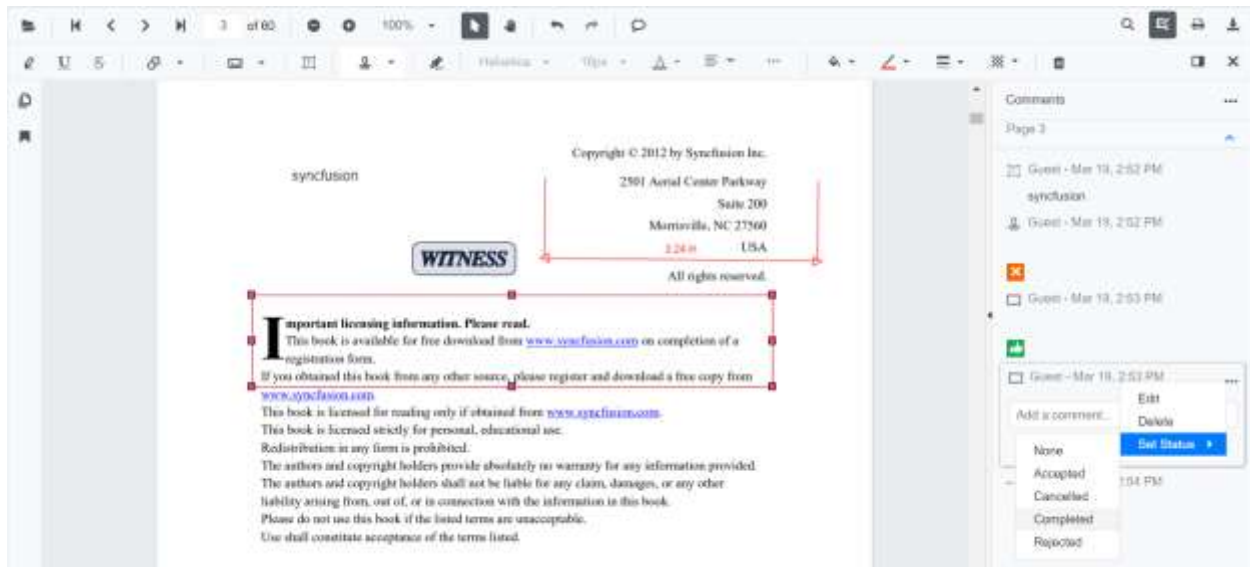


Adding Comment Replies

- PDF Viewer control provides an option to add multiple replies to the comment.
- After adding the annotation comment, you can add reply to the comment.

Adding Comment or Reply Status

- Select the Annotation Comments in the comment panel.
- Click the more options button showing in Comments or reply container.
- Select Set Status option in the context menu that appears.
- Select the status of the annotation comment in the context menu that appears.



Editing the comments and comments replies of the annotations

The comment, comment replies, and status of the annotation can be edited using the comment panel.

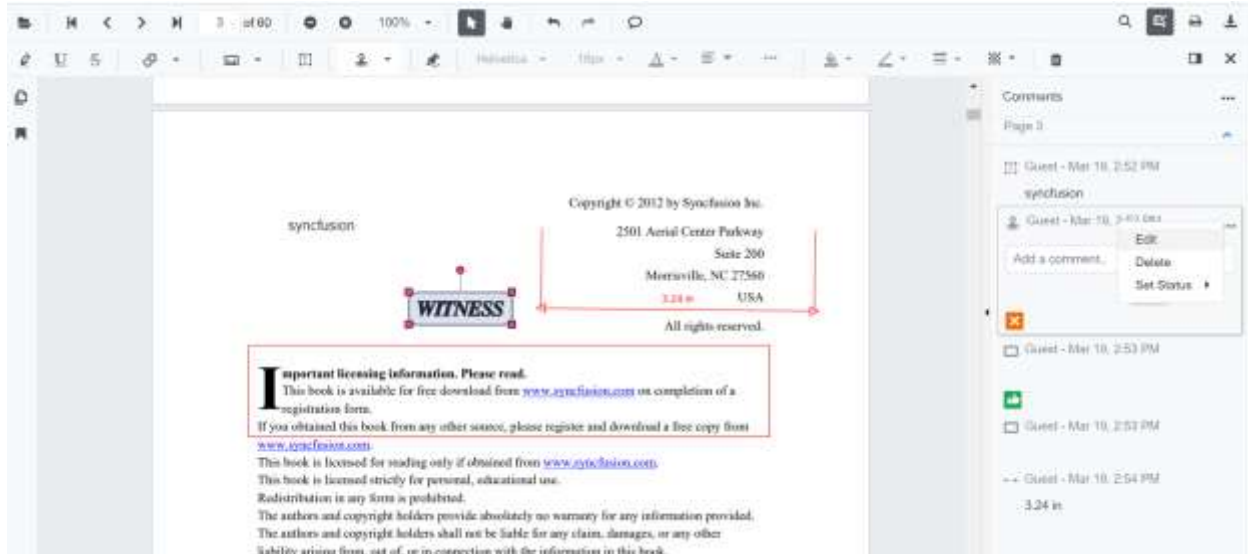
Editing Comment or Comment Replies

The annotation comment and comment replies can be edited by the following ways:

1. Using Context menu.
 - Select the Annotation Comments in comment panel.
 - Click the More option button showing in Comments or reply container.
 - Select Edit option in the context menu that appears.
 - Now, editable text box appears. You can change the content of the annotation comment or comment reply.
2. Using Mouse Click.
 - Select the annotation comments in comment panel.
 - Double click the comment or comment reply content.
 - Now, editable text box appears. You can change the content of the annotation comment or comment reply.

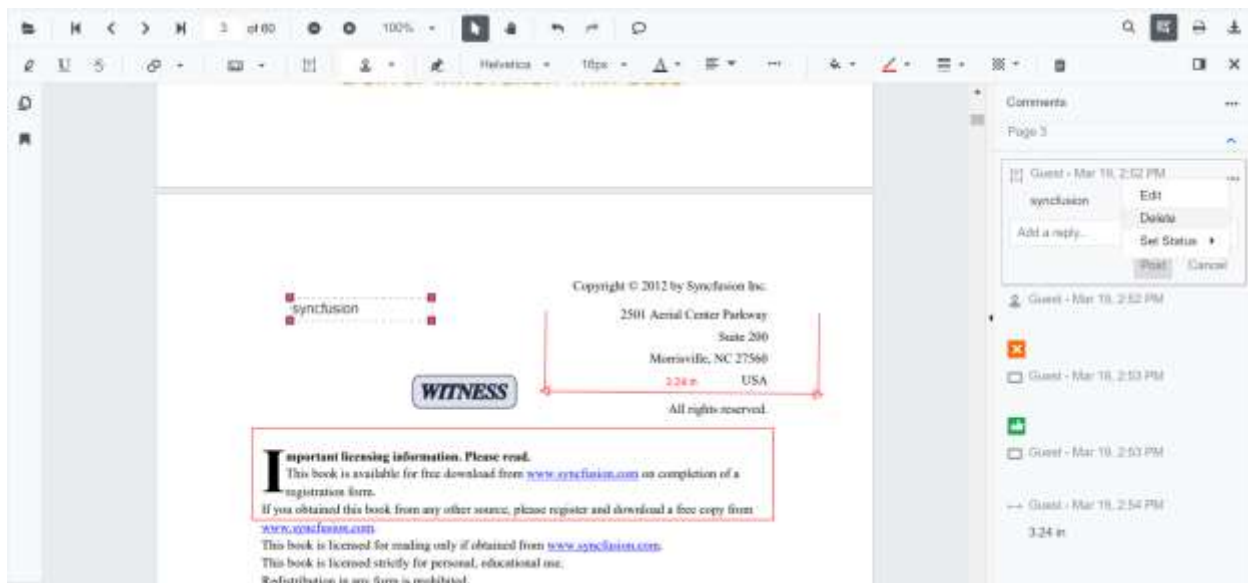
Editing Comment or Reply Status

- Select the Annotation Comments in comment panel.
- Click the more options button showing in Comments or reply container.
- Select Set Status option in the context menu that appears.
- Select the status of the annotation comment in the context menu that appears.
- Status 'None' is the default state. If status set to 'None', the comments or reply does not appear.



Delete Comment or Comment Replies

- Select the Annotation Comments in comment panel.
- Click the more options button shown in Comments or reply container.
- Select Delete option in the context menu that appears.



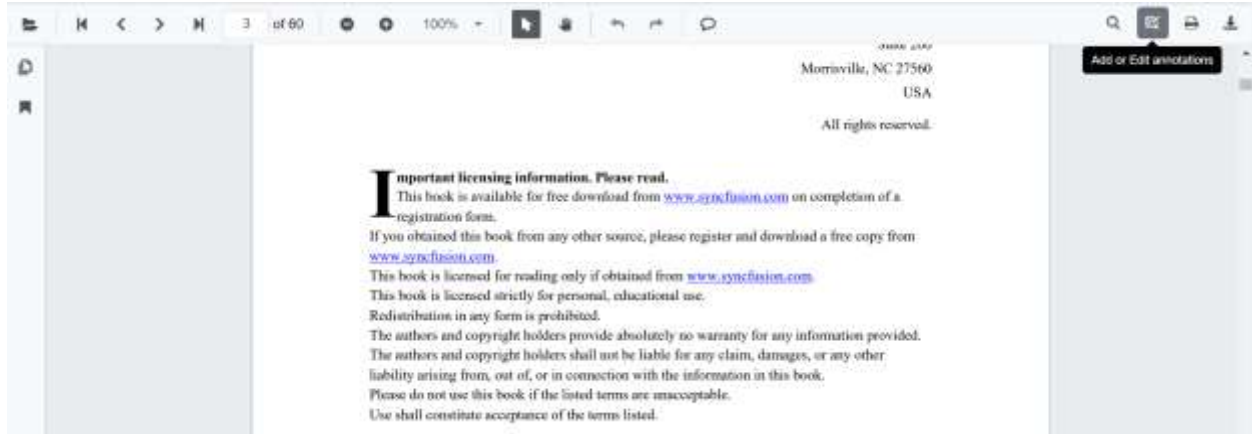
The annotation will be deleted on deleting the comment using comment panel.

You can refer to our [Blazor PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor PDF Viewer example](#) to understand how to explain core features of PDF Viewer.

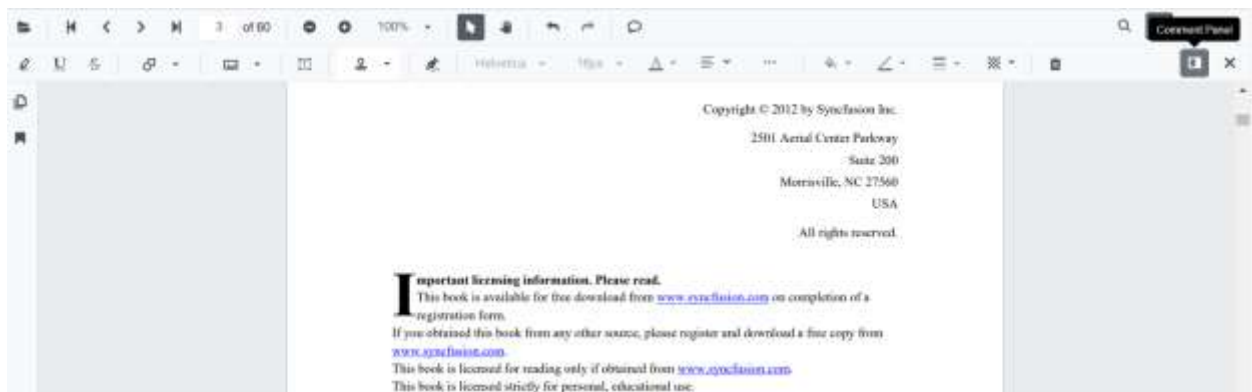
Import and Export annotations in Blazor PDF Viewer Component

The PDF Viewer control provides the support to import and export annotations using JSON object in the PDF document.

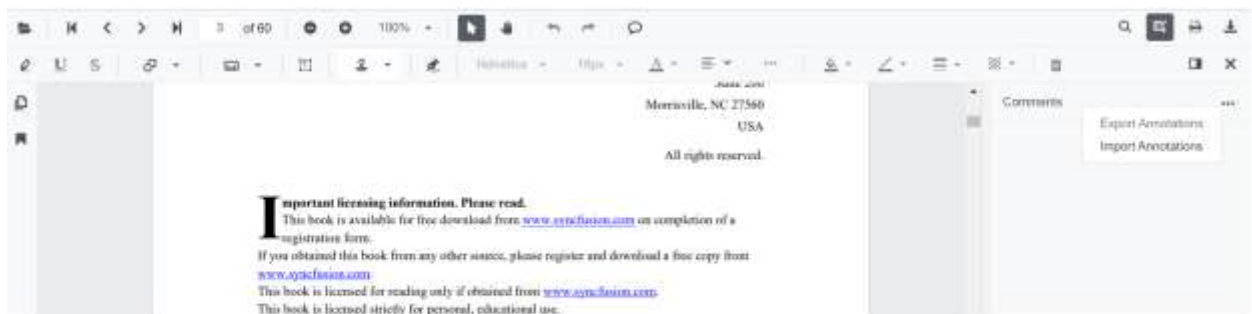
- Click the Add or Edit annotation button in the PDF Viewer toolbar.



- The annotation toolbar will appear.
- Click the Comment Panel button in the annotation toolbar.



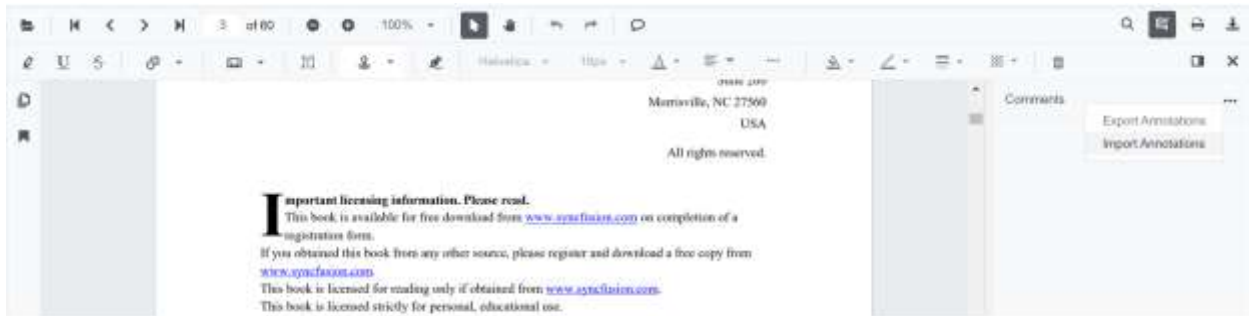
- The comments panel will displayed.
- Click the **More Option** button in the comment panel container.



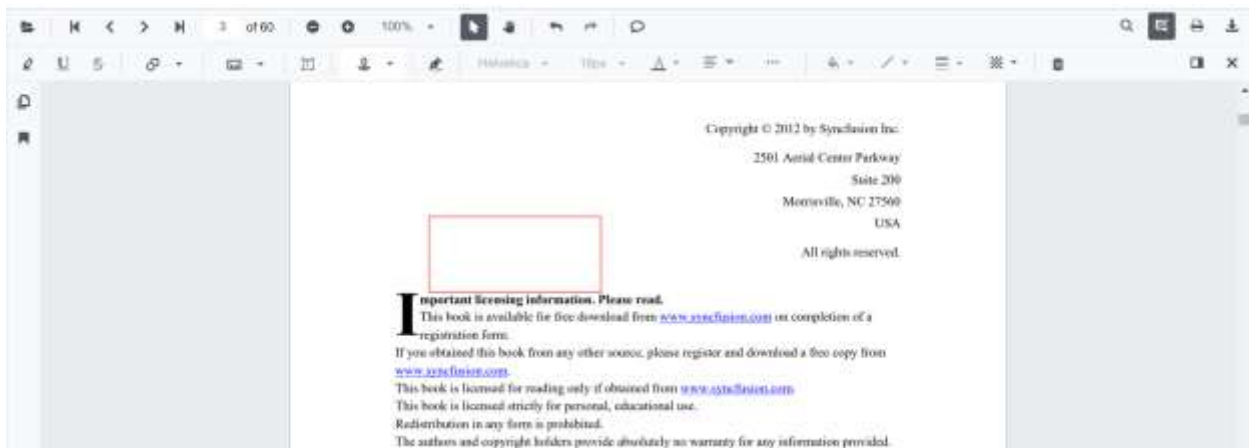
Importing annotation to the PDF document

- Click the Add or Edit annotation button in the PDF Viewer toolbar.

- The annotation toolbar will appear.
- Click the Comment Panel button in the annotation toolbar.
- The comments panel will displayed.
- Click the **More Option** button in the comment panel container.
- Select the Import Annotations Option.



- Then the file explorer dialog will opened. Choose the JSON file to be imported in to the loaded PDF document.



Importing annotation using PDF Viewer API

You can import annotations using JSON file or JSON object in code behind like the below code snippet

ASPX-CS

```
@using Syncfusion.Blazor.PdfViewerServer
@using Syncfusion.Blazor.Buttons
<SfButton OnClick="@OnImportAnnotationsClick">Import Annotations</SfButton>
<SfPdfViewerServer @ref=Viewer Width="1060px" Height="500px"
DocumentPath="@DocumentPath" />
@code{
    SfPdfViewerServer Viewer;
    public string DocumentPath { get; set; } =
        "wwwroot/data/PDF_Succinctly.pdf";
    public void OnImportAnnotationsClick(MouseEventArgs args)
    {
```

```
Viewer.ImportAnnotations("wwwroot/data/ImportedAnnotation.json"); //The json
file has been placed inside the data folder.
}
}
```

The JSON file for importing the annotation should be placed in the desired location and the path has to be provided correctly.

Exporting annotation from the PDF document

The PDF Viewer control provides the support to export the annotations as JSON file and JSON object using annotation toolbar.

- Click the Add or Edit annotation button in the PDF Viewer toolbar.
- The annotation toolbar will appear.
- Click the Comment Panel button in the annotation toolbar.
- The comments panel will displayed.
- Click the **More Option** button in the comment panel container.
- Select the Export Annotations Option.



Export annotations will be in the disabled state when the loaded PDF document does not contain any annotations.

Exporting annotation using PDF Viewer API

You can export annotations as JSON file in code behind like the following code snippet

ASPX-CS

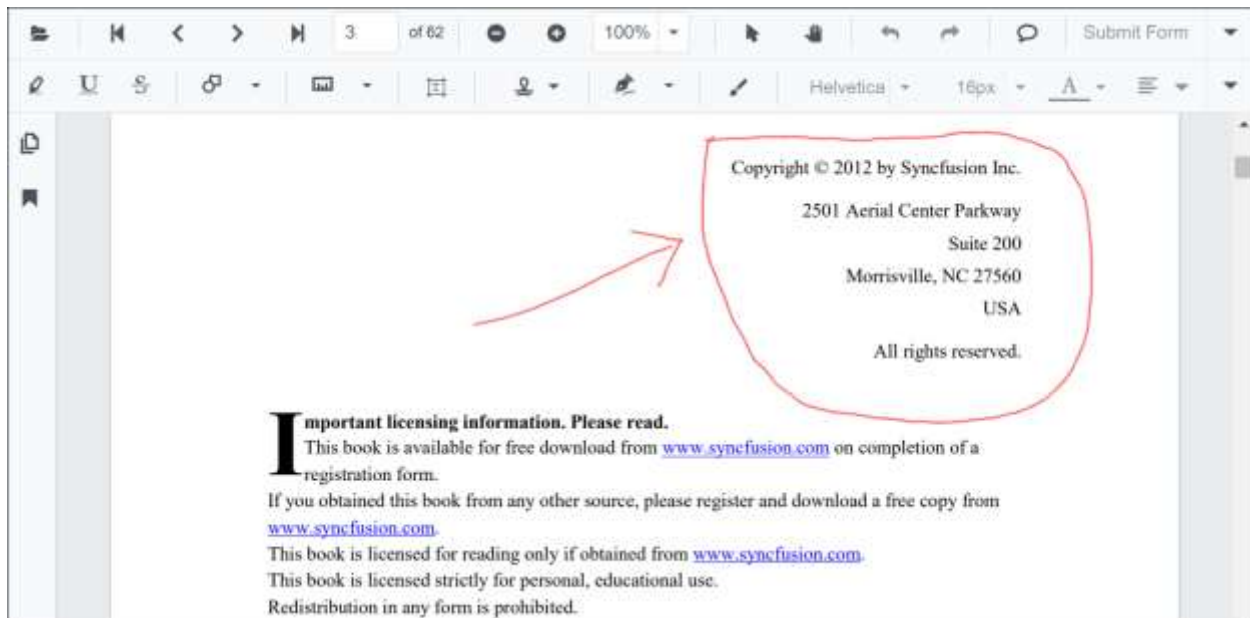
```
@using Syncfusion.Blazor.PdfViewerServer
@using Syncfusion.Blazor.Buttons
<SfButton OnClick="@OnExportAnnotationsClick">Export Annotations</SfButton>
<SfPdfViewerServer Width="1060px" Height="500px"
DocumentPath="@DocumentPath" @ref="@Viewer" />
@code{
SfPdfViewerServer Viewer;
public string DocumentPath { get; set; } =
"wwwroot/data/PDF_Succinctly.pdf";
public void OnExportAnnotationsClick(MouseEventArgs args)
{
Viewer.ExportAnnotations();
}
```

```
}  
}
```

You can refer to our [Blazor PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor PDF Viewer example](#) to understand how to explain core features of PDF Viewer.

Ink Annotation in the Blazor PDF Viewer component

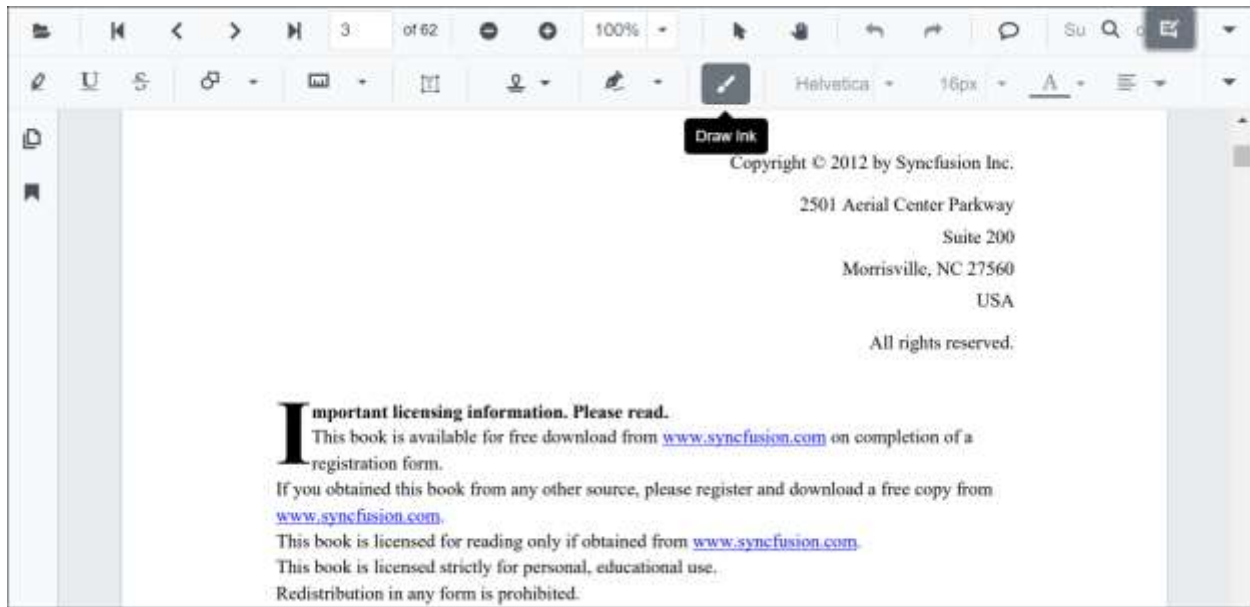
The PDF Viewer control provides the options to add, edit, and delete the ink annotations.



Adding an ink annotation to the PDF document

The ink annotations can be added to the PDF document using the annotation toolbar.

- Click the **Edit Annotation** button in the PDF Viewer toolbar. A toolbar appears below it.
- Select the **Draw Ink** button in the annotation toolbar. It enables the ink annotation mode.
- You can draw anything over the pages of the PDF document.



Refer to the following code sample to switch to the ink annotation mode.

CSHARP

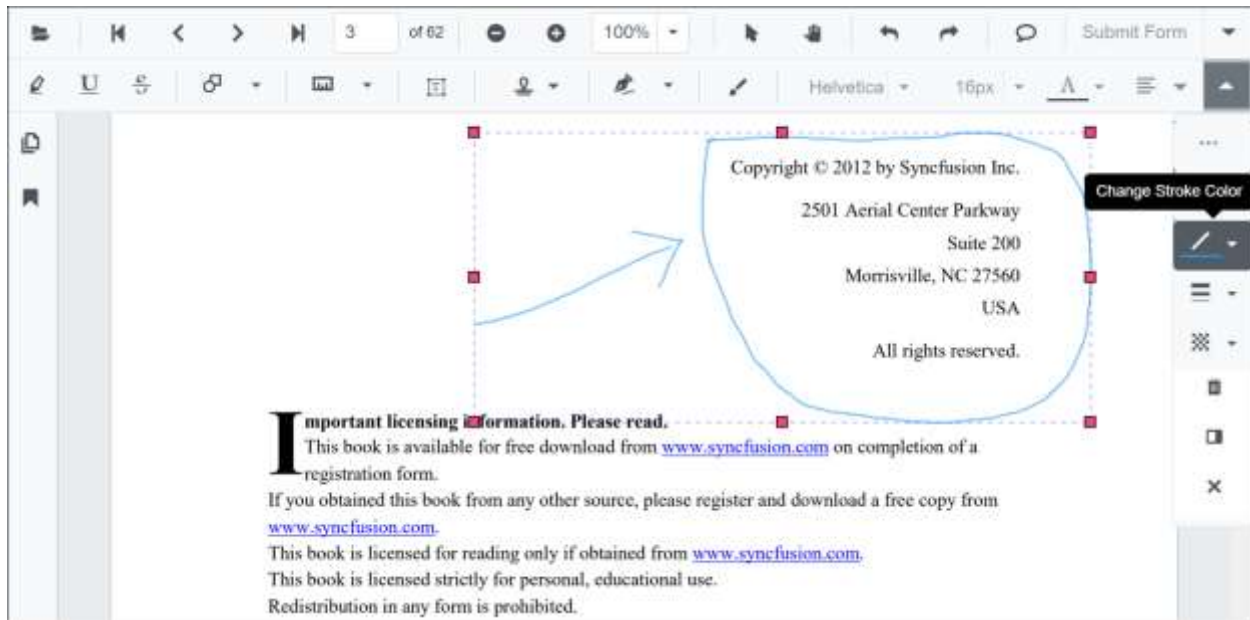
```
@using Syncfusion.Blazor.PdfViewer
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.PdfViewerServer
<SfButton OnClick="OnClick">Click Here</SfButton>
<SfPdfViewerServer DocumentPath="@DocumentPath" @ref="viewer" Height="640px"
Width="100%">
</SfPdfViewerServer>
@code{
SfPdfViewerServer viewer;
private string DocumentPath { get; set; } = "wwwroot/data/PDF
Succinctly.pdf";
public void OnClick(MouseEventArgs args)
{
viewer.SetAnnotationMode(AnnotationType.Ink);
}
}
```

Editing the properties of the ink annotation

The stroke color, thickness, and opacity of the ink annotation can be edited using the Edit stroke color tool, Edit thickness tool, and Edit opacity tool in the annotation toolbar.

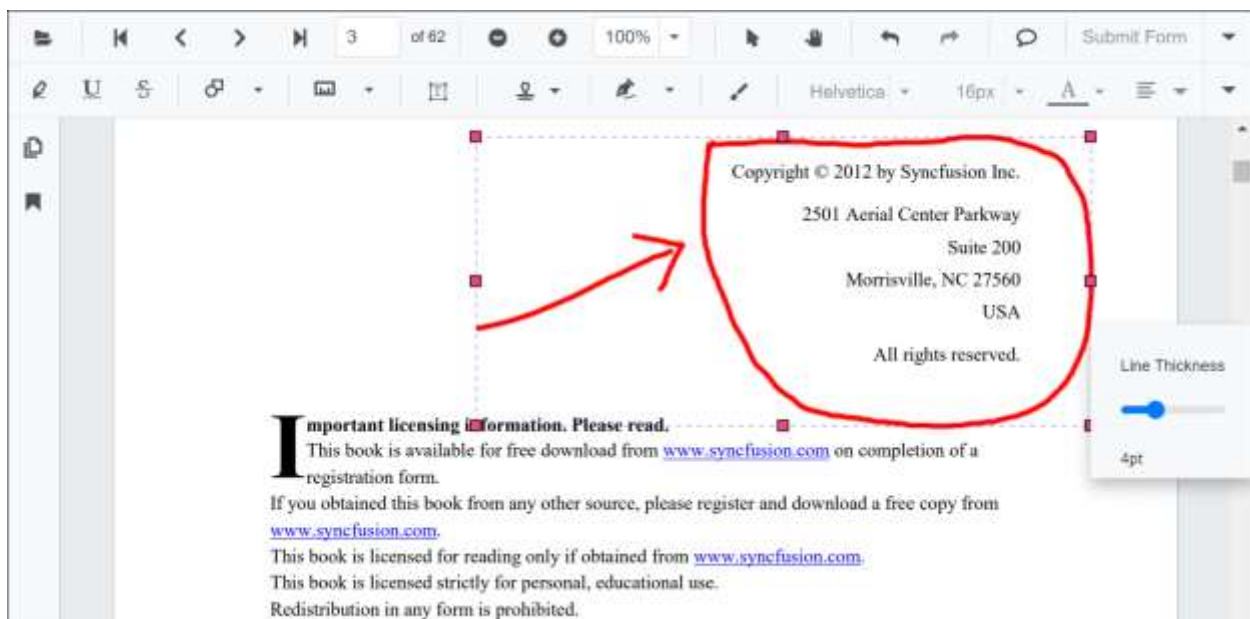
Editing stroke color

The stroke color of the annotation can be edited using the color palette provided in the Edit Stroke Color tool.



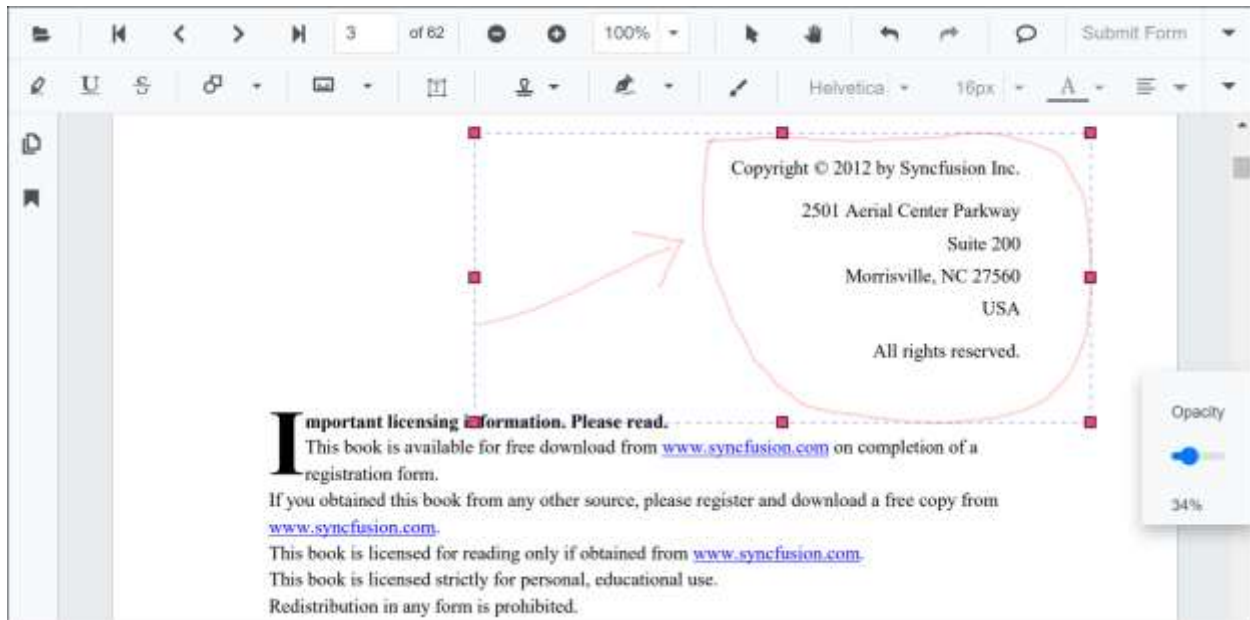
Editing thickness

The thickness of the border of the annotation can be edited using the range slider provided in the Edit Thickness tool.



Editing opacity

The opacity of the annotation can be edited using the range slider provided in the Edit Opacity tool.



Setting default properties during the control initialization

The properties of the ink annotation can be set before creating the control using the InkAnnotationSettings.

After editing the default values, they will be changed to the selected values.

Refer to the following code sample to set the default ink annotation settings.

CSHARP

```
@using Syncfusion.Blazor.PdfViewer
@using Syncfusion.Blazor.PdfViewerServer
<SfPdfViewerServer DocumentPath="@DocumentPath" @ref="viewer"
InkAnnotationSettings="@InkAnnotationSettings">
</SfPdfViewerServer>
@code{
SfPdfViewerServer viewer;
private string DocumentPath { get; set; } = "wwwroot/data/PDF
Succinctly.pdf";
PdfViewerInkAnnotationSettings InkAnnotationSettings = new
PdfViewerInkAnnotationSettings { Author="Syncfusion", StrokeColor="green",
Thickness=3, Opacity=0.6 };
}
```

Form filling in Blazor PDF Viewer Component

PDF Viewer component allows you to display the form fields available in the PDF document. By using this you can edit and download the form fields.

The form fields displayed in the PDF Viewer are:

- Text box
- Password box
- Combo box
- Check box

- Radio Button
- Signature Field
- List box

The screenshot shows a PDF document with a form. The form fields are: Name (text input), Email (text input), Gender (radio buttons for Male, Female, Unspecified), Date Of Birth (text input), Coming from (dropdown menu showing Alabama), Would you like to receive our Newsletters? (checkbox), and a Signature field with a red cursor.

Disabling form fields

The PDF Viewer control provides an option to disable the form fields feature . The code snippet for disabling the feature is as follows.

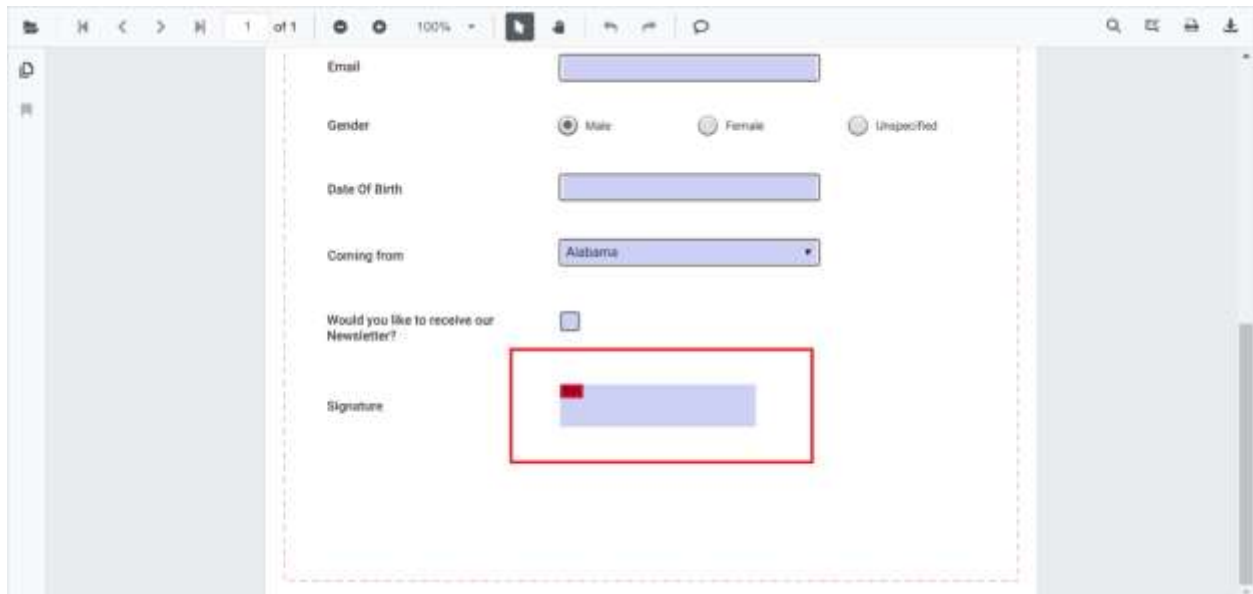
ASPX-CS

```
@using Syncfusion.Blazor.PdfViewerServer
<SfPdfViewerServer Width="1060px" Height="500px" EnableFormFields=false />
@code{
    public string DocumentPath { get; set; } =
    "wwwroot/data/PDF_Succinctly.pdf";
}
```

How to draw handwritten signature in the signature field

Signature can be added to the Signature field by using the following steps:

- Click the Signature Field in the PDF document. The signature panel will appear.




The screenshot shows a PDF form with the following fields:

- Email:
- Gender: ☒ Male ☐ Female ☐ Unspecified
- Date Of Birth:
- Coming from:
- Would you like to receive our Newsletter?: ☐
- Signature: (highlighted with a red rectangle, showing a small red cursor icon)

- Draw the signature in the signature panel.




- Click the **CREATE** button, the drawn signature will be added in the signature field.

Email:
 Gender: ☒ Male ☐ Female ☐ Unspecified
 Date Of Birth:
 Coming from:
 Would you like to receive our Newsletter? ☐
 Signature: 

Delete the signature inside the signature field

You can also delete the signature in the signature field by using Delete Option in the annotation toolbar.

Email:
 Gender: ☒ Male ☐ Female ☐ Unspecified
 Date Of Birth:
 Coming from:
 Would you like to receive our Newsletter? ☐
 Signature: 

Import and export FormFields

The PDF Viewer control provides the support to import and export form fields using a JSON object in the PDF document.

Importing FormFields using PDF Viewer API

You can import the form fields using JSON file or JSON object in code behind like the below code snippet

ASPX-CS

```

@using Syncfusion.Blazor.PdfViewerServer
@using Syncfusion.Blazor.Buttons
<SfButton OnClick="@OnImportFormFieldsClick">Import FormFields</SfButton>
<SfPdfViewerServer @ref=Viewer Width="1060px" Height="500px"
DocumentPath="@DocumentPath" />
@code{

```

```
SfPdfViewerServer Viewer;
public string DocumentPath { get; set; } =
"wwwroot/data/FormFillingDocument.pdf";
public void OnImportFormFieldsClick(MouseEventArgs args)
{
Viewer.ImportFormFields("wwwroot/data/ImportedFormFields.json"); //The json
file has been placed inside the data folder.
}
}
```

The JSON file for importing the form fields should be placed in the desired location and the path should be provided correctly.

Exporting FormFields from the PDF document using PDF Viewer API

You can export the form fields as JSON file in code behind as the following code snippet

ASPX-CS

```
@using Syncfusion.Blazor.PdfViewerServer
@using Syncfusion.Blazor.Buttons
<SfButton OnClick="@OnExportFormFieldsClick">Export FormFields</SfButton>
<SfPdfViewerServer Width="1060px" Height="500px"
DocumentPath="@DocumentPath" @ref="@Viewer" />
@code{
SfPdfViewerServer Viewer;
public string DocumentPath { get; set; } =
"wwwroot/data/PDF_Succinctly.pdf";
public void OnExportFormFieldsClick(MouseEventArgs args)
{
Viewer.ExportFormFields();
}
}
```

You can refer to our [Blazor PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor PDF Viewer example](#) to understand how to explain core features of PDF Viewer.

Handwritten Signature in Blazor PDF Viewer Component

The PDF Viewer control supports adding handwritten signatures to a PDF document. The handwritten signature reduces the paper work of reviewing the content and verifies it digitally.

Adding a handwritten signature to the PDF document

The handwritten signature can be added to the PDF document using the annotation toolbar.

- Click the **Edit Annotation** button in the PDF Viewer toolbar. A toolbar appears below it.
- Select the **HandWritten Signature** button in the annotation toolbar. The signature panel will appear.

Form **8850**
(Rev. March 2016)
Department of the Treasury
Internal Revenue Service

OMB No. 1545-1500

Information about Form 8850 and its separate instructions is at www.irs.gov/form8850.

Job applicant: Fill in the lines below and check any boxes that apply. Complete only this side.

Your name Social security number

Street address where you live

City or town, state, and ZIP code

County Telephone number

If you are under age 40, enter your date of birth (month, day, year)

1 ☐ Check here if you received a conditional certification from the state workforce agency (SWA) or a participating local agency for the work opportunity credit.

2 ☐ Check here if any of the following statements apply to you.

- Draw the signature in the signature panel.

Draw Signature

Clear Cancel Create

- Then click **Create** button and move the signature using the mouse and place them in the desired location.

Street address

City or town, state, and ZIP code

If, based on the individual's age and home address, he or she is a member of group 4 or 6 (as described under *Members of Targeted Groups* in the separate instructions), enter that group number (4 or 6)

Date applicant:
Gave information Was offered job Was hired Started job

Under penalties of perjury, I declare that the applicant provided the information on this form on or before the day a job was offered to the applicant and that the information I have furnished is, to the best of my knowledge, true, correct, and complete. Based on the information the job applicant furnished on page 1, I believe the individual is a member of a targeted group. I hereby request a certification that the individual is a member of a targeted group.

Employer's signature Title Date

Privacy Act and Paperwork Reduction Act Notice

Section references are to the Internal Revenue Code.

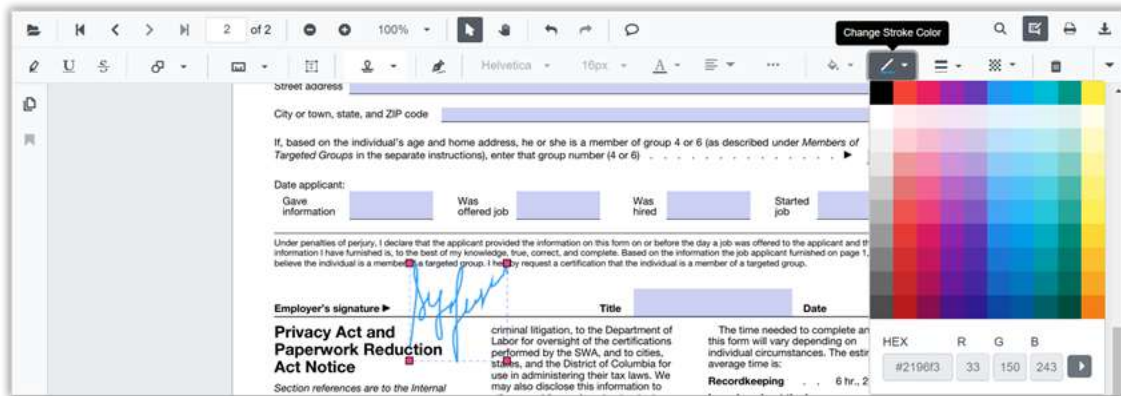
criminal litigation, to the Department of Labor for oversight of the certifications performed by the SWA, and to cities, states, and the District of Columbia for use in administering their tax laws. We may also disclose this information to other countries under a tax treaty, to federal and state agencies to enforce

The time needed to complete and file this form will vary depending on individual circumstances. The estimated average time is:

Recordkeeping 6 hr., 27 min.
Learning about the law or the form 24 min.

Editing the properties of handwritten signature

The stroke color, border thickness, and opacity of the handwritten signature can be edited using the edit stroke color tool, edit thickness tool, and edit opacity tool in the annotation toolbar.



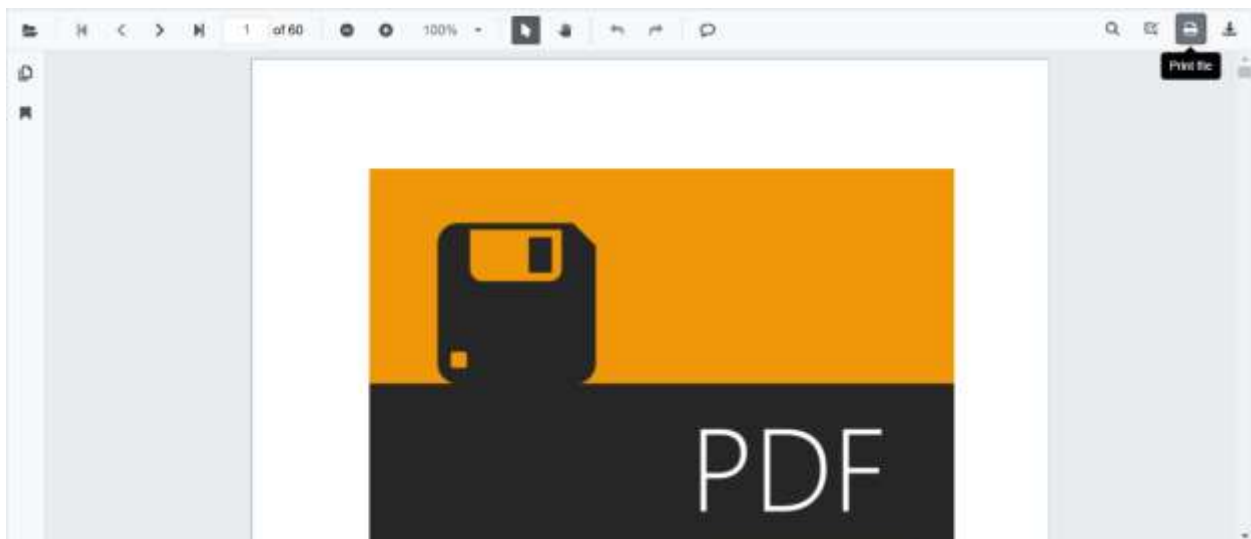
You can refer to our [Blazor PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor PDF Viewer example](#) to understand how to explain core features of PDF Viewer.

Print in Blazor PDF Viewer Component

The PDF Viewer supports printing the loaded PDF file by default. You can enable or disable printing by setting the `EnablePrint` property.

ASPX-CS

```
@using Syncfusion.Blazor.PdfViewerServer
<SfPdfViewerServer Width="1060px" Height="500px"
DocumentPath="@DocumentPath" EnablePrint="true"/>
@code{
public string DocumentPath { get; set; } =
"wwwroot/data/PDF_Succinctly.pdf";
}
```



You can programmatically invoke print action as follows.

ASPX-CS

```
@using Syncfusion.Blazor.PdfViewerServer
@using Syncfusion.Blazor.Buttons
<SfButton OnClick="OnClick">Print</SfButton>
<SfPdfViewerServer Width="1060px" Height="500px"
DocumentPath="@DocumentPath" @ref="@Viewer"/>
@code{
SfPdfViewerServer Viewer;
public string DocumentPath { get; set; } =
"wwwroot/data/PDF_Succinctly.pdf";
public void OnClick(MouseEventArgs args)
{
Viewer.Print();
}
}
```

You can refer to our [Blazor PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor PDF Viewer example](#) to understand how to explain core features of PDF Viewer.

Download in Blazor PDF Viewer Component

The PDF Viewer supports downloading the loaded PDF file from the toolbar by default. You can enable or disable the download option by setting the `EnableDownload` API.

ASPX-CS

```
@using Syncfusion.Blazor.PdfViewerServer
<SfPdfViewerServer Width="1060px" Height="500px"
DocumentPath="@DocumentPath" EnableDownload="true"/>
@code{
public string DocumentPath { get; set; } =
"wwwroot/data/PDF_Succinctly.pdf";
}
```



And, you can programmatically invoke download action as follows.

ASPX-CS

```

@using Syncfusion.Blazor.PdfViewerServer
@using Syncfusion.Blazor.Buttons
<SfButton OnClick="OnClick">Download</SfButton>
<SfPdfViewerServer Width="1060px" Height="500px"
DocumentPath="@DocumentPath" @ref="@Viewer"/>
@code{
SfPdfViewerServer Viewer;
public string DocumentPath { get; set; } =
"wwwroot/data/PDF_Succinctly.pdf";
public void OnClick(MouseEventArgs args)
{
Viewer.Download();
}
}

```

You can refer to our [Blazor PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor PDF Viewer example](#) to understand how to explain core features of PDF Viewer.

Events in Blazor PDF Viewer Component

The following section list out the events provided in PDF Viewer component:

Name	Description
---	---
DocumentLoaded	Triggers when the document is opened or loaded into the PDF Viewer.
DcoumentUnloaded	Triggers when the document gets closed.
DocumentLoadFailed	Triggers when the document loading gets failed.
AjaxRequestFailed	Triggers when the request gets failed in case of server-side component. Triggers when the communication between service and viewer gets failed in case of client-side component
OnPageClick	Triggers when the click or tap is performed over the page of the PDF document.
PageChanged	Triggers when there is change in current page number.
OnHyperlinkClick	Triggers when the hyperlink (TOC entries and links) in the PDF Document is clicked
ZoomChanged	Triggers when there is change in the magnification value.
AnnotationAdded	Triggers when an annotation is added over the page of the PDF document.
AnnotationRemoved	Triggers when an annotation is removed from the page of the PDF document.
AnnotationPropertiesChanged	Triggers when the property of the annotation is changed.
AnnotationResized	Triggers when an annotation is resized.
AnnotationSelected	Triggers when an annotation is selected.

Adding PDF Viewer events to Blazor component

The Syncfusion PDF Viewer events has to be wrapped inside the PdfViewerEvents tag. Refer to the following code snippet to use the DocumentLoaded event.

ASPX-CS

```
<SfPdfViewerServer DocumentPath="@DocumentPath" Height="500px"
Width="1060px" >
<PdfViewerEvents DocumentLoaded="@DocumentLoaded"></PdfViewerEvents>
</SfPdfViewerServer>
@code{
private string DocumentPath { get; set; } =
"wwwroot/Data/PDF_Succinctly.pdf";
private void DocumentLoaded(LoadEventArgs args)
{
object PageData = args.PageData;
}
}
```

You can refer to our [Blazor PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor PDF Viewer example](#) to understand how to explain core features of PDF Viewer.

Globalization and RTL in Blazor PDF Viewer Component

The PDF Viewer component allows you to localize the static text on formatting pane, toolbar, dialog and more. It can be achieved by setting the **Locale** property and providing the localized text through the **LoadLocaleData()** method.

Also, this component provides support to render the user interface suitable for users who use **right-to-left (RTL)** languages (Arabic, Hebrew, Azerbaijani, Persian, Urdu). You can specify the control to render in RTL by setting the **EnableRtl** property to true.

The following code snippet shows how to localize the component for Arabic language by setting the **Locale** and **EnableRtl** properties and providing the localized text.

ASPX-CS

```
@using Syncfusion.Blazor
@using Syncfusion.Blazor.PdfViewerServer
<SfPdfViewerServer Width="1060px" Height="500px"
DocumentPath="@DocumentPath" EnableRtl="true" Locale="ar-AE" />
@code{
public string DocumentPath { get; set; } =
"wwwroot/data/PDF_Succinctly.pdf";
[Inject]
protected IJSRuntime JsRuntime { get; set; }
protected override void OnAfterRender(bool firstRender)
{
this.JsRuntime.Sf().LoadLocaleData("wwwroot/locale.json");
}
}
```

The following table shows the default text values used in PDF Viewer in 'en-US' culture:

Keywords	Values
----------	--------

---	---
-----	-----

PdfViewer	PDF Viewer
-----------	------------

Cancel	Cancel
Download file	Download file
Download	Download
Enter Password	This document is password protected. Please enter a password.
File Corrupted	File corrupted
File Corrupted Content	The file is corrupted and cannot be opened.
Fit Page	Fit page
Fit Width	Fit width
Automatic	Automatic
Go To First Page	Show first page
Invalid Password	Incorrect password. Please try again.
Next Page	Show next page
OK	OK
Open	Open file
Page Number	Current page number
Previous Page	Show previous page
Go To Last Page	Show last page
Zoom	Zoom
Zoom In	Zoom in
Zoom Out	Zoom out
Page Thumbnails	Page thumbnails
Bookmarks	Bookmarks
Print	Print file
Password Protected	Password required
Copy	Copy
Text Selection	Text selection tool
Panning	Pan mode
Text Search	Find text
Find in document	Find in document
Match case	Match case
Apply	Apply
GoToPage	Go to page
No matches	Viewer has finished searching the document. No more matches were found

No Text Found	No Text Found
Undo	Undo
Redo	Redo
Annotation	Add or Edit annotations
Highlight	Highlight Text
Underline	Underline Text
Strikethrough	Strikethrough Text
Delete	Delete annotation
Opacity	Opacity
Color edit	Change Color
Opacity edit	Change Opacity
Highlight context	Highlight
Underline context	Underline
Strikethrough context	Strike through
Server error	Web-service is not listening. PDF Viewer depends on web-service for all it's features.
Please start the web service to continue.	
Open text	Open
First text	First Page
Previous text	Previous Page
Next text	Next Page
Last text	Last Page
Zoom in text	Zoom In
Zoom out text	Zoom Out
Selection text	Selection
Pan text	Pan
Print text	Print
Search text	Search
Annotation Edit text	Edit Annotation
Line Thickness	Line Thickness
Line Properties	Line Properties
Start Arrow	Start Arrow
End Arrow	End Arrow
Line Style	Line Style

Fill Color	Fill Color
Line Color	Line Color
None	None
Open Arrow	Open
Closed Arrow	Closed
Round Arrow	Round
Square Arrow	Square
Diamond Arrow	Diamond
Cut	Cut
Paste	Paste
Delete Context	Delete
Properties	Properties
Add Stamp	Add Stamp
Add Shapes	Add Shapes
Stroke edit	Change Stroke Color
Change thickness	Change Border Thickness
Add line	Add Line
Add arrow	Add Arrow
Add rectangle	Add Rectangle
Add circle	Add Circle
Add polygon	Add Polygon
Add Comments	Add Comments
Comments	Comments
No Comments Yet	No Comments Yet
Accepted	Accepted
Completed	Completed
Cancelled	Cancelled
Rejected	Rejected
Leader Length	Leader Length
Scale Ratio	Scale Ratio
Calibrate	Calibrate
Calibrate Distance	Calibrate Distance
Calibrate Perimeter	Calibrate Perimeter

Calibrate Area	Calibrate Area
Calibrate Radius	Calibrate Radius
Calibrate Volume	Calibrate Volume
Depth	Depth
Closed	Closed
Round	Round
Square	Square
Diamond	Diamond
Edit	Edit
Set Status	Set Status
Post	Post
Page	Page
Add a comment	Add a comment
Add a reply	Add a reply

You can refer to our [Blazor PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor PDF Viewer example](#) to understand how to explain core features of PDF Viewer.

How To

Create PDF Viewer in a popup window in Blazor PDF Viewer Component

For quick view, you might need to display the PDF file in a dialog window. The following code snippet explains how to use the PDF Viewer component inside a dialog window. In this example, the Syncfusion's dialog component is used for Blazor.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.Popups
@using Syncfusion.Blazor.PdfViewerServer
<div id="target" style="width:800px;height:500px">
  <SfButton @onclick="OnClick">Open PDF Viewer</SfButton>
  <SfDialog @ref="@dialog" Target="#target" Width="1060px" Visible="false"
    IsModal="true" Header=" @Header" ShowCloseIcon="true">
    <DialogEvents OnOpen="OnOpen"></DialogEvents>
    <SfPdfViewerServer @ref="@viewer"/>
  </SfDialog>
</div>
@code{
  SfPdfViewerServer viewer;
  SfDialog dialog;
  public void OnClick(MouseEventArgs args)
  {
    this.dialog.Show();
  }
  public void OnOpen(BeforeOpenEventArgs args)
  {
  }
```

```
viewer.Load(DocumentPath, null);
}
public string DocumentPath { get; set; } =
"wwwroot/data/PDF_Succinctly.pdf";
public string Header { get; set; } = "PDF Viewer";
}
```

You can refer to our [Blazor PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor PDF Viewer example](#) to understand how to explain core features of PDF Viewer.

Load PDF documents dynamically in Blazor PDF Viewer Component

At times, you might need to switch or load the PDF documents dynamically after the initial load operation. To achieve this, load the PDF document as a base64 string or file path in PDF Viewer control using the `Load()` method dynamically.

The following code example shows how to load a base64 string dynamically.

ASPX-CS

```
@using Syncfusion.Blazor.PdfViewerServer
@using Syncfusion.Blazor.Buttons
<SfButton @onclick="clicked">Load Document</SfButton>
<SfPdfViewerServer DocumentPath="@DocumentPath" Height="500px"
Width="1060px" @ref="Viewer"></SfPdfViewerServer>
@code{
SfPdfViewerServer Viewer;
private string DocumentPath { get; set; } =
"wwwroot/Data/PDF_Succinctly.pdf";
public void clicked()
{
byte[] byteArray =
System.IO.File.ReadAllBytes("wwwroot/Data/Python_Succinctly.pdf");
string base64String = Convert.ToBase64String(byteArray);
Viewer.Load("data:application/pdf;base64," + base64String, null);
}
}
```

The following code example shows how to load the PDF dynamically by specifying file path.

ASPX-CS

```
@using Syncfusion.Blazor.PdfViewerServer
@using Syncfusion.Blazor.Buttons
<SfButton @onclick="clicked">Load Document</SfButton>
<SfPdfViewerServer DocumentPath="@DocumentPath" Height="500px"
Width="1060px" @ref="Viewer"></SfPdfViewerServer>
@code{
SfPdfViewerServer Viewer;
private string DocumentPath { get; set; } =
"wwwroot/data/PDF_Succinctly.pdf";
public void clicked()
{
Viewer.Load("wwwroot/data/Python_Succinctly.pdf", null);
}
}
```

```
}
```

You can refer to our [Blazor PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor PDF Viewer example](#) to understand how to explain core features of PDF Viewer.

View DOCX file in Blazor application in Blazor PDF Viewer Component

You can achieve this requirement of viewing DOCX file in two ways:

Option 1:

You can use Syncfusion's Word Processor component for Blazor to compose, view, and edit DOC, DOCX, RTF, and SFDT file formats.

To learn more about the [Word Processor component](#), check out Syncfusion online samples and documentation.

Option 2:

You can convert [Word document to PDF](#) using the Syncfusion's Word (DocIO) server-side library and view the resultant PDF file using PDF Viewer component.

You can refer to our [Blazor PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor PDF Viewer example](#) to understand how to explain core features of PDF Viewer.

Unload the PDF document from Viewer in Blazor PDF Viewer Component

The PDF Viewer component will automatically unload and clear the resources occupied by the PDF document when the control is disposed. Also, when loading another PDF file, the resources occupied by previous loaded file in viewer will be automatically unloaded and cleared.

If you want to unload and clear the resources occupied by the PDF file programmatically, invoke the `Unload()` method as shown below.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.PdfViewerServer
<SfButton @onclick="OnClick">Unload Document</SfButton>
<SfPdfViewerServer @ref="@viewer" Height="500px" Width="1060px"
DocumentPath="@DocumentPath" />
@code{
    SfPdfViewerServer viewer;
    public void OnClick(MouseEventArgs args)
    {
        viewer.Unload();
    }
    public string DocumentPath { get; set; } =
        "wwwroot/data/PDF_Succinctly.pdf";
}
```

You can refer to our [Blazor PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor PDF Viewer example](#) to understand how to explain core features of PDF Viewer.

Import annotations as objects in Blazor PDF Viewer Component

The Syncfusion's Blazor PDF Viewer component allows to import annotations from objects or streams instead of loading it as a file. To import such annotation objects, the PDF Viewer control must export the PDF annotations as objects using the [ExportAnnotationsAsObject\(\)](#) method. Only the annotations objects that are exported from the PDF Viewer can be imported.

The following code example shows how to import annotations as objects, that are exported using the [ExportAnnotationsAsObject\(\)](#) method.

ASPX-CS

```
@page "/"
@using Syncfusion.Blazor.PdfViewerServer
@using Syncfusion.Blazor.PdfViewer
<button @onclick="ExportAnnot">ExportAnnot</button>
<button @onclick="ImportAnnot">ImportAnnot</button>
<SfPdfViewerServer @ref="PdfViewer" DocumentPath="@DocumentPath"
Height="500px" Width="1060px">
</SfPdfViewerServer>
@code {
public SfPdfViewerServer PdfViewer { get; set; }
private string DocumentPath { get; set; } = "PDF_Succinctly.pdf";
public object annotation;
//Export the annotations as an object
public async void ExportAnnot()
{
await PdfViewer.ExportAnnotation();
annotation = await PdfViewer.ExportAnnotationsAsObject();
}
//Import the annotations that are exported as objects
public async void ImportAnnot()
{
await PdfViewer.ImportAnnotation(annotation);
}
}
```

You can refer to our [Blazor PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor PDF Viewer example](#) to understand how to explain core features of PDF Viewer.

Suppress the error dialog in Blazor PDF Viewer Component

The Syncfusion's Blazor PDF Viewer component allows you to suppress or disable the error dialog box displayed in the PDF Viewer using the [EnableErrorDialog](#) property. The default value of the property is `true`.

The following code example shows how to suppress the error dialog.

ASPX-CS

```
@using Syncfusion.Blazor.PdfViewer
<SfPdfViewer @ref="PdfViewer" ServiceUrl="https://localhost:44399/pdfviewer"
DocumentPath="@DocumentPath" EnableErrorDialog="false" Height="500px"
Width="1060px">
</SfPdfViewer>
@code{
SfPdfViewer PdfViewer;
```

```
private string DocumentPath { get; set; } = "PDF_Succinctly.pdf";
}
```

Find the sample, [How to suppress the error dialog in the Blazor PDF Viewer](#)

You can refer to our [Blazor PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor PDF Viewer example](#) to understand how to explain the core features of the PDF Viewer.

Include the Authorization token in Blazor PDF Viewer Component

The Syncfusion's Blazor PDF Viewer component allows to include the authorization token in the PDF viewer AJAX request using the properties of the ajaxRequest header available in [AjaxRequestSettings](#), and it will be included in every AJAX request send from PDF Viewer.

The following code example shows how include the authorization token.

ASPX-CS

```
@using Syncfusion.Blazor.PdfViewer
<SfPdfViewer @ref="PdfViewer"
  ServiceUrl="https://ej2services.syncfusion.com/production/web-
  services/api/pdfviewer" DocumentPath="@DocumentPath"
  AjaxRequestSettings="@AjaxRequestSettings" Height="500px" Width="1060px">
</SfPdfViewer>
@code{
  SfPdfViewer PdfViewer;
  private string DocumentPath { get; set; } = "PDF_Succinctly.pdf";
  public PdfViewerAjaxRequestSettings AjaxRequestSettings = new
  PdfViewerAjaxRequestSettings{
    AjaxHeaders = new List<AjaxHeader>() {
      new AjaxHeader {
        HeaderName = "Authorization",
        HeaderValue = "Bearer 64565dfgfsjweiuvbiuyhiueygf"
      },
    },
    WithCredentials = false
  };
}
```

AjaxRequestSettings is applicable for Web assembly blazor alone.

Find the sample, [How to include the authorization token](#)

You can refer to our [Blazor PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor PDF Viewer example](#) to understand how to explain core features of PDF Viewer.

Move the scrollbar to the exact location of annotations

The Syncfusion's Blazor PDF Viewer component allows you to move the scrollbar to the exact location of annotations present in a loaded PDF document using the **GoToBookmark** method.

The following code example shows how to move the scrollbar to annotation location.

CSHARP

```
<button @onclick="navigate">Navigation</button>
```

```

<SfPdfViewerServer @ref="PdfViewer" DocumentPath="@DocumentPath">
<PdfViewerEvents DocumentLoaded="DocumentLoad"></PdfViewerEvents>
</SfPdfViewerServer>
@code{
SfPdfViewerServer PdfViewer;
public string DocumentPath { get; set; } = "wwwroot/data/PDF
Succinctly.pdf";
public Dictionary<int, System.Drawing.SizeF> pageSize { get; set; }
private void DocumentLoad(LoadEventArgs args)
{
    pageSize = args.PageData.PageSizes;
}
private async void navigate()
{
    var annotationCollection = await PdfViewer.GetAnnotations();
    var pageNumber = (annotationCollection[0].PageNumber);
    var Y = annotationCollection[0].Bound.Top;
    await PdfViewer.GoToBookmark(pageNumber, (pageSize[pageNumber].Height - Y));
}
}

```

Find the sample [How to move the scrollbar to exact location of annotations](#)

View the created PDF document

The Syncfusion's Blazor PDF Viewer component allows you to view the created PDF document using the [Created](#) event.

The following code example shows how to view the created PDF document.

CSHARP

```

<SfPdfViewerServer ID="pdfviewer" @ref="@PdfViewer"
DocumentPath="@documentPath">
<PdfViewerEvents Created="created"></PdfViewerEvents>
</SfPdfViewerServer>
@code{
public SfPdfViewerServer PdfViewer { get; set; }
public string documentPath { get; set; }
private void created()
{
    var document = new PdfDocument();
    byte[] bytes;
    //Add a new page to the PDF document.
    PdfPage page = document.Pages.Add();
    //Create a textbox field and add the properties.
    PdfTextBoxField textBoxField = new PdfTextBoxField(page, "FirstName");
    textBoxField.Bounds = new RectangleF(0, 0, 100, 20);
    textBoxField.ToolTip = "First Name";
    //Add the form field to the document.
    document.Form.Fields.Add(textBoxField);
    //stream.Position = 0;
    var stream = new MemoryStream();
    //Save the document.
    document.Save(stream);
    bytes = stream.ToArray();
    string base64string = Convert.ToBase64String(bytes);
}
}

```

```
documentPath = "data:application/pdf;base64," + base64string;
//close the document
document.Close(true);
}
}
```

Find the sample [How to view the created PDF document](#)

Resolve "Unexpected token T in JSON at position 0" error

We suspect that the "Unexpected token T in JSON at position 0" error occurs in the Linux platform due to the missing of the Pdfium.dll in your environment. We have embedded the Pdfium rendering engine in our PDF Viewer, so the Pdfium.dll will be generated on runtime within your project location.

However, we have exposed the **ReferencePath** API to set the pdfium library location path. We can place the pdfium library in project location and refer the project location to the ReferencePath. Please find the below code to set the pdfium location inside the wwwroot folder.

CSHARP

```
PdfRenderer.ReferencePath = _hostingEnvironment.WebRootPath + "\\\";
```

The following code example shows how to resolve the "Unexpected token T in JSON at position 0" error in the Linux platform

CSHARP

```
public IActionResult Load([FromBody] Dictionary<string, string> jsonObject)
{
    PdfRenderer pdfviewer = new PdfRenderer(_cache);
    PdfRenderer.ReferencePath = _hostingEnvironment.WebRootPath + "\\\";
    MemoryStream stream = new MemoryStream();
    object jsonResult = new object();
    if (jsonObject != null && jsonObject.ContainsKey("document"))
    {
        if (bool.Parse(jsonObject["isFileName"]))
        {
            string documentPath = GetDocumentPath(jsonObject["document"]);
            if (!string.IsNullOrEmpty(documentPath))
            {
                byte[] bytes = System.IO.File.ReadAllBytes(documentPath);
                stream = new MemoryStream(bytes);
            }
            else
            {
                return this.Content(jsonObject["document"] + " is not found");
            }
        }
        else
        {
            byte[] bytes = Convert.FromBase64String(jsonObject["document"]);
            stream = new MemoryStream(bytes);
        }
    }
    jsonResult = pdfviewer.Load(stream, jsonObject);
    return Content(JsonConvert.SerializeObject(jsonResult));
}
```

```
}

```

Download the [Pdfium.dll](#).

Note: Kindly use the `Syncfusion.EJ2.PdfViewer.AspNet.Core.Linux` package in your application for Linux environment. Also, ensure whether the library dependencies of lib pdfium. so they are installed properly. If not, please execute the following command to install the RUN apt-get update\&& apt-get install -y --allow-unauthenticated \ libc6-dev \ libgdipplus \ libx11-dev \ curl \ vim \ supervisor \ pro cps

Pivot Table

```
<!-- markdownlint-disable MD024 -->

```

```
<!-- markdownlint-disable MD012 -->

```

```
<!-- markdownlint-disable MD029 -->

```

```
<!-- markdownlint-disable MD009 -->

```

Getting Started with Blazor Pivot Table Component

This section briefly explains about how to include a **Pivot Table** in your Blazor Server-Side application. You can refer [Getting Started with Syncfusion Blazor for Server-Side in Visual Studio](#) page for the introduction and configuring the common specifications.

To get start quickly with Blazor Pivot Table component, you can check out this video.

```
{% youtube

```

```
"youtube:https://www.youtube.com/watch?v=1roaubxylGc"%}

```

Importing Syncfusion Blazor Pivot Table component in the application

1. Install the **Syncfusion.Blazor.PivotTable** NuGet package to the application by using the **NuGet Package Manager**.
2. You can add the client-side resources through [CDN](#) or from [NuGet](#) package in the element of the `~/Pages/_Host.cshtml` page.

HTML

```
<head>
<link href="_content/Syncfusion.Blazor.Themes/bootstrap4.css"
rel="stylesheet" />
<!--CDN-->
@*<link href="https://cdn.syncfusion.com/blazor/{{ site.blazorversion
}}/styles/bootstrap4.css" rel="stylesheet" />*@
<!--Additionally refer the below file to use pivot chart-->
<script
src="https://cdnjs.cloudflare.com/ajax/libs/lodash.js/4.17.20/lodash.min.js"
></script>
</head>

```

For Internet Explorer 11 kindly refer the polyfills. Refer the [documentation](#) for more information.

HTML

```
<head>
<link href="https://cdn.syncfusion.com/blazor/{{ site.blazorversion
}}/styles/bootstrap4.css" rel="stylesheet" />
<script
src="https://github.com/Daddoon/Blazor.Polyfill/releases/download/3.0.1/blaz
or.polyfill.min.js"></script>
<!--Additionally refer the below file to use pivot chart-->
<script
src="https://cdnjs.cloudflare.com/ajax/libs/lodash.js/4.17.20/lodash.min.js"
></script>
</head>
```

Adding component package to the application

Open `~/_Imports.razor` file and import the **Syncfusion.Blazor.PivotView** package.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
```

Add SyncfusionBlazor service in Startup.cs

Open the **Startup.cs** file and add services required by Syncfusion components using **services.AddSyncfusionBlazor()** method. Add this method in the **ConfigureServices** function as follows.

CSHARP

```
using Syncfusion.Blazor;
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void ConfigureServices(IServiceCollection services)
        {
            ....
            ....
            services.AddSyncfusionBlazor();
        }
    }
}
```

The server will be disconnected while performing any UI operations if the bound data source is large. To avoid this, add the following service to increase the buffer size of the message over the SignalR connection.

CSHARP

```
using Syncfusion.Blazor;
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void ConfigureServices(IServiceCollection services)
```

```
{
    ....
    services.AddSignalR(e => {
        e.MaximumReceiveMessageSize = 102400000;
    });
    services.AddSyncfusionBlazor();
}
}
```

To enable custom client side resource loading from CRG or CDN. You need to disable resource loading by `AddSyncfusionBlazor(true)` and load the scripts in the HEAD element of the `~/Pages/_Host.cshtml` page.

HTML

```
<head>
<script src="https://cdn.syncfusion.com/blazor/{{ site.blazorversion
}}/syncfusion-blazor.min.js"></script>
</head>
```

Initializing pivot table component in an application

The Syncfusion Pivot Table component can be initialized in any razor page inside `~/Pages` folder. Here, the pivot table component is initialized inside `~/Pages/Index.razor` page. In a new application, if `Index.razor` page has any default content template, then those content can be completely removed and following code can be added.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails"></SfPivotView>
```

Assigning sample data to the pivot table

The [Blazor Pivot Table](#) component further needs to be populated with an appropriate data source. For illustration purpose, a collection of objects mentioning the sales details of certain products over a period and region has been prepared. This sample data is assigned to the pivot table component through [DataSource](#) property under [PivotViewDataSourceSettings](#) class.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
<PivotViewDataSourceSettings DataSource="@dataSource">
</PivotViewDataSourceSettings>
</SfPivotView>
@code{
    public List<ProductDetails> dataSource { get; set; }
    protected override void OnInitialized()
    {
        this.dataSource = ProductDetails.GetProductData().ToList();
    }
    public class ProductDetails
    {
```

```

public int Sold { get; set; }
public double Amount { get; set; }
public string Country { get; set; }
public string Products { get; set; }
public string Year { get; set; }
public string Quarter { get; set; }
public static List<ProductDetails> GetProductData()
{
    List<ProductDetails> productData = new List<ProductDetails>();
    productData.Add(new ProductDetails { Sold = 31, Amount = 52824, Country =
    "France", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q1" });
    productData.Add(new ProductDetails { Sold = 51, Amount = 86904, Country =
    "France", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q2" });
    productData.Add(new ProductDetails { Sold = 90, Amount = 153360, Country =
    "France", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q3" });
    productData.Add(new ProductDetails { Sold = 25, Amount = 42600, Country =
    "France", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q4" });
    productData.Add(new ProductDetails { Sold = 27, Amount = 46008, Country =
    "France", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q1" });
    productData.Add(new ProductDetails { Sold = 49, Amount = 83496, Country =
    "France", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q2" });
    productData.Add(new ProductDetails { Sold = 95, Amount = 161880, Country =
    "France", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q3" });
    productData.Add(new ProductDetails { Sold = 67, Amount = 114168, Country =
    "France", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q4" });
    productData.Add(new ProductDetails { Sold = 75, Amount = 127800, Country =
    "France", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q1" });
    productData.Add(new ProductDetails { Sold = 67, Amount = 114168, Country =
    "France", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q2" });
    productData.Add(new ProductDetails { Sold = 69, Amount = 117576, Country =
    "France", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q3" });
    productData.Add(new ProductDetails { Sold = 90, Amount = 153360, Country =
    "France", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q4" });
    productData.Add(new ProductDetails { Sold = 16, Amount = 27264, Country =
    "France", Products = "Mountain Bikes", Year = "FY 2018", Quarter = "Q1" });
    productData.Add(new ProductDetails { Sold = 83, Amount = 124422, Country =
    "France", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q1" });
    productData.Add(new ProductDetails { Sold = 57, Amount = 85448, Country =
    "France", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q2" });
    productData.Add(new ProductDetails { Sold = 20, Amount = 29985, Country =
    "France", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q3" });
    productData.Add(new ProductDetails { Sold = 67, Amount = 70008, Country =
    "France", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q1" });
    productData.Add(new ProductDetails { Sold = 89, Amount = 60496, Country =
    "France", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q2" });
    productData.Add(new ProductDetails { Sold = 75, Amount = 801880, Country =
    "France", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q3" });
    productData.Add(new ProductDetails { Sold = 57, Amount = 204168, Country =
    "France", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q4" });
    productData.Add(new ProductDetails { Sold = 75, Amount = 737800, Country =
    "France", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q1" });
    productData.Add(new ProductDetails { Sold = 87, Amount = 884168, Country =
    "France", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q2" });
    productData.Add(new ProductDetails { Sold = 39, Amount = 729576, Country =
    "France", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q3" });
    productData.Add(new ProductDetails { Sold = 90, Amount = 38860, Country =
    "France", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q4" });
}

```



```

productData.Add(new ProductDetails { Sold = 93, Amount = 139412, Country =
"France", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q4" });
productData.Add(new ProductDetails { Sold = 51, Amount = 92824, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q1" });
productData.Add(new ProductDetails { Sold = 61, Amount = 76904, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q2" });
productData.Add(new ProductDetails { Sold = 70, Amount = 43360, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q3" });
productData.Add(new ProductDetails { Sold = 85, Amount = 62600, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q4" });
productData.Add(new ProductDetails { Sold = 97, Amount = 86008, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q1" });
productData.Add(new ProductDetails { Sold = 69, Amount = 93496, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q2" });
productData.Add(new ProductDetails { Sold = 45, Amount = 301880, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q3" });
productData.Add(new ProductDetails { Sold = 77, Amount = 404168, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q4" });
productData.Add(new ProductDetails { Sold = 15, Amount = 137800, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q1" });
productData.Add(new ProductDetails { Sold = 37, Amount = 184168, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q2" });
productData.Add(new ProductDetails { Sold = 49, Amount = 89576, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q3" });
productData.Add(new ProductDetails { Sold = 40, Amount = 33360, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q4" });
productData.Add(new ProductDetails { Sold = 96, Amount = 77264, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2018", Quarter = "Q1" });
productData.Add(new ProductDetails { Sold = 23, Amount = 24422, Country =
"Germany", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q1" });
productData.Add(new ProductDetails { Sold = 67, Amount = 75448, Country =
"Germany", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q2" });
productData.Add(new ProductDetails { Sold = 70, Amount = 52345, Country =
"Germany", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q3" });
productData.Add(new ProductDetails { Sold = 13, Amount = 135612, Country =
"Germany", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q4" });
productData.Add(new ProductDetails { Sold = 57, Amount = 90008, Country =
"Germany", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q1" });
productData.Add(new ProductDetails { Sold = 29, Amount = 90496, Country =
"Germany", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q2" });
productData.Add(new ProductDetails { Sold = 45, Amount = 301880, Country =
"Germany", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q3" });
productData.Add(new ProductDetails { Sold = 77, Amount = 404168, Country =
"Germany", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q4" });
productData.Add(new ProductDetails { Sold = 15, Amount = 137800, Country =
"Germany", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q1" });
productData.Add(new ProductDetails { Sold = 37, Amount = 184168, Country =
"Germany", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q2" });
productData.Add(new ProductDetails { Sold = 99, Amount = 829576, Country =
"Germany", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q3" });
productData.Add(new ProductDetails { Sold = 80, Amount = 38360, Country =
"Germany", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q4" });
productData.Add(new ProductDetails { Sold = 91, Amount = 67824, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2015", Quarter =
"Q1" });

```

```
productData.Add(new ProductDetails { Sold = 81, Amount = 99904, Country =  
"United States", Products = "Mountain Bikes", Year = "FY 2015", Quarter =  
"Q2" });  
productData.Add(new ProductDetails { Sold = 70, Amount = 49360, Country =  
"United States", Products = "Mountain Bikes", Year = "FY 2015", Quarter =  
"Q3" });  
productData.Add(new ProductDetails { Sold = 65, Amount = 69600, Country =  
"United States", Products = "Mountain Bikes", Year = "FY 2015", Quarter =  
"Q4" });  
productData.Add(new ProductDetails { Sold = 57, Amount = 90008, Country =  
"United States", Products = "Mountain Bikes", Year = "FY 2016", Quarter =  
"Q1" });  
productData.Add(new ProductDetails { Sold = 29, Amount = 90496, Country =  
"United States", Products = "Mountain Bikes", Year = "FY 2016", Quarter =  
"Q2" });  
productData.Add(new ProductDetails { Sold = 85, Amount = 391880, Country =  
"United States", Products = "Mountain Bikes", Year = "FY 2016", Quarter =  
"Q3" });  
productData.Add(new ProductDetails { Sold = 97, Amount = 904168, Country =  
"United States", Products = "Mountain Bikes", Year = "FY 2016", Quarter =  
"Q4" });  
productData.Add(new ProductDetails { Sold = 85, Amount = 237800, Country =  
"United States", Products = "Mountain Bikes", Year = "FY 2017", Quarter =  
"Q1" });  
productData.Add(new ProductDetails { Sold = 77, Amount = 384168, Country =  
"United States", Products = "Mountain Bikes", Year = "FY 2017", Quarter =  
"Q2" });  
productData.Add(new ProductDetails { Sold = 99, Amount = 829576, Country =  
"United States", Products = "Mountain Bikes", Year = "FY 2017", Quarter =  
"Q3" });  
productData.Add(new ProductDetails { Sold = 80, Amount = 38360, Country =  
"United States", Products = "Mountain Bikes", Year = "FY 2017", Quarter =  
"Q4" });  
productData.Add(new ProductDetails { Sold = 76, Amount = 97264, Country =  
"United States", Products = "Mountain Bikes", Year = "FY 2018", Quarter =  
"Q1" });  
productData.Add(new ProductDetails { Sold = 53, Amount = 94422, Country =  
"United States", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q1"  
});  
productData.Add(new ProductDetails { Sold = 90, Amount = 45448, Country =  
"United States", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q2"  
});  
productData.Add(new ProductDetails { Sold = 29, Amount = 92345, Country =  
"United States", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q3"  
});  
productData.Add(new ProductDetails { Sold = 67, Amount = 235612, Country =  
"United States", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q4"  
});  
productData.Add(new ProductDetails { Sold = 97, Amount = 90008, Country =  
"United States", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q1"  
});  
productData.Add(new ProductDetails { Sold = 79, Amount = 90496, Country =  
"United States", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q2"  
});  
productData.Add(new ProductDetails { Sold = 95, Amount = 501880, Country =  
"United States", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q3"  
});
```

```

productData.Add(new ProductDetails { Sold = 97, Amount = 104168, Country =
"United States", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q4"
});
productData.Add(new ProductDetails { Sold = 95, Amount = 837800, Country =
"United States", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q1"
});
productData.Add(new ProductDetails { Sold = 87, Amount = 684168, Country =
"United States", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q2"
});
productData.Add(new ProductDetails { Sold = 109, Amount = 29576, Country =
"United States", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q3"
});
productData.Add(new ProductDetails { Sold = 99, Amount = 345860, Country =
"United States", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q4"
});
return productData;
}
}
}

```

Adding fields to row, column, value and filter axes

Now that pivot table is initialized and assigned with sample data, will further move to showcase the component by organizing appropriate fields in row, column, value and filter axes.

In [PivotViewDataSourceSettings](#) class, four major axes - [PivotViewRows](#), [PivotViewColumns](#), [PivotViewValues](#) and [PivotViewFilters](#) plays a vital role in defining and organizing fields from the bound data source, to render the entire pivot table component in a desired format.

[PivotViewRows](#) – Collection of fields that needs to be displayed in row axis of the pivot table.

[PivotViewColumns](#) – Collection of fields that needs to be displayed in column axis of the pivot table.

[PivotViewValues](#) – Collection of fields that needs to be displayed as aggregated numeric values in the pivot table.

[PivotViewFilters](#) - Collection of fields that would act as master filter over the data bound in row, column and value axes of the pivot table.

In-order to define each field in the respective axis, the following basic properties should be set.

- [Name](#): It allows to set the field name from the bound data source. It's casing should match exactly like in the data source and if not set properly, the pivot table will not be rendered.
- [Caption](#): It allows to set the field caption, which is the alias name of the field that needs to be displayed in the pivot table.
- [Type](#): It allows to set the summary type of the field. By default, [SummaryTypes.Sum](#) is applied.

In this illustration, "Year" and "Quarter" are added in column, "Country" and "Products" in row, and "Sold" and "Amount" in value section respectively.

ASPX-CS

```

@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails" Height="300">
  <PivotViewDataSourceSettings DataSource="@dataSource">
    <PivotViewColumns>

```

```

<PivotViewColumn Name="Year"></PivotViewColumn>
<PivotViewColumn Name="Quarter"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Country"></PivotViewRow>
<PivotViewRow Name="Products"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Units Sold"></PivotViewValue>
<PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
</PivotViewValues>
</PivotViewDataSourceSettings>
</SfPivotView>
@code{
public List<ProductDetails> dataSource { get; set; }
protected override void OnInitialized()
{
this.dataSource = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section for more details.
}
}

```

Applying formatting to a value field

Formatting defines a way in which values should be displayed. For example, format “C” denotes the values should be displayed in currency pattern. To do so, define the [PivotViewFormatSetting](#) class with its [Name](#) and [Format](#) properties and add it to [PivotViewFormatSettings](#). In this illustration, the [Name](#) property is set as **Amount**, a field from value section and its format is set as currency. Likewise, we can set format for other value fields as well and add it to [PivotViewFormatSettings](#).

Only fields from value section, which is in the form of numeric data values are applicable for formatting.

ASPX-CS

```

@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails" Height="300">
<PivotViewDataSourceSettings DataSource="@dataSource">
<PivotViewColumns>
<PivotViewColumn Name="Year"></PivotViewColumn>
<PivotViewColumn Name="Quarter"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Country"></PivotViewRow>
<PivotViewRow Name="Products"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Units Sold"></PivotViewValue>
<PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
</PivotViewFormatSettings>
</PivotViewDataSourceSettings>
</SfPivotView>
@code{
public List<ProductDetails> dataSource { get; set; }
}

```

```
protected override void OnInitialized()
{
    this.dataSource = ProductDetails.GetProductData().ToList();
    //Bind the data source collection here. Refer "Assigning sample data to the
    pivot table" section for more details.
}
}
```

After successful compilation of the application, simply press F5 to run the same. The pivot table component will render in the default web browser like below.

	FY 2015		FY 2016		FY 2017
	Units Sold	Sold Amount	Units Sold	Sold Amount	Units Sold
France	450	\$714,955	526	\$1,542,104	
Germany	440	\$563,515	496	\$1,772,104	
United States	546	\$754,515	636	\$2,263,104	
Grand Total	1436	\$2,032,985	1658	\$5,577,312	1

Enable Field List

The field list allows to add or remove fields and also rearrange the fields between different axes, including column, row, value, and filter along with filter and sort options dynamically at runtime. It can be enabled by setting the [ShowFieldList](#) property to **true**. To know more about field list, [refer](#) here.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails" ShowFieldList="true" Height="300">
<PivotViewDataSourceSettings DataSource="@dataSource">
<PivotViewColumns>
<PivotViewColumn Name="Year"></PivotViewColumn>
<PivotViewColumn Name="Quarter"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Country"></PivotViewRow>
<PivotViewRow Name="Products"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Units Sold"></PivotViewValue>
<PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
</PivotViewFormatSettings>
</PivotViewDataSourceSettings>
</SfPivotView>
@code{
public List<ProductDetails> dataSource { get; set; }
protected override void OnInitialized()
{
    this.dataSource = ProductDetails.GetProductData().ToList();
}
```

```
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section for more details.
}
}
```

	FY 2015		FY 2016		FY 2017
	Units Sold	Sold Amount	Units Sold	Sold Amount	Units Sold
France	450	\$714,955	526	\$1,542,104	
Germany	440	\$563,515	496	\$1,772,104	
United States	546	\$754,515	636	\$2,263,104	
Grand Total	1436	\$2,032,985	1658	\$5,577,312	1

Field List

All Fields

☒ Quarter
 ☒ Year
 ☒ Products
 ☒ Country
 ☒ Sold Amount
 ☒ Units Sold

Filters

Drop filter here

Columns

Year

Quarter

Rows

Country

Products

Values

Sum of Units Sold

Sum of Sold Amount

Close

Enable Grouping Bar

The grouping bar feature automatically populates fields from the bound data source and allows end users to drag fields between different axes such as columns, rows, values, and filters, and alter pivot table at runtime. It also provides option to sort, filter and remove fields. It can be enabled by setting the [ShowGroupingBar](#) property to **true**. To know more about grouping bar, [refer](#) here.

ASPX-CS

```

@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails" ShowGroupingBar="true" Height="300">
  <PivotViewDataSourceSettings DataSource="@dataSource">
    <PivotViewColumns>
      <PivotViewColumn Name="Year"></PivotViewColumn>
      <PivotViewColumn Name="Quarter"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Country"></PivotViewRow>
      <PivotViewRow Name="Products"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Sold" Caption="Units Sold"></PivotViewValue>
      <PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
    </PivotViewValues>
    <PivotViewFormatSettings>
      <PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
    </PivotViewFormatSettings>
  </PivotViewDataSourceSettings>
</SfPivotView>
@code{
public List<ProductDetails> dataSource { get; set; }
protected override void OnInitialized()
{
  this.dataSource = ProductDetails.GetProductData().ToList();
  //Bind the data source collection here. Refer "Assigning sample data to the
  pivot table" section for more details.
}
}

```

Sum of Units Sold ▾ ×	Drop filter here				
Sum of Sold Amount ▾ ×	Year ↑ ▾ ×		Quarter ↑ ▾ ×		
Country ↑ ▾ ×	FY 2015		FY 2016		▶
Products ↑ ▾ ×	Units Sold	Sold Amount	Units Sold	Sold Amount	Un
▶ France	450	\$714,955	526	\$1,542,104	
▶ Germany	440	\$563,515	496	\$1,772,104	
▶ United States	546	\$754,515	636	\$2,263,104	
Grand Total	1436	\$2,032,985	1658	\$5,577,312	

Exploring Filter Axis

The filter axis contains collection of fields that would act as master filter over the data bound in row, column and value axes of the pivot table. The fields along with filter members could be set to filter axis either through report via code behind or by dragging and dropping fields from other axes to filter axis via grouping bar or field list at runtime.

ASPX-CS

```

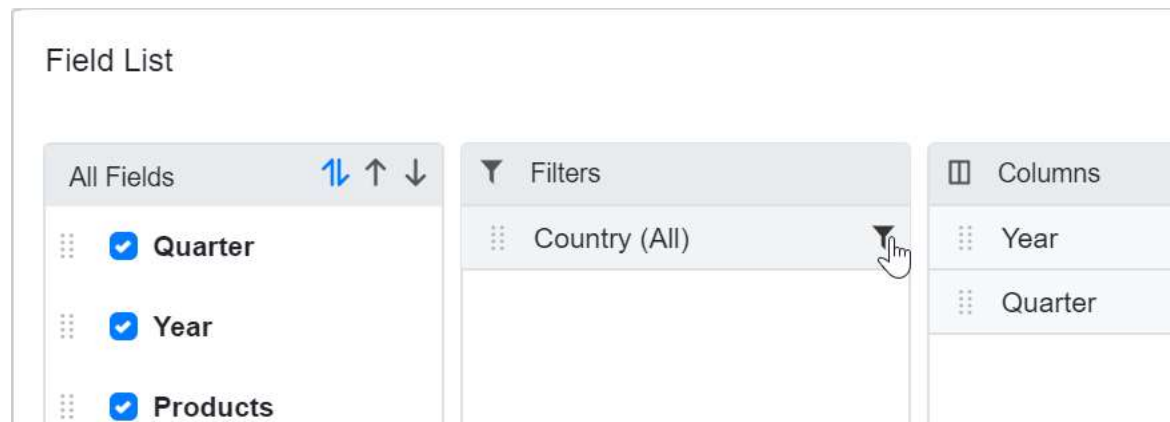
@using Syncfusion.Blazor.PivotView

```

```

<SfPivotView TValue="ProductDetails" ShowGroupingBar="true"
ShowFieldList="true" Height="300">
  <PivotViewDataSourceSettings DataSource="@dataSource">
    <PivotViewColumns>
      <PivotViewColumn Name="Year"></PivotViewColumn>
      <PivotViewColumn Name="Quarter"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Products"></PivotViewRow>
    </PivotViewRows>
    <PivotViewFilters>
      <PivotViewFilter Name="Country"></PivotViewFilter>
    </PivotViewFilters>
    <PivotViewValues>
      <PivotViewValue Name="Sold" Caption="Units Sold"></PivotViewValue>
      <PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
    </PivotViewValues>
    <PivotViewFormatSettings>
      <PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
    </PivotViewFormatSettings>
  </PivotViewDataSourceSettings>
</SfPivotView>
@code{
public List<ProductDetails> dataSource { get; set; }
protected override void OnInitialized()
{
  this.dataSource = ProductDetails.GetProductData().ToList();
  //Bind the data source collection here. Refer "Assigning sample data to the
  pivot table" section for more details.
}
}

```



Sum of Units Sold ▾ ×	Country (All) ▾ ×		
Sum of Sold Amount ▾ ×	Year ↑ ▾ × Quarter ↑ ▾ ×		
Products ↑ ▾ ×	FY 2015		FY 2016
	Units Sold	Sold Amount	Units Sold
Mountain Bikes	771	\$898,064	794

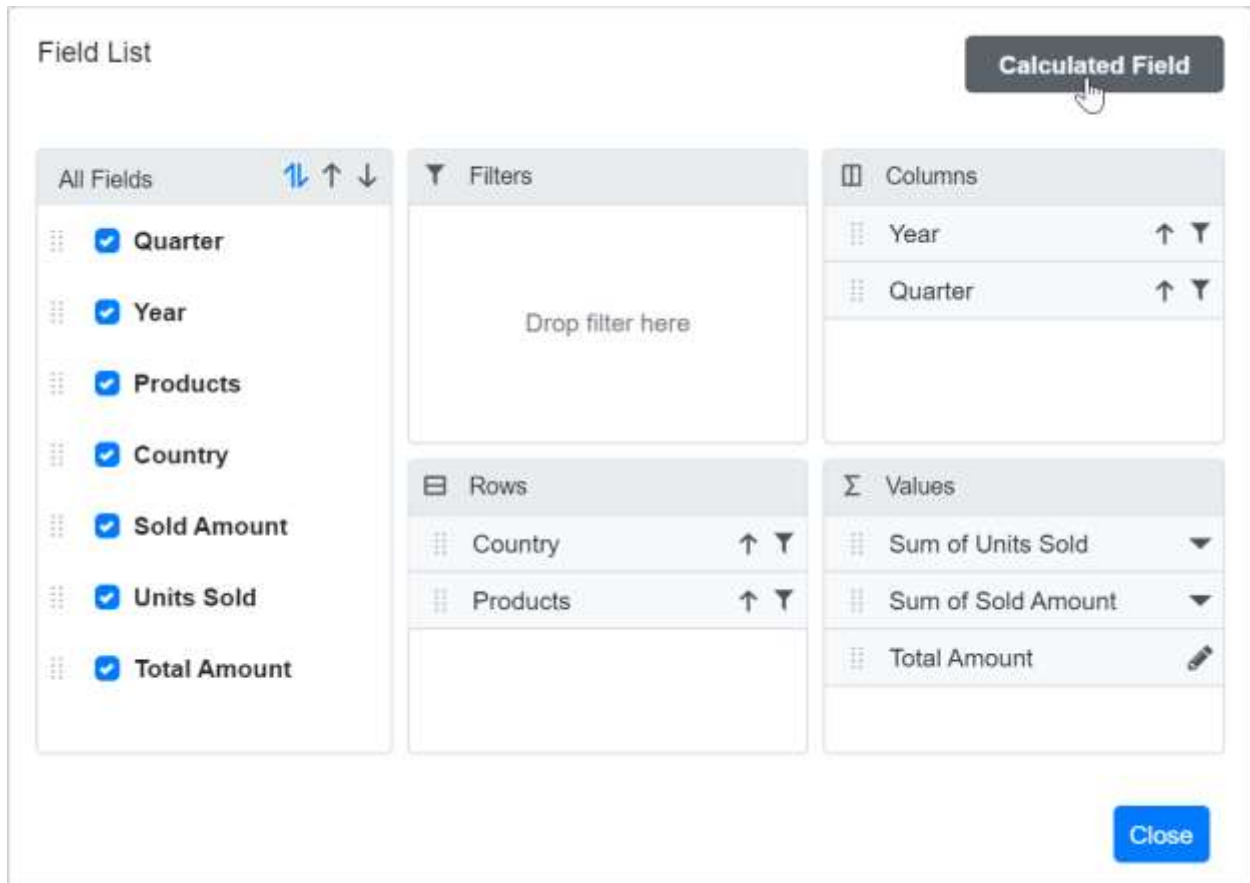
Calculated Field

The calculated field feature allows user to insert or add a new calculated field based on the available fields from the bound data source using basic arithmetic operators. The calculated field can be included in pivot table using the [PivotViewCalculatedFieldSetting](#) class from code behind. Or else, calculated fields can be added at run time through the built-in dialog by just setting the [AllowCalculatedField](#) property to **true** in pivot table. You will see a button enabled in the Field List UI automatically to invoke the calculated field dialog and perform necessary operation. To know more about calculated field, [refer here](#).

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails" ShowFieldList="true"
AllowCalculatedField="true" Height="300">
<PivotViewDataSourceSettings DataSource="@dataSource">
<PivotViewColumns>
<PivotViewColumn Name="Year"></PivotViewColumn>
<PivotViewColumn Name="Quarter"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Country"></PivotViewRow>
<PivotViewRow Name="Products"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
<PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
<PivotViewValue Name="Total" Caption="Total Amount"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
<PivotViewFormatSetting Name="Total" Format="C"></PivotViewFormatSetting>
</PivotViewFormatSettings>
<PivotViewCalculatedFieldSettings>
<PivotViewCalculatedFieldSetting Name="Total"
Formula="@totalPrice"></PivotViewCalculatedFieldSetting>
</PivotViewCalculatedFieldSettings>
</PivotViewDataSourceSettings>
</SfPivotView>
@code{
public string totalPrice = "\" + "Sum(Amount)" + "\" + "+" + "\" + "Sum(Sold)" + "\"";
public List<ProductDetails> dataSource { get; set; }
protected override void OnInitialized()
{
this.dataSource = ProductDetails.GetProductData().ToList();
}
```

```
//Bind the data source collection here. Refer "Assigning sample data to the  
pivot table" section for more details.  
}  
}
```



Create Calculated Field ×

Total Amount

Drag and drop fields to formula

Total Amount (Calculated F...

Units Sold (Sum) ▶

Year (Count) ▶

Formula

"Sum(Amount)"+"Sum(Sold)"

Format

C0

OK

Cancel

	FY 2015			FY 2016	
	Units Sold	Sold Amount	Total Amount	Units Sold	Sold Amou
France	450	\$714,955	\$715,405	526	\$1,542,
Germany	440	\$563,515	\$563,955	496	\$1,772,
United States	546	\$754,515	\$755,061	636	\$2,263,
Grand Total	1436	\$2,032,985	\$2,034,421	1658	\$5,577,

You can also explore our [Blazor Pivot Table example](#) to know how to render and configure the pivot table.

See Also

- [Getting Started with Syncfusion Blazor for Client-Side in .NET Core CLI](#)
- [Getting Started with Syncfusion Blazor for Client-side in Visual Studio 2019](#)

- [Getting Started with Syncfusion Blazor for Server-Side in .NET Core CLI](#)

Data Binding in Blazor Pivot Table Component

JSON

For JSON data binding, the `type` property under [PivotViewDataSourceSettings](#) needs to be set as `JSON`. The default value is assumed as `JSON`.

Binding JSON data via local

In-order to bind local JSON data to the pivot table, user can assign the local variable holding the JSON data to the [DataSource](#) property under [PivotViewDataSourceSettings](#).

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
  <PivotViewDataSourceSettings DataSource="@data">
    <PivotViewColumns>
      <PivotViewColumn Name="Year"></PivotViewColumn>
      <PivotViewColumn Name="Quarter"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Country"></PivotViewRow>
      <PivotViewRow Name="Products"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
      <PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
    </PivotViewValues>
    <PivotViewFormatSettings>
      <PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
    </PivotViewFormatSettings>
  </PivotViewDataSourceSettings>
</SfPivotView>

@code{
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
  this.data = ProductDetails.GetProductData().ToList();
  //Bind the data source collection here. Refer "Assigning sample data to the
  pivot table" section in getting started for more details.
}
}
```

	FY 2015		FY 2016		FY 2017
	Units Sold	Sold Amount	Units Sold	Sold Amount	Units Sold
France	450	\$714,955	526	\$1,542,104	
Germany	440	\$563,515	496	\$1,772,104	
United States	546	\$754,515	636	\$2,263,104	
Grand Total	1436	\$2,032,985	1658	\$5,577,312	

Using local variable, the JSON data can also be bound to the pivot table using [SfDataManager](#) option with the help of [JsonAdaptor](#). Here the instance of [SfDataManager](#) holding JSON data is assigned to [DataSource](#) property under [PivotViewDataSourceSettings](#). The use of [SfDataManager](#) is optional here.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
@using Syncfusion.Blazor.Data
<SfPivotView TValue="ProductDetails" Width="1500" Height="300">
  <PivotViewDataSourceSettings TValue="ProductDetails" ExpandAll=false
  EnableSorting=true>
    <SfDataManager Json="@dataSource"
    Adaptor="Syncfusion.Blazor.Adaptors.JsonAdaptor"></SfDataManager>
    <PivotViewColumns>
      <PivotViewColumn Name="Year"></PivotViewColumn>
      <PivotViewColumn Name="Quarter"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Country"></PivotViewRow>
      <PivotViewRow Name="Products"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
      <PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
    </PivotViewValues>
    <PivotViewFormatSettings>
      <PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
    </PivotViewFormatSettings>
    </PivotViewDataSourceSettings>
    <PivotViewGridSettings ColumnWidth="120"></PivotViewGridSettings>
  </SfPivotView>
@code{
  public List<ProductDetails> data { get; set; }
  protected override void OnInitialized()
  {
    this.data = ProductDetails.GetProductData().ToList();
    //Bind the data source collection here. Refer "Assigning sample data to the
    pivot table" section in getting started for more details.
  }
}
```

	FY 2015		FY 2016		FY 2017
	Units Sold	Sold Amount	Units Sold	Sold Amount	Units Sold
France	450	\$714,955	526	\$1,542,104	
Germany	440	\$563,515	496	\$1,772,104	
United States	546	\$754,515	636	\$2,263,104	
Grand Total	1436	\$2,032,985	1658	\$5,577,312	

In the meantime, the JSON data from the local *.json file type can also be connected to the pivot table. Here, the file can be read by the **StreamReader** option, which will give the result in the string form. And

the resulting string needs to be converted to JSON data that can be assigned to the [DataSource](#) property under [PivotViewDataSourceSettings](#).

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
@using System;
@using System.Collections.Generic;
@using System.IO;
@using System.Net;

<SfPivotView TValue="PivotProductDetails" Width="1500" Height="300"
ShowFieldList="true">
  <PivotViewDataSourceSettings TValue="PivotProductDetails"
DataSource="@dataSource" ExpandAll=false EnableSorting=true>
    <PivotViewColumns>
      <PivotViewColumn Name="EnerType" Caption="Energy Type"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="EneSource" Caption="Energy Source"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="PowUnits" Caption="Units (GWh)"></PivotViewValue>
      <PivotViewValue Name="ProCost" Caption="Cost (MM)"></PivotViewValue>
    </PivotViewValues>
    <PivotViewFormatSettings>
      <PivotViewFormatSetting Name="PowUnits" Format="N0"
UseGrouping=true></PivotViewFormatSetting>
      <PivotViewFormatSetting Name="ProCost" Format="C0"
UseGrouping=true></PivotViewFormatSetting>
    </PivotViewFormatSettings>
  </PivotViewDataSourceSettings>
  <PivotViewGridSettings ColumnWidth="120"></PivotViewGridSettings>
</SfPivotView>

@code{
public List<PivotProductDetails> dataSource { get; set; }
protected override void OnInitialized()
{
    // Put appropriate file path here
    string url = AppDomain.CurrentDomain.BaseDirectory + "sales-analysis.json";
    WebClient myWebClient = new WebClient();
    Stream myStream = myWebClient.OpenRead(url);
    StreamReader stream = new StreamReader(myStream);
    string result = stream.ReadToEnd();
    stream.Close();
    this.dataSource =
Newtonsoft.Json.JsonConvert.DeserializeObject<List<PivotProductDetails>>(res
ult);
}
public class PivotProductDetails
{
    public string Date { get; set; }
    public string Sector { get; set; }
    public string EnerType { get; set; }
    public string EneSource { get; set; }
    public int PowUnits { get; set; }
    public int ProCost { get; set; }
}
```

```
}

```

	Biomass		Fire Energy		Grand Total	
	Units (GWh)	Cost (MM)	Units (GWh)	Cost (MM)	Units (GWh)	Cost (MM)
Bio-diesel	1,042	\$1,429			1,042	\$1,429
Ethanol Fuel	595	\$1,031			595	\$1,031
Geo-thermal			1,528	\$2,115	1,528	\$2,115
Hydro-electric			3,378	\$3,244	3,378	\$3,244
Solar			7,509	\$6,210	7,509	\$6,210
Waste-to-energy	712	\$1,043			712	\$1,043
Wind			15,571	\$7,695	15,571	\$7,695
Wood	228	\$349			228	\$349

Binding JSON data via remote

In-order to bind remote JSON data, mention the endpoint [Url](#) under [PivotViewDataSourceSettings](#) property. The [Url](#) property supports both direct downloadable file (*.json) and web service URL.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="PivotProductDetails" Width="1500" Height="300"
ShowFieldList="true">
<PivotViewDataSourceSettings TValue="PivotProductDetails" ExpandAll=false
EnableSorting=true Url="https://cdn.syncfusion.com/data/sales-analysis.json"
Type=DataSourceType.JSON>
<PivotViewColumns>
<PivotViewColumn Name="EnerType" Caption="Energy Type"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="EneSource" Caption="Energy Source"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="PowUnits" Caption="Units (GWh)"></PivotViewValue>
<PivotViewValue Name="ProCost" Caption="Cost (MM)"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="PowUnits" Format="N0"
UseGrouping=true></PivotViewFormatSetting>
<PivotViewFormatSetting Name="ProCost" Format="C0"
UseGrouping=true></PivotViewFormatSetting>
</PivotViewFormatSettings>
</PivotViewDataSourceSettings>
<PivotViewGridSettings ColumnWidth="120"></PivotViewGridSettings>
</SfPivotView>
@code{
public class PivotProductDetails
{
public string EnerType { get; set; }
public string EneSource { get; set; }
public double PowUnits { get; set; }
public double ProCost { get; set; }
}
}
```

	Biomass		Fire Energy		Grand Total	
	Units (GWh)	Cost (MM)	Units (GWh)	Cost (MM)	Units (GWh)	Cost (MM)
Bio-diesel	1,042	\$1,428			1,042	\$1,428
Ethanol Fuel	555	\$1,031			555	\$1,031
Geo-thermal			1,528	\$2,115	1,528	\$2,115
Hydro-electric			3,378	\$3,244	3,378	\$3,244
Solar			7,509	\$6,210	7,509	\$6,210
Waste-to-energy	712	\$1,043			712	\$1,043
Wind			15,571	\$7,695	15,571	\$7,695
Wood	228	\$349			228	\$349

CSV

For CSV data binding, the `type` property under [PivotViewDataSourceSettings](#) needs to be set as `CSV` mandatorily.

The CSV format is considered to be the most compact format compared to JSON since it is half the size of JSON. This helps to reduce the bandwidth while transferring to the browser.

Binding CSV data via local

In-order to bind local CSV data to the pivot table, user needs to convert it as string array and then directly assign it to the [DataSource](#) property under [PivotViewDataSourceSettings](#).

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
@using Syncfusion.Blazor.Data
<SfPivotView TValue="string[]" Width="1500" Height="300">
  <PivotViewDataSourceSettings TValue="string[]" DataSource="@dataSource"
  ExpandAll=false EnableSorting=true Type=DataSourceType.CSV>
    <PivotViewColumns>
      <PivotViewColumn Name="Item Type"></PivotViewColumn>
      <PivotViewColumn Name="Sales Channel"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Region"></PivotViewRow>
      <PivotViewRow Name="Country"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Total Cost"></PivotViewValue>
      <PivotViewValue Name="Total Revenue"></PivotViewValue>
      <PivotViewValue Name="Total Profit"></PivotViewValue>
    </PivotViewValues>
    <PivotViewFormatSettings>
      <PivotViewFormatSetting Name="Total Cost" Format="C0"
      UseGrouping=true></PivotViewFormatSetting>
      <PivotViewFormatSetting Name="Total Revenue" Format="C0"
      UseGrouping=true></PivotViewFormatSetting>
      <PivotViewFormatSetting Name="Total Profit" Format="C0"
      UseGrouping=true></PivotViewFormatSetting>
    </PivotViewFormatSettings>
  </PivotViewDataSourceSettings>
  <PivotViewGridSettings ColumnWidth="120"></PivotViewGridSettings>
</SfPivotView>

@code{
public List<string[]> dataSource { get; set; }
protected override void OnInitialized()
{
  string data = "Region,Country,Item Type,Sales Channel,Order Priority,Order
Date,Order ID,Ship Date,Units Sold,Unit Price,Unit Cost,Total Revenue,Total
```



```

Cost,Total Profit\r\nMiddle East and North
Africa,Libya,Cosmetics,Offline,M,10/18/2014,686800706,10/31/2014,8446,437.20
,263.33,3692591.20,2224085.18,1468506.02\r\nNorth
America,Canada,Vegetables,Online,M,11/7/2011,185941302,12/8/2011,3018,154.06
,90.93,464953.08,274426.74,190526.34\r\nMiddle East and North
Africa,Libya,Baby
Food,Offline,C,10/31/2016,246222341,12/9/2016,1517,255.28,159.42,387259.76,2
41840.14,145419.62\r\nAsia,Japan,Cereal,Offline,C,4/10/2010,161442649,5/12/2
010,3322,205.70,117.11,683335.40,389039.42,294295.98\r\nSub-Saharan
Africa,Chad,Fruits,Offline,H,8/16/2011,645713555,8/31/2011,9845,9.33,6.92,91
853.85,68127.40,23726.45\r\nEurope,Armenia,Cereal,Online,H,11/24/2014,683458
888,12/28/2014,9528,205.70,117.11,1959909.60,1115824.08,844085.52\r\nSub-
Saharan
Africa,Eritrea,Cereal,Online,H,3/4/2015,679414975,4/17/2015,2844,205.70,117.
11,585010.80,333060.84,251949.96\r\nEurope,Montenegro,Clothes,Offline,M,5/17
/2012,208630645,6/28/2012,7299,109.28,35.84,797634.72,261596.16,536038.56\r\
nCentral America and the
Caribbean,Jamaica,Vegetables,Online,H,1/29/2015,266467225,3/7/2015,2428,154.
06,90.93,374057.68,220778.04,153279.64\r\nAustralia and
Oceania,Fiji,Vegetables,Offline,H,12/24/2013,118598544,1/19/2014,4800,154.06
,90.93,739488.00,436464.00,303024.00\r\nSub-Saharan
Africa,Togo,Clothes,Online,M,12/29/2015,451010930,1/19/2016,3012,109.28,35.8
4,329151.36,107950.08,221201.28\r\nEurope,Montenegro,Snacks,Offline,M,2/27/2
010,220003211,3/18/2010,2694,152.58,97.44,411050.52,262503.36,148547.16\r\nE
urope,Greece,Household,Online,C,11/17/2016,702186715,12/22/2016,1508,668.27,
502.54,1007751.16,757830.32,249920.84\r\nSub-Saharan
Africa,Sudan,Cosmetics,Online,C,12/20/2015,544485270,1/5/2016,4146,437.20,26
3.33,1812631.20,1091766.18,720865.02\r\nAsia,Maldives,Fruits,Offline,L,1/8/2
011,714135205,2/6/2011,7332,9.33,6.92,68407.56,50737.44,17670.12\r\nEurope,M
ontenegro,Clothes,Offline,H,6/28/2010,448685348,7/22/2010,4820,109.28,35.84,
526729.60,172748.80,353980.80\r\nEurope,Estonia,Office
Supplies,Online,H,4/25/2016,405997025,5/12/2016,2397,651.21,524.96,1560950.3
7,1258329.12,302621.25\r\nNorth
America,Greenland,Beverages,Online,M,7/27/2012,414244067,8/7/2012,2880,47.45
,31.79,136656.00,91555.20,45100.80\r\nSub-Saharan Africa,Cape
Verde,Clothes,Online,C,9/8/2014,821912801,10/3/2014,1117,109.28,35.84,122065
.76,40033.28,82032.48\r\nSub-Saharan
Africa,Senegal,Household,Offline,L,8/27/2012,247802054,9/8/2012,8989,668.27,
502.54,6007079.03,4517332.06,1489746.97\r\nAustralia and Oceania,Federated
States of
Micronesia,Snacks,Online,C,9/3/2012,531023156,10/15/2012,407,152.58,97.44,62
100.06,39658.08,22441.98\r\nEurope,Bulgaria,Clothes,Online,L,8/27/2010,88099
9934,9/16/2010,6313,109.28,35.84,689884.64,226257.92,463626.72\r\nMiddle
East and North Africa,Algeria,Personal
Care,Online,H,2/20/2011,127468717,3/9/2011,9681,81.73,56.67,791228.13,548622
.27,242605.86\r\nAsia,Mongolia,Clothes,Online,L,12/12/2015,770478332,1/24/20
16,515,109.28,35.84,56279.20,18457.60,37821.60\r\nCentral America and the
Caribbean,Grenada,Cereal,Online,H,10/28/2012,430390107,11/13/2012,852,205.70
,117.11,175256.40,99777.72,75478.68\r\nCentral America and the
Caribbean,Grenada,Beverages,Online,M,1/30/2017,397877871,3/20/2017,9759,47.4
5,31.79,463064.55,310238.61,152825.94\r\nSub-Saharan
Africa,Senegal,Beverages,Offline,M,10/22/2014,683927953,11/4/2014,8334,47.45
,31.79,395448.30,264937.86,130510.44\r\nNorth
America,Greenland,Fruits,Offline,M,1/31/2012,469839179,2/22/2012,4709,9.33,6
.92,43934.97,32586.28,11348.69\r\nSub-Saharan
Africa,Chad,Meat,Offline,H,1/20/2016,357222878,3/9/2016,9043,421.89,364.69,3
815151.27,3297891.67,517259.60\r\nSub-Saharan Africa,Mauritius ,Personal

```

Care,Online,C,1/1/2016,118002879,1/7/2016,8529,81.73,56.67,697075.17,483338.43,213736.74\r\nMiddle East and North
Africa,Morocco,Beverages,Offline,C,6/1/2017,944415509,6/23/2017,2391,47.45,31.79,113452.95,76009.89,37443.06\r\nCentral America and the
Caribbean,Honduras,Office
Supplies,Online,H,6/30/2015,499009597,7/9/2015,6884,651.21,524.96,4482929.64,3613824.64,869105.00\r\nSub-Saharan
Africa,Benin,Fruits,Online,L,1/28/2014,564646470,3/16/2014,293,9.33,6.92,2733.69,2027.56,706.13\r\nEurope,Greece,Baby
Food,Offline,M,4/8/2014,294499957,4/8/2014,7937,255.28,159.42,2026157.36,1265316.54,760840.82\r\nCentral America and the
Caribbean,Jamaica,Beverages,Offline,L,9/4/2010,262056386,10/24/2010,7163,47.45,31.79,339884.35,227711.77,112172.58\r\nSub-Saharan Africa,Equatorial
Guinea,Office
Supplies,Online,M,5/2/2010,211114585,5/14/2010,2352,651.21,524.96,1531645.92,1234705.92,296940.00\r\nSub-Saharan Africa,Swaziland,Office
Supplies,Offline,H,10/3/2013,405785882,10/22/2013,9915,651.21,524.96,6456747.15,5204978.40,1251768.75\r\nCentral America and the Caribbean,Trinidad and
Tobago,Vegetables,Offline,M,3/6/2011,280494105,4/14/2011,3294,154.06,90.93,507473.64,299523.42,207950.22\r\nEurope,Sweden,Baby
Food,Online,L,8/7/2016,689975583,8/12/2016,7963,255.28,159.42,2032794.64,1269461.46,763333.18\r\nEurope,Belarus,Office
Supplies,Online,L,1/11/2011,759279143,2/18/2011,6426,651.21,524.96,4184675.46,3373392.96,811282.50\r\nSub-Saharan Africa,Guinea-Bissau,Office
Supplies,Offline,C,5/21/2014,133766114,6/12/2014,3221,651.21,524.96,2097547.41,1690896.16,406651.25\r\nAsia,Mongolia,Beverages,Online,M,8/3/2013,329110324,9/2/2013,9913,47.45,31.79,470371.85,315134.27,155237.58\r\nMiddle East
and North
Africa,Turkey,Meat,Online,L,10/5/2011,681298100,11/20/2011,103,421.89,364.69,43454.67,37563.07,5891.60\r\nSub-Saharan Africa,Central African
Republic,Snacks,Offline,L,11/15/2016,596628272,12/30/2016,4419,152.58,97.44,674251.02,430587.36,243663.66\r\nSub-Saharan Africa,Equatorial Guinea,Office
Supplies,Offline,L,4/3/2015,901712167,4/17/2015,5523,651.21,524.96,3596632.83,2899354.08,697278.75\r\nAsia,Laos,Beverages,Online,M,3/22/2013,693473613,4/21/2013,3107,47.45,31.79,147427.15,98771.53,48655.62\r\nEurope,Armenia,Meat
,Online,C,8/2/2010,489148938,9/1/2010,8896,421.89,364.69,3753133.44,3244282.24,508851.20\r\nEurope,Greece,Household,Online,L,1/5/2012,876286971,2/15/2012,1643,668.27,502.54,1097967.61,825673.22,272294.39\r\nMiddle East and North
Africa,Israel,Personal
Care,Offline,H,8/26/2015,262749040,8/30/2015,2135,81.73,56.67,174493.55,120990.45,53503.10\r\nAsia,Bhutan,Meat,Online,H,12/9/2016,726708972,1/26/2017,8189,421.89,364.69,3454857.21,2986446.41,468410.80\r\nAustralia and
Oceania,Vanuatu,Vegetables,Online,L,5/17/2012,366653096,5/31/2012,9654,154.06,90.93,1487295.24,877838.22,609457.02\r\nSub-Saharan
Africa,Burundi,Vegetables,Online,M,11/17/2010,951380240,12/20/2010,3410,154.06,90.93,525344.60,310071.30,215273.30\r\nEurope,Ukraine,Cosmetics,Online,M,11/13/2014,270001733,1/1/2015,8368,437.20,263.33,3658489.60,2203545.44,1454944.16\r\nEurope,Croatia,Beverages,Online,C,6/16/2016,681941401,7/28/2016,470,47.45,31.79,22301.50,14941.30,7360.20\r\nSub-Saharan
Africa,Madagascar,Fruits,Online,L,5/31/2016,566935575,6/7/2016,7690,9.33,6.92,71747.70,53214.80,18532.90\r\nAsia,Malaysia,Snacks,Offline,M,10/6/2012,175033080,11/5/2012,5033,152.58,97.44,767935.14,490415.52,277519.62\r\nAsia,Uzbekistan,Office
Supplies,Offline,L,3/10/2012,276595246,3/15/2012,9535,651.21,524.96,6209287.35,5005493.60,1203793.75\r\nEurope,Italy,Office
Supplies,Online,M,1/26/2011,812295901,2/13/2011,5263,651.21,524.96,3427318.23,2762864.48,664453.75\r\nAsia,Nepal,Vegetables,Offline,C,6/2/2014,443121373

```
,6/19/2014,8316,154.06,90.93,1281162.96,756173.88,524989.08\r\nAustralia and
Oceania,Fiji,Personal
Care,Offline,H,12/17/2016,600370490,1/25/2017,1824,81.73,56.67,149075.52,103
366.08,45709.44\r\nEurope,Portugal,Office
Supplies,Online,L,6/27/2014,535654580,7/29/2014,949,651.21,524.96,617998.29,
498187.04,119811.25\r\nCentral America and the
Caribbean,Panama,Cosmetics,Offline,H,3/17/2015,470897471,4/22/2015,7881,437.
20,263.33,3445573.20,2075303.73,1370269.47\r\nEurope,Belarus,Beverages,Offli
ne,L,4/3/2013,248335492,4/4/2013,6846,47.45,31.79,324842.70,217634.34,107208
.36\r\nSub-Saharan
Africa,Botswana,Clothes,Offline,C,3/8/2015,680517470,3/25/2015,9097,109.28,3
5.84,994120.16,326036.48,668083.68\r\nSub-Saharan Africa,Tanzania,Personal
Care,Online,M,6/21/2013,400304734,7/29/2013,7921,81.73,56.67,647383.33,44888
3.07,198500.26\r\nEurope,Romania,Office
Supplies,Offline,C,1/6/2013,810871112,1/8/2013,3636,651.21,524.96,2367799.56
,1908754.56,459045.00\r\nSub-Saharan
Africa,Mali,Cereal,Online,L,3/17/2012,235702931,4/3/2012,8590,205.70,117.11,
1766963.00,1005974.90,760988.10\r\nSub-Saharan Africa,Central African
Republic,Office
Supplies,Offline,C,4/18/2014,668599021,5/12/2014,2163,651.21,524.96,1408567.
23,1135488.48,273078.75\r\nSub-Saharan Africa,Niger,Baby
Food,Online,M,1/3/2016,123670709,2/1/2016,5766,255.28,159.42,1471944.48,9192
15.72,552728.76\r\nEurope,Austria,Office
Supplies,Online,L,5/12/2011,285341823,6/8/2011,7841,651.21,524.96,5106137.61
,4116211.36,989926.25\r\nAsia,India,Fruits,Online,H,7/29/2010,658348691,8/22
/2010,8862,9.33,6.92,82682.46,61325.04,21357.42\r\nEurope,Luxembourg,Baby
Food,Offline,L,8/2/2013,817740142,8/19/2013,6335,255.28,159.42,1617198.80,10
09925.70,607273.10\r\nSub-Saharan Africa,Cape
Verde,Beverages,Offline,H,10/23/2013,858877503,11/6/2013,9794,47.45,31.79,46
4725.30,311351.26,153374.04\r\nEurope,Sweden,Vegetables,Offline,M,2/5/2017,9
47434604,2/19/2017,5808,154.06,90.93,894780.48,528121.44,366659.04\r\nEurope
,Iceland,Meat,Offline,H,3/20/2015,869397771,4/17/2015,2975,421.89,364.69,125
5122.75,1084952.75,170170.00\r\nMiddle East and North Africa,Qatar,Personal
Care,Offline,L,5/6/2012,481065833,5/8/2012,6925,81.73,56.67,565980.25,392439
.75,173540.50\r\nSub-Saharan Africa,South
Sudan,Meat,Online,C,9/30/2013,159050118,10/1/2013,5319,421.89,364.69,2244032
.91,1939786.11,304246.80\r\nEurope,United Kingdom,Office
Supplies,Online,M,5/20/2014,350274455,6/14/2014,2850,651.21,524.96,1855948.5
0,1496136.00,359812.50\r\nMiddle East and North Africa,Tunisia
,Cereal,Online,L,4/9/2010,221975171,5/17/2010,6241,205.70,117.11,1283773.70,
730883.51,552890.19\r\nNorth America,United States of America,Office
Supplies,Online,C,6/9/2017,811701095,7/19/2017,9247,651.21,524.96,6021738.87
,4854305.12,1167433.75\r\nSub-Saharan
Africa,Liberia,Cereal,Online,L,2/8/2015,977313554,3/29/2015,7653,205.70,117.
11,1574222.10,896242.83,677979.27\r\nSub-Saharan
Africa,Eritrea,Snacks,Offline,L,1/25/2010,546986377,2/10/2010,4279,152.58,97
.44,652889.82,416945.76,235944.06\r\nAsia,South
Korea,Fruits,Offline,L,3/7/2010,769205892,3/17/2010,3972,9.33,6.92,37058.76,
27486.24,9572.52\r\nSub-Saharan
Africa,Kenya,Clothes,Offline,M,1/3/2013,262770926,2/8/2013,8611,109.28,35.84
,941010.08,308618.24,632391.84\r\nSub-Saharan
Africa,Rwanda,Snacks,Online,M,3/6/2017,866792809,3/18/2017,2109,152.58,97.44
,321791.22,205500.96,116290.26\r\nCentral America and the
Caribbean,Cuba,Beverages,Offline,C,1/9/2011,890695369,2/23/2011,5408,47.45,3
1.79,256609.60,171920.32,84689.28\r\nMiddle East and North
Africa,Libya,Cereal,Offline,M,3/27/2014,964214932,3/31/2014,1480,205.70,117.
11,304436.00,173322.80,131113.20\r\nEurope,Czech
```

Republic, Snacks, Online, C, 6/28/2013, 887400329, 8/17/2013, 332, 152.58, 97.44, 5065
 6.56, 32350.08, 18306.48\r\nEurope, Montenegro, Beverages, Offline, M, 9/4/2011, 980
 612885, 9/4/2011, 3999, 47.45, 31.79, 189752.55, 127128.21, 62624.34\r\nEurope, Mont
 enegro, Clothes, Offline, M, 7/14/2016, 734526431, 8/2/2016, 1549, 109.28, 35.84, 1692
 74.72, 55516.16, 113758.56\r\nAsia, Philippines, Baby
 Food, Online, L, 2/23/2014, 160127294, 3/23/2014, 4079, 255.28, 159.42, 1041287.12, 65
 0274.18, 391012.94\r\nCentral America and the Caribbean, El
 Salvador, Clothes, Offline, L, 8/7/2010, 238714301, 9/13/2010, 9721, 109.28, 35.84, 10
 62310.88, 348400.64, 713910.24\r\nAustralia and
 Oceania, Tonga, Household, Online, M, 1/14/2013, 671898782, 2/6/2013, 8635, 668.27, 50
 2.54, 5770511.45, 4339432.90, 1431078.55\r\nSub-Saharan Africa, Democratic
 Republic of the Congo, Personal
 Care, Offline, H, 9/30/2010, 331604564, 11/17/2010, 8014, 81.73, 56.67, 654984.22, 454
 153.38, 200830.84\r\nMiddle East and North
 Africa, Afghanistan, Cereal, Online, M, 10/13/2016, 410067975, 11/20/2016, 7081, 205.
 70, 117.11, 1456561.70, 829255.91, 627305.79\r\nAustralia and
 Oceania, Tuvalu, Snacks, Offline, L, 3/16/2011, 369837844, 3/23/2011, 2091, 152.58, 97
 .44, 319044.78, 203747.04, 115297.74\r\nSub-Saharan
 Africa, Sudan, Fruits, Online, L, 12/26/2012, 193775498, 1/31/2013, 1331, 9.33, 6.92, 1
 2418.23, 9210.52, 3207.71\r\nSub-Saharan
 Africa, Niger, Clothes, Online, M, 9/2/2015, 835054767, 10/9/2015, 117, 109.28, 35.84,
 12785.76, 4193.28, 8592.48\r\nSub-Saharan
 Africa, Gabon, Household, Offline, C, 11/11/2013, 167161977, 12/24/2013, 5798, 668.27
 , 502.54, 3874629.46, 2913726.92, 960902.54\r\nAustralia and Oceania, East
 Timor, Vegetables, Offline, C, 8/4/2014, 633895957, 8/22/2014, 2755, 154.06, 90.93, 42
 4435.30, 250512.15, 173923.15\r\nNorth America, United States of
 America, Clothes, Offline, C, 10/21/2010, 699368035, 12/7/2010, 7398, 109.28, 35.84, 8
 08453.44, 265144.32, 543309.12\r\nMiddle East and North
 Africa, Jordan, Vegetables, Offline, L, 6/13/2015, 698002040, 7/29/2015, 3170, 154.06
 , 90.93, 488370.20, 288248.10, 200122.10\r\nEurope, Cyprus, Snacks, Offline, H, 3/29/
 2012, 584534299, 5/18/2012, 5544, 152.58, 97.44, 845903.52, 540207.36, 305696.16\r\n
 Sub-Saharan
 Africa, Malawi, Vegetables, Online, L, 6/22/2012, 384013640, 7/19/2012, 7025, 154.06,
 90.93, 1082271.50, 638783.25, 443488.25\r\nEurope, Iceland, Personal
 Care, Online, M, 5/10/2013, 641801393, 5/24/2013, 2149, 81.73, 56.67, 175637.77, 12178
 3.83, 53853.94\r\nMiddle East and North Africa, Israel, Personal
 Care, Online, M, 12/10/2016, 173571383, 1/11/2017, 2484, 81.73, 56.67, 203017.32, 1407
 68.28, 62249.04\r\nMiddle East and North Africa, United Arab
 Emirates, Snacks, Offline, H, 3/20/2011, 115309941, 4/6/2011, 1629, 152.58, 97.44, 248
 552.82, 158729.76, 89823.06\r\nAsia, China, Cosmetics, Offline, L, 9/22/2011, 773315
 894, 11/4/2011, 213, 437.20, 263.33, 93123.60, 56089.29, 37034.31\r\nSub-Saharan
 Africa, Kenya, Beverages, Online, M, 5/11/2012, 274200570, 6/26/2012, 897, 47.45, 31.7
 9, 42562.65, 28515.63, 14047.02\r\nMiddle East and North
 Africa, Somalia, Clothes, Offline, M, 11/15/2011, 414887797, 11/17/2011, 3374, 109.28
 , 35.84, 368710.72, 120924.16, 247786.56\r\nAustralia and
 Oceania, Tonga, Beverages, Offline, L, 1/27/2010, 812613904, 1/29/2010, 9367, 47.45, 3
 1.79, 444464.15, 297776.93, 146687.22\r\nAsia, Bangladesh, Baby
 Food, Online, H, 8/17/2011, 254927718, 9/7/2011, 7632, 255.28, 159.42, 1948296.96, 121
 6693.44, 731603.52\r\nMiddle East and North
 Africa, Egypt, Beverages, Offline, M, 9/6/2014, 749690568, 10/26/2014, 8954, 47.45, 31
 .79, 424867.30, 284647.66, 140219.64\r\nSub-Saharan
 Africa, Eritrea, Cereal, Offline, C, 9/3/2014, 775076282, 9/19/2014, 1150, 205.70, 117
 .11, 236555.00, 134676.50, 101878.50\r\nSub-Saharan Africa, Swaziland, Office
 Supplies, Online, H, 9/5/2015, 229571187, 9/18/2015, 4071, 651.21, 524.96, 2651075.91
 , 2137112.16, 513963.75\r\nAsia, Vietnam, Baby
 Food, Online, C, 6/20/2011, 881974112, 7/11/2011, 4594, 255.28, 159.42, 1172756.32, 73
 2375.48, 440380.84\r\nAustralia and Oceania, Marshall

Islands, Snacks, Online, L, 1/12/2012, 521396386, 2/14/2012, 1632, 152.58, 97.44, 2490
 10.56, 159022.08, 89988.48\r\nAsia, Taiwan, Household, Online, C, 1/23/2017, 6072618
 36, 2/22/2017, 1127, 668.27, 502.54, 753140.29, 566362.58, 186777.71\r\nEurope, Irel
 and, Vegetables, Online, M, 3/4/2012, 419306790, 3/12/2012, 1052, 154.06, 90.93, 16207
 1.12, 95658.36, 66412.76\r\nSub-Saharan
 Africa, Rwanda, Meat, Offline, H, 7/18/2010, 207580077, 7/18/2010, 6413, 421.89, 364.6
 9, 2705580.57, 2338756.97, 366823.60\r\nEurope, Sweden, Snacks, Online, M, 4/12/2011
 , 742443025, 4/15/2011, 4245, 152.58, 97.44, 647702.10, 413632.80, 234069.30\r\nSub-
 Saharan
 Africa, Gabon, Snacks, Offline, M, 10/3/2010, 164569461, 10/5/2010, 8615, 152.58, 97.4
 4, 1314476.70, 839445.60, 475031.10\r\nSub-Saharan Africa, South Africa, Baby
 Food, Online, L, 12/29/2013, 734945714, 2/12/2014, 5624, 255.28, 159.42, 1435694.72, 8
 96578.08, 539116.64\r\nEurope, United
 Kingdom, Clothes, Offline, C, 9/19/2015, 284870612, 10/7/2015, 8399, 109.28, 35.84, 91
 7842.72, 301020.16, 616822.56\r\nEurope, Albania, Fruits, Offline, M, 9/17/2011, 765
 955483, 10/7/2011, 2104, 9.33, 6.92, 19630.32, 14559.68, 5070.64\r\nAsia, Malaysia, S
 nacks, Offline, H, 3/11/2010, 600124156, 4/21/2010, 8929, 152.58, 97.44, 1362386.82, 8
 70041.76, 492345.06\r\nSub-Saharan
 Africa, Ghana, Household, Offline, L, 11/10/2012, 529612958, 12/11/2012, 3098, 668.27
 , 502.54, 2070300.46, 1556868.92, 513431.54\r\nCentral America and the
 Caribbean, Cuba, Clothes, Offline, H, 2/16/2011, 466970717, 3/18/2011, 5867, 109.28, 3
 5.84, 641145.76, 210273.28, 430872.48\r\nCentral America and the
 Caribbean, Saint
 Lucia, Cosmetics, Online, C, 8/13/2012, 845058763, 9/22/2012, 522, 437.20, 263.33, 228
 218.40, 137458.26, 90760.14\r\nEurope, Romania, Snacks, Offline, L, 8/28/2014, 36705
 0921, 8/31/2014, 7379, 152.58, 97.44, 1125887.82, 719009.76, 406878.06\r\nEurope, Po
 rtugal, Office
 Supplies, Online, L, 8/19/2015, 956433522, 9/12/2015, 8788, 651.21, 524.96, 5722833.4
 8, 4613348.48, 1109485.00\r\nEurope, Macedonia, Beverages, Online, C, 3/20/2011, 107
 005393, 5/4/2011, 4129, 47.45, 31.79, 195921.05, 131260.91, 64660.14\r\nAsia, China,
 Beverages, Offline, C, 4/16/2013, 332877862, 5/7/2013, 4811, 47.45, 31.79, 228281.95,
 152941.69, 75340.26\r\nEurope, Germany, Baby
 Food, Online, L, 11/13/2015, 618474757, 12/31/2015, 9279, 255.28, 159.42, 2368743.12,
 1479258.18, 889484.94\r\nEurope, Ireland, Household, Online, M, 1/10/2014, 46853240
 7, 2/11/2014, 8006, 668.27, 502.54, 5350169.62, 4023335.24, 1326834.38\r\nEurope, Po
 land, Office
 Supplies, Offline, M, 4/9/2015, 358099639, 4/29/2015, 8496, 651.21, 524.96, 5532680.1
 6, 4460060.16, 1072620.00\r\nSub-Saharan
 Africa, Namibia, Household, Online, H, 1/7/2013, 382537782, 1/27/2013, 285, 668.27, 50
 2.54, 190456.95, 143223.90, 47233.05\r\nAsia, Uzbekistan, Personal
 Care, Offline, H, 2/12/2013, 707520663, 3/15/2013, 9942, 81.73, 56.67, 812559.66, 5634
 13.14, 249146.52\r\nSub-Saharan
 Africa, Zimbabwe, Meat, Online, M, 11/28/2014, 219034612, 12/10/2014, 6064, 421.89, 36
 4.69, 2558340.96, 2211480.16, 346860.80\r\nAsia, Mongolia, Meat, Offline, M, 1/3/201
 5, 573378455, 1/17/2015, 4281, 421.89, 364.69, 1806111.09, 1561237.89, 244873.20\r\nn
 Europe, Norway, Personal
 Care, Online, H, 2/3/2011, 347163522, 3/22/2011, 2256, 81.73, 56.67, 184382.88, 127847
 .52, 56535.36\r\nMiddle East and North
 Africa, Oman, Snacks, Offline, M, 4/9/2013, 887313640, 4/21/2013, 4679, 152.58, 97.44,
 713921.82, 455921.76, 258000.06\r\nEurope, Serbia, Cosmetics, Online, H, 7/26/2017,
 461065137, 8/19/2017, 8275, 437.20, 263.33, 3617830.00, 2179055.75, 1438774.25\r\nS
 ub-Saharan Africa, Democratic Republic of the
 Congo, Fruits, Offline, H, 4/15/2017, 105966842, 5/19/2017, 6798, 9.33, 6.92, 63425.34
 , 47042.16, 16383.18\r\nEurope, Bulgaria, Baby
 Food, Online, M, 5/16/2014, 479880082, 5/23/2014, 6035, 255.28, 159.42, 1540614.80, 96
 2099.70, 578515.10\r\nAsia, Brunei, Baby
 Food, Online, H, 8/12/2015, 510978686, 9/30/2015, 8803, 255.28, 159.42, 2247229.84, 14

```

03374.26,843855.58\r\nEurope,Serbia,Snacks,Offline,C,9/20/2013,547748982,10/
14/2013,9951,152.58,97.44,1518323.58,969625.44,548698.14\r\nSub-Saharan
Africa,Ghana,Cereal,Offline,M,10/31/2013,108989799,12/9/2013,1358,205.70,117
.11,279340.60,159035.38,120305.22\r\nSub-Saharan
Africa,Malawi,Cereal,Offline,M,7/30/2014,133812463,8/9/2014,6936,205.70,117.
11,1426735.20,812274.96,614460.24\r\nSub-Saharan
Africa,Zimbabwe,Fruits,Offline,L,11/12/2011,731640803,12/30/2011,7627,9.33,6
.92,71159.91,52778.84,18381.07\r\nEurope,Cyprus,Snacks,Offline,C,3/25/2010,7
32211148,4/14/2010,6405,152.58,97.44,977274.90,624103.20,353171.70\r\nCentra
l America and the
Caribbean,Nicaragua,Cereal,Online,M,7/4/2011,835572326,8/8/2011,3274,205.70,
117.11,673461.80,383418.14,290043.66\r\nEurope,Estonia,Baby
Food,Offline,C,1/1/2011,462085664,1/15/2011,271,255.28,159.42,69180.88,43202
.82,25978.06\r\nEurope,Estonia,Clothes,Online,C,6/16/2016,902424991,7/4/2016
,6463,109.28,35.84,706276.64,231633.92,474642.72\r\nEurope,Lithuania,Fruits,
Offline,H,12/17/2013,367576634,1/5/2014,2949,9.33,6.92,27514.17,20407.08,710
7.09\r\nSub-Saharan Africa,Republic of the
Congo,Meat,Offline,H,3/1/2017,738839423,3/31/2017,7859,421.89,364.69,3315633
.51,2866098.71,449534.80\r\nEurope,Czech Republic,Baby
Food,Online,C,7/2/2010,817824685,7/27/2010,1353,255.28,159.42,345393.84,2156
95.26,129698.58\r\nSub-Saharan
Africa,Cameroon,Snacks,Online,C,7/16/2013,376456248,8/1/2013,624,152.58,97.4
4,95209.92,60802.56,34407.36\r\nAsia,Vietnam,Office
Supplies,Online,H,8/16/2016,606970441,9/16/2016,4897,651.21,524.96,3188975.3
7,2570729.12,618246.25\r\nEurope,Moldova
,Meat,Offline,L,12/16/2014,971916091,1/19/2015,424,421.89,364.69,178881.36,1
54628.56,24252.80\r\nMiddle East and North Africa,Bahrain,Office
Supplies,Offline,L,5/14/2012,554154527,5/15/2012,5494,651.21,524.96,3577747.
74,2884130.24,693617.50\r\nEurope,Hungary,Household,Online,L,7/18/2017,30685
9576,7/19/2017,5423,668.27,502.54,3624028.21,2725274.42,898753.79\r\nAustral
ia and Oceania,Marshall Islands,Personal
Care,Offline,L,7/9/2017,803517568,7/21/2017,7559,81.73,56.67,617797.07,42836
8.53,189428.54\r\nMiddle East and North Africa,Iraq,Office
Supplies,Online,C,9/30/2011,887927329,10/2/2011,6283,651.21,524.96,4091552.4
3,3298323.68,793228.75\r\nEurope,Albania,Vegetables,Online,H,11/24/2015,8242
00189,11/26/2015,8006,154.06,90.93,1233404.36,727985.58,505418.78\r\nSub-
Saharan Africa,Lesotho,Office
Supplies,Online,H,8/14/2012,946759974,9/14/2012,6170,651.21,524.96,4017965.7
0,3239003.20,778962.50\r\nMiddle East and North
Africa,Lebanon,Clothes,Offline,H,12/6/2015,310343015,12/28/2015,6249,109.28,
35.84,682890.72,223964.16,458926.56\r\nEurope,Hungary,Vegetables,Online,C,5/
25/2014,739998137,7/9/2014,748,154.06,90.93,115236.88,68015.64,47221.24\r\nA
sia,Japan,Beverages,Online,H,10/18/2012,981086671,11/21/2012,4203,47.45,31.7
9,199432.35,133613.37,65818.98\r\nEurope,Georgia,Office
Supplies,Offline,L,3/13/2013,749282443,3/25/2013,8180,651.21,524.96,5326897.
80,4294172.80,1032725.00\r\nEurope,Estonia,Office
Supplies,Online,C,2/23/2011,280571782,3/11/2011,6280,651.21,524.96,4089598.8
0,3296748.80,792850.00\r\nEurope,Luxembourg,Household,Online,C,8/15/2014,781
253516,9/1/2014,9131,668.27,502.54,6101973.37,4588692.74,1513280.63\r\nSub-
Saharan Africa,Swaziland,Personal
Care,Online,H,7/6/2017,377938973,7/11/2017,9396,81.73,56.67,767935.08,532471
.32,235463.76\r\nEurope,Romania,Clothes,Offline,C,12/31/2010,867551982,1/3/2
011,6765,109.28,35.84,739279.20,242457.60,496821.60\r\nSub-Saharan
Africa,Ethiopia,Personal
Care,Offline,C,1/13/2010,967328870,1/15/2010,2964,81.73,56.67,242247.72,1679
69.88,74277.84\r\nSub-Saharan Africa,Chad,Office
Supplies,Offline,C,9/17/2011,364818465,10/16/2011,6746,651.21,524.96,4393062

```

```
.66,3541380.16,851682.50\r\nMiddle East and North Africa,Morocco,Office
Supplies,Online,C,3/8/2014,167882096,3/31/2014,8898,651.21,524.96,5794466.58
,4671094.08,1123372.50\r\nNorth
America,Mexico,Clothes,Online,H,11/18/2012,654693591,12/1/2012,7237,109.28,3
5.84,790859.36,259374.08,531485.28\r\nSub-Saharan Africa,Nigeria,Personal
Care,Offline,H,11/18/2011,823739278,12/29/2011,1612,81.73,56.67,131748.76,91
352.04,40396.72\r\nCentral America and the Caribbean,Trinidad and
Tobago,Beverages,Offline,L,7/12/2012,643817985,8/19/2012,8904,47.45,31.79,42
2494.80,283058.16,139436.64\r\nEurope,Moldova ,Personal
Care,Offline,H,3/30/2017,604041039,5/15/2017,8022,81.73,56.67,655638.06,4546
06.74,201031.32\r\nAustralia and Oceania,Solomon Islands,Personal
Care,Online,H,7/26/2010,363832271,9/12/2010,4909,81.73,56.67,401212.57,27819
3.03,123019.54\r\nAsia,India,Personal
Care,Online,L,12/24/2015,102928006,1/31/2016,7539,81.73,56.67,616162.47,4272
35.13,188927.34\r\nSub-Saharan Africa,Burkina Faso,Office
Supplies,Offline,M,5/15/2016,971377074,5/15/2016,917,651.21,524.96,597159.57
,481388.32,115771.25\r\nAustralia and
Oceania,Kiribati,Meat,Online,L,11/3/2010,139540803,12/4/2010,2079,421.89,364
.69,877109.31,758190.51,118918.80\r\nMiddle East and North
Africa,Israel,Meat,Offline,M,12/1/2010,248093020,12/16/2010,5093,421.89,364.
69,2148685.77,1857366.17,291319.60\r\nSub-Saharan
Africa,Comoros,Snacks,Offline,L,1/16/2014,858020055,1/17/2014,6056,152.58,97
.44,924024.48,590096.64,333927.84\r\nMiddle East and North Africa,Iran,Baby
Food,Offline,H,12/11/2014,700620734,1/5/2015,8099,255.28,159.42,2067512.72,1
291142.58,776370.14\r\nAsia,Vietnam,Cosmetics,Offline,L,12/24/2016,827506387
,1/30/2017,6384,437.20,263.33,2791084.80,1681098.72,1109986.08\r\nCentral
America and the
Caribbean,Belize,Household,Online,M,3/21/2013,560600841,4/14/2013,3101,668.2
7,502.54,2072305.27,1558376.54,513928.73\r\nEurope,Belarus,Personal
Care,Offline,H,12/8/2012,642140424,1/16/2013,2476,81.73,56.67,202363.48,1403
14.92,62048.56\r\nNorth America,United States of America,Baby
Food,Offline,C,2/13/2015,984673964,3/5/2015,5763,255.28,159.42,1471178.64,91
8737.46,552441.18\r\nEurope,Poland,Beverages,Online,L,3/28/2012,221062791,4/
18/2012,6247,47.45,31.79,296420.15,198592.13,97828.02\r\nNorth
America,Canada,Vegetables,Offline,L,10/7/2016,654480731,11/8/2016,4247,154.0
6,90.93,654292.82,386179.71,268113.11\r\nMiddle East and North
Africa,Israel,Beverages,Offline,C,12/15/2011,608414113,12/23/2011,2111,47.45
,31.79,100166.95,67108.69,33058.26\r\nMiddle East and North
Africa,Lebanon,Household,Online,L,3/8/2016,276661765,4/20/2016,9219,668.27,5
02.54,6160781.13,4632916.26,1527864.87\r\nEurope,Andorra,Baby
Food,Online,L,1/18/2011,373335015,2/28/2011,6982,255.28,159.42,1782364.96,11
13070.44,669294.52\r\nEurope,Slovakia,Clothes,Online,L,4/11/2013,782857692,5
/28/2013,3843,109.28,35.84,419963.04,137733.12,282229.92\r\nSub-Saharan
Africa,Liberia,Fruits,Online,H,5/18/2010,109966123,6/5/2010,274,9.33,6.92,25
56.42,1896.08,660.34\r\nCentral America and the Caribbean,Antigua and
Barbuda
,Cereal,Offline,M,6/5/2017,629709136,6/6/2017,3782,205.70,117.11,777957.40,4
42910.02,335047.38\r\nAsia,China,Personal
Care,Online,L,9/11/2012,637448060,9/15/2012,3901,81.73,56.67,318828.73,22106
9.67,97759.06\r\nSub-Saharan Africa,Niger,Baby
Food,Online,H,3/8/2017,298856723,4/3/2017,7200,255.28,159.42,1838016.00,1147
824.00,690192.00\r\nEurope,United
Kingdom,Household,Offline,L,1/28/2015,299921452,2/23/2015,2278,668.27,502.54
,1522319.06,1144786.12,377532.94\r\nAsia,Bangladesh,Personal
Care,Offline,M,7/26/2010,496941077,7/29/2010,4763,81.73,56.67,389279.99,2699
19.21,119360.78\r\nAsia,Myanmar,Snacks,Online,L,6/24/2016,366526925,7/14/201
6,2317,152.58,97.44,353527.86,225768.48,127759.38\r\nAustralia and
```

```

Oceania,Tonga,Meat,Offline,M,8/18/2012,355602824,9/15/2012,9633,421.89,364.6
9,4064066.37,3513058.77,551007.60\r\nSub-Saharan Africa,Guinea-
Bissau,Vegetables,Online,C,3/11/2010,531405103,4/19/2010,3434,154.06,90.93,5
29042.04,312253.62,216788.42\r\nAustralia and
Oceania,Nauru,Vegetables,Offline,M,1/14/2010,131482589,1/20/2010,7475,154.06
,90.93,1151598.50,679701.75,471896.75\r\nSub-Saharan
Africa,Swaziland,Cereal,Online,L,2/10/2014,713696610,3/28/2014,7542,205.70,1
17.11,1551389.40,883243.62,668145.78\r\nEurope,Finland,Vegetables,Online,C,1
/21/2014,306220996,1/30/2014,6452,154.06,90.93,993995.12,586680.36,407314.76
\r\nAustralia and Oceania,Papua New
Guinea,Household,Offline,L,2/28/2010,157542073,3/15/2010,9055,668.27,502.54,
6051184.85,4550499.70,1500685.15\r\nSub-Saharan Africa,Mauritius ,Personal
Care,Online,L,2/18/2015,686458671,3/8/2015,7230,81.73,56.67,590907.90,409724
.10,181183.80\r\nSub-Saharan Africa,Mozambique,Office
Supplies,Online,M,6/14/2012,132082116,7/22/2012,4888,651.21,524.96,3183114.4
8,2566004.48,617110.00\r\nEurope,Bulgaria,Clothes,Online,L,3/5/2013,40383623
8,4/3/2013,2972,109.28,35.84,324780.16,106516.48,218263.68\r\nEurope,Spain,H
ousehold,Online,C,4/10/2014,331457364,4/23/2014,4455,668.27,502.54,2977142.8
5,2238815.70,738327.15\r\nAustralia and
Oceania,Vanuatu,Meat,Online,H,7/26/2017,614994323,9/12/2017,9341,421.89,364.
69,3940874.49,3406569.29,534305.20\r\nEurope,Belgium,Fruits,Offline,L,10/19/
2010,674808442,10/24/2010,9669,9.33,6.92,90211.77,66909.48,23302.29\r\nEurop
e,Belgium,Baby
Food,Offline,L,11/8/2016,901573550,12/23/2016,4503,255.28,159.42,1149525.84,
717868.26,431657.58\r\nSub-Saharan Africa,Guinea-
Bissau,Clothes,Online,L,3/31/2014,406275975,5/10/2014,4944,109.28,35.84,5402
80.32,177192.96,363087.36\r\nSub-Saharan
Africa,Togo,Vegetables,Online,C,8/18/2016,170214545,8/19/2016,9121,154.06,90
.93,1405181.26,829372.53,575808.73\r\nSub-Saharan Africa,Cote
d'Ivoire,Personal
Care,Offline,C,1/3/2016,795000588,1/8/2016,7196,81.73,56.67,588129.08,407797
.32,180331.76\r\nSub-Saharan Africa,Republic of the
Congo,Fruits,Offline,C,10/21/2016,252557933,11/4/2016,6360,9.33,6.92,59338.8
0,44011.20,15327.60\r\nMiddle East and North Africa,Libya,Baby
Food,Offline,M,12/10/2016,635122907,12/13/2016,5837,255.28,159.42,1490069.36
,930534.54,559534.82\r\nAustralia and Oceania,East
Timor,Vegetables,Online,C,8/12/2011,505244338,9/19/2011,1882,154.06,90.93,28
9940.92,171130.26,118810.66\r\nEurope,Switzerland,Clothes,Offline,H,3/23/201
2,745783555,5/9/2012,2782,109.28,35.84,304016.96,99706.88,204310.08\r\nAustr
alia and
Oceania,Palau,Snacks,Offline,M,4/27/2012,509914386,6/11/2012,3853,152.58,97.
44,587890.74,375436.32,212454.42\r\nMiddle East and North
Africa,Jordan,Household,Online,M,1/29/2014,371123158,2/9/2014,2445,668.27,50
2.54,1633920.15,1228710.30,405209.85\r\nEurope,Slovenia,Household,Online,H,1
2/13/2016,973208701,12/28/2016,2936,668.27,502.54,1962040.72,1475457.44,4865
83.28\r\nAsia,South Korea,Baby
Food,Online,L,11/21/2013,780282342,12/27/2013,1739,255.28,159.42,443931.92,2
77231.38,166700.54\r\nEurope,Norway,Clothes,Online,H,4/7/2010,126767909,5/22
/2010,2296,109.28,35.84,250906.88,82288.64,168618.24\r\nMiddle East and
North Africa,Afghanistan,Baby
Food,Online,M,7/8/2012,767401731,7/30/2012,80,255.28,159.42,20422.40,12753.6
0,7668.80\r\nAsia,Bangladesh,Personal
Care,Online,L,10/15/2016,927232635,11/24/2016,7597,81.73,56.67,620902.81,430
521.99,190380.82\r\nSub-Saharan
Africa,Guinea,Meat,Offline,M,9/18/2012,251621949,10/20/2012,9381,421.89,364.
69,3957750.09,3421156.89,536593.20\r\nCentral America and the
Caribbean,Cuba,Office

```


Supplies,Offline,H,7/2/2017,256243503,7/23/2017,7002,651.21,524.96,4559772.42,3675769.92,884002.50\r\nEurope,Russia,Cosmetics,Offline,C,7/21/2011,277083623,9/2/2011,4056,437.20,263.33,1773283.20,1068066.48,705216.72\r\nSub-Saharan Africa,Seychelles

,Vegetables,Offline,L,6/1/2010,620441138,6/22/2010,1175,154.06,90.93,181020.50,106842.75,74177.75\r\nAsia,South Korea,Office

Supplies,Offline,M,7/26/2015,312927377,9/7/2015,1020,651.21,524.96,664234.20,535459.20,128775.00\r\nSub-Saharan Africa,Ghana,Baby

Food,Offline,L,8/6/2010,251466166,9/8/2010,3282,255.28,159.42,837828.96,523216.44,314612.52\r\nCentral America and the Caribbean,Costa Rica,Office

Supplies,Online,H,6/20/2010,953293836,7/22/2010,9685,651.21,524.96,6306968.85,5084237.60,1222731.25\r\nEurope,Romania,Cereal,Online,C,4/8/2012,305959212,4/23/2012,8985,205.70,117.11,1848214.50,1052233.35,795981.15\r\nEurope,Czech Republic,Cereal,Online,L,2/27/2014,317323625,3/24/2014,1967,205.70,117.11,404611.90,230355.37,174256.53\r\nEurope,Liechtenstein,Household,Offline,L,7/25/2011,365560901,9/1/2011,6449,668.27,502.54,4309673.23,3240880.46,1068792.77\r\nSub-Saharan Africa,Seychelles ,Baby

Food,Online,M,2/18/2016,349157369,4/5/2016,2279,255.28,159.42,581783.12,363318.18,218464.94\r\nMiddle East and North Africa,Somalia,Baby

Food,Online,L,1/24/2014,236911857,2/25/2014,6338,255.28,159.42,1617964.64,1010403.96,607560.68\r\nAustralia and Oceania,Solomon Islands,Personal

Care,Offline,H,5/10/2015,517935693,6/16/2015,7536,81.73,56.67,615917.28,427065.12,188852.16\r\nSub-Saharan Africa,Uganda,Clothes,Offline,C,2/13/2012,851652705,3/27/2012,1816,109.28,35.84,198452.48,65085.44,133367.04\r\nSub-Saharan Africa,Equatorial Guinea,Cereal,Offline,M,9/7/2012,517799222,10/23/2012,7151,205.70,117.11,1470960.70,837453.61,633507.09\r\nCentral America and the Caribbean,Costa Rica,Office

Supplies,Offline,C,2/4/2015,666424071,3/4/2015,8547,651.21,524.96,5565891.87,4486833.12,1079058.75\r\nEurope,Moldova

,Fruits,Offline,C,11/16/2010,267888581,12/22/2010,3039,9.33,6.92,28353.87,21029.88,7323.99\r\nSub-Saharan Africa,Burkina Faso,Vegetables,Online,L,7/20/2011,162866580,7/26/2011,4695,154.06,90.93,723311.70,426916.35,296395.35\r\nCentral America and the Caribbean,Guatemala,Beverages,Offline,H,7/26/2014,812344396,8/30/2014,9614,47.45,31.79,456184.30,305629.06,150555.24\r\nSub-Saharan Africa,Swaziland,Meat,Online,M,8/24/2014,947620856,9/3/2014,924,421.89,364.69,389826.36,336973.56,52852.80\r\nAsia,Maldives,Vegetables,Online,H,2/25/2015,720307290,3/28/2015,3789,154.06,90.93,583733.34,344533.77,239199.57\r\nAsia,Thailand,Household,Online,H,9/21/2016,352327525,10/27/2016,399,668.27,502.54,266639.73,200513.46,66126.27\r\nSub-Saharan Africa,Sudan,Household,Online,C,6/28/2013,585917890,7/23/2013,4979,668.27,502.54,3327316.33,2502146.66,825169.67\r\nCentral America and the Caribbean,Costa Rica,Meat,Offline,L,1/5/2012,433627212,2/13/2012,8783,421.89,364.69,3705459.87,3203072.27,502387.60\r\nEurope,Denmark,Beverages,Online,C,5/1/2012,328316819,5/30/2012,5098,47.45,31.79,241900.10,162065.42,79834.68\r\nSub-Saharan Africa,Angola,Cereal,Offline,C,10/13/2011,773160541,11/21/2011,4240,205.70,117.11,872168.00,496546.40,375621.60\r\nAustralia and Oceania,Papua New Guinea,Household,Online,M,4/27/2016,991644704,5/18/2016,8559,668.27,502.54,5719722.93,4301239.86,1418483.07\r\nAsia,North Korea,Meat,Online,M,1/19/2014,277568137,2/7/2014,7435,421.89,364.69,3136752.15,2711470.15,425282.00\r\nCentral America and the Caribbean,El Salvador,Fruits,Online,C,11/6/2016,245042169,12/15/2016,2278,9.33,6.92,21253.74,15763.76,5489.98\r\nSub-Saharan Africa,Burkina Faso,Household,Online,M,2/28/2011,778490626,3/24/2011,1531,668.27,502.54,102

```

3121.37,769388.74,253732.63\r\nMiddle East and North Africa,Yemen,Baby
Food,Online,C,10/11/2014,482649838,11/13/2014,5668,255.28,159.42,1446927.04,
903592.56,543334.48\r\nSub-Saharan Africa,Republic of the
Congo,Beverages,Online,L,6/25/2012,732568633,7/5/2012,2193,47.45,31.79,10405
7.85,69715.47,34342.38\r\nEurope,Andorra,Household,Online,M,11/6/2012,723608
338,11/23/2012,642,668.27,502.54,429029.34,322630.68,106398.66\r\nCentral
America and the Caribbean,Dominican
Republic,Household,Offline,H,2/24/2014,621442782,4/14/2014,7584,668.27,502.5
4,5068159.68,3811263.36,1256896.32\r\nMiddle East and North
Africa,Israel,Baby
Food,Offline,M,9/19/2015,212058293,10/6/2015,1616,255.28,159.42,412532.48,25
7622.72,154909.76\r\nAustralia and Oceania,Solomon
Islands,Snacks,Offline,L,3/4/2014,251753699,3/24/2014,8369,152.58,97.44,1276
942.02,815475.36,461466.66\r\nSub-Saharan
Africa,Liberia,Fruits,Online,M,10/8/2014,217140328,10/30/2014,5503,9.33,6.92
,51342.99,38080.76,13262.23\r\nSub-Saharan
Africa,Mali,Vegetables,Online,C,6/19/2012,555142009,7/10/2012,7712,154.06,90
.93,1188110.72,701252.16,486858.56\r\nAsia,Uzbekistan,Clothes,Online,C,11/11
/2010,432995069,12/13/2010,1718,109.28,35.84,187743.04,61573.12,126169.92\r\
nMiddle East and North Africa,Tunisia ,Personal
Care,Offline,H,11/1/2010,888248336,11/7/2010,1276,81.73,56.67,104287.48,7231
0.92,31976.56\r\nEurope,Vatican
City,Vegetables,Online,C,4/28/2014,778763139,5/9/2014,2173,154.06,90.93,3347
72.38,197590.89,137181.49\r\nSub-Saharan
Africa,Djibouti,Snacks,Offline,H,12/22/2012,832713305,2/9/2013,7227,152.58,9
7.44,1102695.66,704198.88,398496.78\r\nEurope,Ukraine,Household,Offline,M,8/
25/2014,498585164,9/29/2014,1285,668.27,502.54,858726.95,645763.90,212963.05
\r\nAustralia and Oceania,East
Timor,Fruits,Offline,M,11/26/2016,195177543,12/23/2016,6227,9.33,6.92,58097.
91,43090.84,15007.07\r\nSub-Saharan
Africa,Uganda,Cereal,Online,C,10/20/2010,861601769,12/2/2010,5965,205.70,117
.11,1227000.50,698561.15,528439.35\r\nSub-Saharan
Africa,Guinea,Meat,Online,H,12/18/2014,807281672,1/26/2015,1441,421.89,364.6
9,607943.49,525518.29,82425.20\r\nSub-Saharan Africa,Equatorial
Guinea,Clothes,Offline,H,3/20/2011,661953580,4/24/2011,5629,109.28,35.84,615
137.12,201743.36,413393.76\r\nEurope,Malta,Cosmetics,Online,M,7/12/2016,2256
66320,8/21/2016,8534,437.20,263.33,3731064.80,2247258.22,1483806.58\r\nEurop
e,Cyprus,Household,Offline,L,1/26/2011,718781220,2/19/2011,2191,668.27,502.5
4,1464179.57,1101065.14,363114.43\r\nEurope,Czech Republic,Office
Supplies,Online,L,2/24/2010,731972110,4/15/2010,5668,651.21,524.96,3691058.2
8,2975473.28,715585.00\r\nMiddle East and North
Africa,Libya,Vegetables,Online,C,1/2/2015,276225316,2/9/2015,64,154.06,90.93
,9859.84,5819.52,4040.32\r\nAsia,Vietnam,Office
Supplies,Offline,C,7/26/2016,332839667,7/27/2016,3509,651.21,524.96,2285095.
89,1842084.64,443011.25\r\nMiddle East and North
Africa,Jordan,Vegetables,Online,C,6/1/2014,603426492,6/15/2014,6163,154.06,9
0.93,949471.78,560401.59,389070.19\r\nSub-Saharan
Africa,Mali,Beverages,Offline,H,12/21/2012,859909617,1/29/2013,5220,47.45,31
.79,247689.00,165943.80,81745.20\r\nEurope,Czech
Republic,Household,Online,L,2/27/2010,494525372,3/2/2010,9902,668.27,502.54,
6617209.54,4976151.08,1641058.46\r\nEurope,Slovakia,Vegetables,Online,M,4/24
/2016,769822585,5/15/2016,6465,154.06,90.93,995997.90,587862.45,408135.45\r\
nSub-Saharan
Africa,Zimbabwe,Vegetables,Offline,C,7/22/2012,768662583,8/10/2012,3195,154.
06,90.93,492221.70,290521.35,201700.35\r\nCentral America and the
Caribbean,Honduras,Cereal,Online,M,2/22/2015,544219195,3/9/2015,5409,205.70,
117.11,1112631.30,633447.99,479183.31\r\nEurope,Switzerland,Beverages,Offlin

```

```

e,L,2/10/2011,669978749,3/20/2011,455,47.45,31.79,21589.75,14464.45,7125.30\r\nSub-Saharan Africa,South
Africa,Cosmetics,Offline,L,1/21/2015,889740073,1/26/2015,2715,437.20,263.33,1186998.00,714940.95,472057.05\r\nSub-Saharan
Africa,Uganda,Beverages,Online,M,5/10/2012,567614495,6/28/2012,8598,47.45,31.79,407975.10,273330.42,134644.68\r\nMiddle East and North
Africa,Iran,Vegetables,Offline,M,12/16/2015,938025844,1/21/2016,1547,154.06,90.93,238330.82,140668.71,97662.11\r\nMiddle East and North
Africa,Algeria,Vegetables,Online,C,2/25/2017,155710446,2/25/2017,7036,154.06,90.93,1083966.16,639783.48,444182.68\r\nSub-Saharan Africa,Central African
Republic,Baby
Food,Online,L,1/31/2012,945717132,2/13/2012,7570,255.28,159.42,1932469.60,1206809.40,725660.20\r\nCentral America and the Caribbean,The
Bahamas,Cosmetics,Offline,C,2/7/2013,253407227,2/15/2013,7685,437.20,263.33,3359882.00,2023691.05,1336190.95\r\nSub-Saharan Africa,South
Africa,Household,Offline,H,9/14/2014,494454562,9/22/2014,8948,668.27,502.54,5979679.96,4496727.92,1482952.04\r\nSub-Saharan
Africa,Benin,Cereal,Online,M,6/5/2012,104845464,7/24/2012,4957,205.70,117.11,1019654.90,580514.27,439140.63\r\nEurope,Hungary,Cosmetics,Online,M,4/5/2014,290878760,4/7/2014,6344,437.20,263.33,2773596.80,1670565.52,1103031.28\r\nEurope,Austria,Office
Supplies,Offline,C,2/2/2014,979165780,2/10/2014,5768,651.21,524.96,3756179.28,3027969.28,728210.00\r\nAsia,Tajikistan,Office
Supplies,Offline,C,9/1/2010,366630351,10/12/2010,2923,651.21,524.96,1903486.83,1534458.08,369028.75\r\nEurope,Portugal,Office
Supplies,Online,C,5/17/2011,770508801,6/25/2011,9532,651.21,524.96,6207333.72,5003918.72,1203415.00\r\nEurope,Belgium,Beverages,Offline,M,4/21/2013,978349959,5/21/2013,4349,47.45,31.79,206360.05,138254.71,68105.34\r\nEurope,Slovenia,Beverages,Offline,L,8/10/2014,298015153,8/14/2014,8161,47.45,31.79,387239.45,259438.19,127801.26\r\nEurope,Czech
Republic,Snacks,Online,M,9/19/2010,807678210,10/30/2010,8786,152.58,97.44,1340567.88,856107.84,484460.04\r\nAustralia and Oceania,Marshall
Islands,Personal
Care,Offline,C,4/13/2013,605825459,5/14/2013,6071,81.73,56.67,496182.83,344043.57,152139.26\r\nSub-Saharan
Africa,Sudan,Fruits,Online,C,1/28/2016,561255729,2/1/2016,6897,9.33,6.92,64349.01,47727.24,16621.77\r\nCentral America and the Caribbean,Dominican
Republic,Clothes,Online,H,12/9/2013,263080346,12/14/2013,175,109.28,35.84,19124.00,6272.00,12852.00\r\nSub-Saharan
Africa,Tanzania,Cereal,Offline,C,2/21/2014,270723140,3/9/2014,1848,205.70,117.11,380133.60,216419.28,163714.32\r\nEurope,Switzerland,Clothes,Offline,H,6/9/2017,763920438,7/10/2017,9888,109.28,35.84,1080560.64,354385.92,726174.72\r\nNorth
America,Greenland,Household,Online,L,4/17/2014,192721068,5/20/2014,9302,668.27,502.54,6216247.54,4674627.08,1541620.46\r\nAustralia and
Oceania,Tonga,Fruits,Offline,L,5/20/2011,227486360,6/8/2011,7124,9.33,6.92,66466.92,49298.08,17168.84\r\nMiddle East and North Africa,Saudi
Arabia,Vegetables,Online,M,8/28/2012,808890140,9/22/2012,7422,154.06,90.93,1143433.32,674882.46,468550.86\r\nCentral America and the
Caribbean,Belize,Cosmetics,Offline,C,7/25/2015,597918736,9/11/2015,6296,437.20,263.33,2752611.20,1657925.68,1094685.52\r\nSub-Saharan
Africa,Angola,Cosmetics,Online,H,10/27/2014,125870978,11/20/2014,6874,437.20,263.33,3005312.80,1810130.42,1195182.38\r\nAsia,Malaysia,Household,Online,H,10/18/2013,444358193,10/21/2013,4319,668.27,502.54,2886258.13,2170470.26,715787.87\r\nSub-Saharan
Africa,Ethiopia,Beverages,Online,C,2/15/2013,875304210,3/12/2013,822,47.45,31.79,39003.90,26131.38,12872.52\r\nNorth America,Greenland,Baby

```

```

Food,Offline,C,7/8/2014,360945355,8/16/2014,607,255.28,159.42,154954.96,9676
7.94,58187.02\r\nSub-Saharan
Africa,Benin,Cereal,Offline,C,12/12/2016,613830459,1/16/2017,4928,205.70,117
.11,1013689.60,577118.08,436571.52\r\nMiddle East and North
Africa,Yemen,Cereal,Offline,H,11/24/2012,266820847,12/10/2012,7073,205.70,11
7.11,1454916.10,828319.03,626597.07\r\nSub-Saharan Africa,Rwanda,Baby
Food,Offline,M,11/3/2014,723090350,11/27/2014,7358,255.28,159.42,1878350.24,
1173012.36,705337.88\r\nSub-Saharan
Africa,Mauritania,Meat,Offline,M,8/3/2013,306125295,8/15/2013,8132,421.89,36
4.69,3430809.48,2965659.08,465150.40\r\nAustralia and Oceania,New
Zealand,Personal
Care,Online,L,5/23/2012,109724509,6/16/2012,8775,81.73,56.67,717180.75,49727
9.25,219901.50\r\nAustralia and Oceania,Samoa
,Clothes,Offline,M,7/22/2015,847999322,8/6/2015,699,109.28,35.84,76386.72,25
052.16,51334.56\r\nAustralia and
Oceania,Fiji,Clothes,Online,H,2/1/2017,605373561,3/2/2017,2344,109.28,35.84,
256152.32,84008.96,172143.36\r\nSub-Saharan
Africa,Malawi,Beverages,Online,H,1/28/2012,686583554,2/22/2012,4186,47.45,31
.79,198625.70,133072.94,65552.76\r\nAustralia and Oceania,Marshall
Islands,Beverages,Offline,M,8/3/2015,666678130,9/21/2015,3729,47.45,31.79,17
6941.05,118544.91,58396.14\r\nCentral America and the Caribbean,Grenada,Baby
Food,Online,M,6/23/2013,641018617,6/30/2013,508,255.28,159.42,129682.24,8098
5.36,48696.88\r\nEurope,Luxembourg,Meat,Online,C,8/3/2011,775278842,9/22/201
1,1093,421.89,364.69,461125.77,398606.17,62519.60\r\nSub-Saharan
Africa,Zimbabwe,Meat,Offline,M,10/30/2016,855445134,12/6/2016,4080,421.89,36
4.69,1721311.20,1487935.20,233376.00\r\nAsia,China,Vegetables,Online,H,9/5/2
010,737816321,9/21/2010,5100,154.06,90.93,785706.00,463743.00,321963.00\r\nN
orth America,United States of
America,Beverages,Online,L,7/13/2013,799003732,7/14/2013,1815,47.45,31.79,86
121.75,57698.85,28422.90\r\nSub-Saharan Africa,Central African
Republic,Vegetables,Online,L,10/9/2012,585931193,11/21/2012,8916,154.06,90.9
3,1373598.96,810731.88,562867.08\r\nCentral America and the
Caribbean,Antigua and Barbuda
,Vegetables,Online,M,1/3/2012,165835034,1/5/2012,3127,154.06,90.93,481745.62
,284338.11,197407.51\r\nCentral America and the Caribbean,Guatemala,Baby
Food,Offline,H,2/8/2012,576264083,3/14/2012,8203,255.28,159.42,2094061.84,13
07722.26,786339.58\r\nMiddle East and North
Africa,Qatar,Clothes,Offline,L,1/3/2015,675079667,1/7/2015,9930,109.28,35.84
,1085150.40,355891.20,729259.20\r\nMiddle East and North
Africa,Israel,Personal
Care,Online,L,5/13/2012,290455615,5/28/2012,1126,81.73,56.67,92027.98,63810.
42,28217.56\r\nAsia,Thailand,Snacks,Offline,L,1/13/2012,670878255,2/15/2012,
6639,152.58,97.44,1012978.62,646904.16,366074.46\r\nAsia,Singapore,Cereal,On
line,M,7/30/2011,435146415,8/12/2011,8349,205.70,117.11,1717389.30,977751.39
,739637.91\r\nAsia,North
Korea,Snacks,Online,C,3/13/2017,522371423,4/8/2017,167,152.58,97.44,25480.86
,16272.48,9208.38\r\nEurope,Austria,Office
Supplies,Online,L,3/23/2017,141977107,4/18/2017,3036,651.21,524.96,1977073.5
6,1593778.56,383295.00\r\nAsia,Japan,Baby
Food,Offline,L,3/16/2016,823699796,4/19/2016,9929,255.28,159.42,2534675.12,1
582881.18,951793.94\r\nSub-Saharan
Africa,Zimbabwe,Beverages,Offline,L,12/18/2015,567588317,12/26/2015,851,47.4
5,31.79,40379.95,27053.29,13326.66\r\nEurope,Lithuania,Fruits,Offline,H,10/2
5/2011,594003999,11/16/2011,7838,9.33,6.92,73128.54,54238.96,18889.58\r\nEur
ope,Luxembourg,Baby
Food,Offline,H,6/30/2011,393620669,8/2/2011,9958,255.28,159.42,2542078.24,15
87504.36,954573.88\r\nSub-Saharan Africa,Central African

```

Republic, Cosmetics, Offline, H, 3/27/2016, 877424657, 4/10/2016, 8309, 437.20, 263.33, 3632694.80, 2188008.97, 1444685.83\r\nEurope, Norway, Household, Offline, M, 12/23/2016, 326714789, 1/21/2017, 1021, 668.27, 502.54, 682303.67, 513093.34, 169210.33\r\nSub-Saharan Africa, Democratic Republic of the Congo, Fruits, Offline, M, 4/18/2013, 243102395, 4/18/2013, 8256, 9.33, 6.92, 77028.48, 57131.52, 19896.96\r\nAustralia and Oceania, New Zealand, Household, Offline, L, 3/8/2017, 398511302, 4/20/2017, 7205, 668.27, 502.54, 4814885.35, 3620800.70, 1194084.65\r\nEurope, Ukraine, Personal Care, Offline, M, 6/3/2011, 185177838, 7/4/2011, 7092, 81.73, 56.67, 579629.16, 401903.64, 177725.52\r\nAsia, Taiwan, Personal Care, Offline, H, 11/30/2013, 865650832, 12/31/2013, 4173, 81.73, 56.67, 341059.29, 236483.91, 104575.38\r\nEurope, Italy, Cereal, Offline, C, 5/11/2013, 622791612, 5/31/2013, 6733, 205.70, 117.11, 1384978.10, 788501.63, 596476.47\r\nEurope, Finland, Personal Care, Online, L, 11/13/2010, 409774005, 11/27/2010, 89, 81.73, 56.67, 7273.97, 5043.63, 2230.34\r\nSub-Saharan Africa, Sudan, Office Supplies, Online, H, 3/9/2016, 800084340, 4/21/2016, 1591, 651.21, 524.96, 1036075.11, 835211.36, 200863.75\r\nEurope, Croatia, Snacks, Offline, M, 8/19/2013, 637521445, 9/12/2013, 5618, 152.58, 97.44, 857194.44, 547417.92, 309776.52\r\nSub-Saharan Africa, Mauritania, Beverages, Online, L, 4/19/2011, 186196649, 5/28/2011, 8581, 47.45, 31.79, 407168.45, 272789.99, 134378.46\r\nAustralia and Oceania, New Zealand, Baby Food, Offline, L, 7/20/2014, 680533778, 7/25/2014, 3923, 255.28, 159.42, 1001463.44, 625404.66, 376058.78\r\nMiddle East and North Africa, Pakistan, Beverages, Online, L, 9/8/2014, 275269162, 9/15/2014, 7117, 47.45, 31.79, 337701.65, 226249.43, 111452.22\r\nEurope, Poland, Household, Online, C, 6/4/2015, 795451629, 6/19/2015, 668, 668.27, 502.54, 446404.36, 335696.72, 110707.64\r\nEurope, Lithuania, Cereal, Offline, M, 12/19/2013, 986442506, 1/1/2014, 9113, 205.70, 117.11, 1874544.10, 1067223.43, 807320.67\r\nMiddle East and North Africa, Pakistan, Cereal, Offline, M, 5/18/2012, 563915622, 6/10/2012, 4019, 205.70, 117.11, 826708.30, 470665.09, 356043.21\r\nAustralia and Oceania, East Timor, Cosmetics, Offline, C, 10/12/2013, 663857305, 11/13/2013, 8984, 437.20, 263.33, 3927804.80, 2365756.72, 1562048.08\r\nAustralia and Oceania, Marshall Islands, Fruits, Online, L, 1/2/2011, 692566382, 1/14/2011, 4638, 9.33, 6.92, 43272.54, 32094.96, 11177.58\r\nCentral America and the Caribbean, Cuba, Household, Offline, L, 2/2/2013, 576654183, 2/3/2013, 3642, 668.27, 502.54, 2433839.34, 1830250.68, 603588.66\r\nNorth America, Greenland, Baby Food, Offline, H, 3/19/2011, 313044536, 4/14/2011, 5689, 255.28, 159.42, 1452287.92, 906940.38, 545347.54\r\nEurope, Luxembourg, Personal Care, Offline, C, 10/3/2012, 418973767, 10/27/2012, 2503, 81.73, 56.67, 204570.19, 141845.01, 62725.18\r\nMiddle East and North Africa, Israel, Beverages, Online, C, 10/23/2014, 581990706, 11/15/2014, 2838, 47.45, 31.79, 134663.10, 90220.02, 44443.08\r\nSub-Saharan Africa, Djibouti, Baby Food, Online, H, 7/12/2015, 109956681, 7/24/2015, 7480, 255.28, 159.42, 1909494.40, 1192461.60, 717032.80\r\nEurope, Bulgaria, Cereal, Online, L, 5/6/2010, 181045520, 5/27/2010, 4247, 205.70, 117.11, 873607.90, 497366.17, 376241.73\r\nAsia, Mongolia, Vegetables, Online, C, 11/27/2010, 693743550, 1/9/2011, 2988, 154.06, 90.93, 460331.28, 271698.84, 188632.44\r\nCentral America and the Caribbean, Dominican Republic, Clothes, Offline, L, 12/26/2010, 716849601, 12/31/2010, 582, 109.28, 35.84, 63600.96, 20858.88, 42742.08\r\nMiddle East and North Africa, Yemen, Cosmetics, Online, L, 12/30/2012, 739474999, 1/1/2013, 5940, 437.20, 263.33, 2596968.00, 1564180.20, 1032787.80\r\nAustralia and Oceania, Federated States of Micronesia, Personal Care, Online, M, 8/21/2016, 421043574, 9/7/2016, 5005, 81.73, 56.67, 409058.65, 283633.35, 125425.30\r\nEurope, Finland, Personal Care, Online, M, 2/3/2015, 841291654, 3/20/2015, 5751, 81.73, 56.67, 470029.23, 325909.17, 144120.06\r\nCentral America and the Caribbean, The

```

Bahamas,Cereal,Offline,L,12/19/2013,450268065,1/4/2014,3181,205.70,117.11,65
4331.70,372526.91,281804.79\r\nCentral America and the
Caribbean,Grenada,Meat,Online,L,5/12/2012,918334138,6/12/2012,4334,421.89,36
4.69,1828471.26,1580566.46,247904.80\r\nSub-Saharan Africa,Sao Tome and
Principe,Meat,Offline,M,3/28/2014,386163699,4/19/2014,3275,421.89,364.69,138
1689.75,1194359.75,187330.00\r\nCentral America and the Caribbean,El
Salvador,Personal
Care,Offline,C,1/11/2017,214743077,2/18/2017,6103,81.73,56.67,498798.19,3458
57.01,152941.18\r\nEurope,Sweden,Baby
Food,Online,M,6/21/2015,935371100,7/6/2015,5949,255.28,159.42,1518660.72,948
389.58,570271.14\r\nAsia,Turkmenistan,Cosmetics,Offline,H,11/29/2012,8996590
97,12/3/2012,7974,437.20,263.33,3486232.80,2099793.42,1386439.38\r\nEurope,M
onaco,Vegetables,Online,H,1/1/2010,329530894,2/13/2010,4369,154.06,90.93,673
088.14,397273.17,275814.97\r\nMiddle East and North
Africa,Turkey,Meat,Online,H,1/6/2016,867222821,2/6/2016,9359,421.89,364.69,3
948468.51,3413133.71,535334.80\r\nSub-Saharan
Africa,Mozambique,Beverages,Online,M,10/14/2014,625283706,10/23/2014,4199,47
.45,31.79,199242.55,133486.21,65756.34\r\nMiddle East and North
Africa,Yemen,Office
Supplies,Offline,C,12/9/2013,936574876,1/9/2014,2173,651.21,524.96,1415079.3
3,1140738.08,274341.25\r\nAsia,Philippines,Cereal,Offline,M,10/19/2010,50427
0160,11/25/2010,3601,205.70,117.11,740725.70,421713.11,319012.59\r\nSub-
Saharan Africa,Democratic Republic of the Congo,Personal
Care,Offline,M,5/2/2011,351855885,6/2/2011,830,81.73,56.67,67835.90,47036.10
,20799.80\r\nAustralia and
Oceania,Fiji,Snacks,Online,H,3/17/2011,673130881,3/23/2011,3241,152.58,97.44
,494511.78,315803.04,178708.74\r\nEurope,Macedonia,Office
Supplies,Offline,H,9/24/2014,382206475,10/13/2014,2244,651.21,524.96,1461315
.24,1178010.24,283305.00\r\nMiddle East and North Africa,Tunisia
,Cereal,Online,H,12/1/2015,263506495,12/14/2015,6283,205.70,117.11,1292413.1
0,735802.13,556610.97\r\nEurope,Liechtenstein,Cereal,Offline,L,6/2/2017,7217
67270,7/18/2017,5829,205.70,117.11,1199025.30,682634.19,516391.11\r\nMiddle
East and North
Africa,Qatar,Cosmetics,Online,M,8/4/2011,432037627,8/18/2011,8390,437.20,263
.33,3668108.00,2209338.70,1458769.30\r\nAustralia and
Oceania,Tonga,Meat,Offline,C,7/29/2014,389678895,8/24/2014,3499,421.89,364.6
9,1476193.11,1276050.31,200142.80\r\nEurope,Belgium,Meat,Online,L,1/26/2017,
760364902,2/24/2017,7726,421.89,364.69,3259522.14,2817594.94,441927.20\r\nSu
b-Saharan
Africa,Chad,Fruits,Offline,C,11/5/2010,430081975,12/9/2010,9669,9.33,6.92,90
211.77,66909.48,23302.29\r\nAsia,Thailand,Baby
Food,Online,H,3/26/2015,155128943,5/5/2015,4957,255.28,159.42,1265422.96,790
244.94,475178.02\r\nEurope,Iceland,Clothes,Offline,H,10/7/2012,312117135,10/
16/2012,1251,109.28,35.84,136709.28,44835.84,91873.44\r\nCentral America and
the Caribbean,Saint
Lucia,Meat,Offline,L,7/22/2013,447970378,9/2/2013,3245,421.89,364.69,1369033
.05,1183419.05,185614.00\r\nAsia,Japan,Cosmetics,Offline,C,8/16/2013,6299250
00,8/18/2013,7661,437.20,263.33,3349389.20,2017371.13,1332018.07\r\nAsia,Ind
ia,Personal
Care,Offline,L,11/5/2013,995529830,12/17/2013,8254,81.73,56.67,674599.42,467
754.18,206845.24\r\nAustralia and Oceania,Vanuatu,Office
Supplies,Online,L,3/1/2016,402646195,3/28/2016,812,651.21,524.96,528782.52,4
26267.52,102515.00\r\nSub-Saharan
Africa,Chad,Meat,Offline,M,3/25/2012,479447925,4/4/2012,8150,421.89,364.69,3
438403.50,2972223.50,466180.00\r\nSub-Saharan
Africa,Malawi,Cosmetics,Offline,H,6/20/2017,674421346,6/29/2017,5118,437.20,
263.33,2237589.60,1347722.94,889866.66\r\nEurope,Finland,Cosmetics,Online,L,

```

```

4/18/2014,506365287,5/16/2014,3596,437.20,263.33,1572171.20,946934.68,625236
.52\r\nMiddle East and North
Africa,Turkey,Clothes,Online,C,6/28/2016,914391076,8/4/2016,7494,109.28,35.8
4,818944.32,268584.96,550359.36\r\nSub-Saharan Africa,South
Africa,Meat,Online,L,5/20/2012,207922542,7/1/2012,7755,421.89,364.69,3271756
.95,2828170.95,443586.00\r\nEurope,Lithuania,Office
Supplies,Offline,M,2/4/2014,816696012,2/16/2014,7353,651.21,524.96,4788347.1
3,3860030.88,928316.25\r\nEurope,Russia,Beverages,Offline,L,11/15/2015,74076
0314,11/21/2015,6293,47.45,31.79,298602.85,200054.47,98548.38\r\nCentral
America and the Caribbean,The Bahamas,Baby
Food,Online,C,1/11/2013,300476777,2/28/2013,6610,255.28,159.42,1687400.80,10
53766.20,633634.60\r\nCentral America and the Caribbean,The
Bahamas,Snacks,Online,L,4/24/2013,786519229,6/7/2013,7373,152.58,97.44,11249
72.34,718425.12,406547.22\r\nMiddle East and North
Africa,Turkey,Cosmetics,Offline,C,5/15/2010,409873998,6/3/2010,9679,437.20,2
63.33,4231658.80,2548771.07,1682887.73\r\nSub-Saharan Africa,Mauritius
,Cosmetics,Offline,H,4/17/2010,151839911,5/22/2010,1659,437.20,263.33,725314
.80,436864.47,288450.33\r\nEurope,Bulgaria,Vegetables,Online,H,8/25/2012,614
028298,9/9/2012,3473,154.06,90.93,535050.38,315799.89,219250.49\r\nMiddle
East and North
Africa,Iran,Household,Offline,L,4/19/2014,668362987,5/13/2014,2315,668.27,50
2.54,1547045.05,1163380.10,383664.95\r\nSub-Saharan
Africa,Ghana,Household,Online,M,2/26/2013,607080304,4/5/2013,7408,668.27,502
.54,4950544.16,3722816.32,1227727.84\r\nSub-Saharan Africa,Malawi,Baby
Food,Online,M,12/28/2011,792729079,1/17/2012,5006,255.28,159.42,1277931.68,7
98056.52,479875.16\r\nSub-Saharan Africa,Zimbabwe,Baby
Food,Offline,M,8/21/2014,308170640,10/10/2014,3395,255.28,159.42,866675.60,5
41230.90,325444.70\r\nAsia,Tajikistan,Vegetables,Online,H,8/21/2014,10657881
4,10/2/2014,7894,154.06,90.93,1216149.64,717801.42,498348.22\r\nEurope,Czech
Republic,Cereal,Online,H,3/2/2014,761439931,3/28/2014,5851,205.70,117.11,120
3550.70,685210.61,518340.09\r\nSub-Saharan Africa,Mauritius
,Household,Online,L,9/6/2012,216552817,9/27/2012,1646,668.27,502.54,1099972.
42,827180.84,272791.58\r\nSub-Saharan
Africa,Lesotho,Fruits,Online,H,9/16/2010,536028802,9/22/2010,1689,9.33,6.92,
15758.37,11687.88,4070.49\r\nSub-Saharan
Africa,Mali,Beverages,Online,L,8/13/2013,254291713,8/15/2013,9424,47.45,31.7
9,447168.80,299588.96,147579.84\r\nEurope,Georgia,Personal
Care,Online,C,9/11/2012,226077878,10/23/2012,323,81.73,56.67,26398.79,18304.
41,8094.38\r\nEurope,Albania,Office
Supplies,Offline,M,8/31/2011,476436126,10/15/2011,6892,651.21,524.96,4488139
.32,3618024.32,870115.00\r\nEurope,Cyprus,Cosmetics,Offline,L,4/21/2015,6507
27784,6/6/2015,3667,437.20,263.33,1603212.40,965631.11,637581.29\r\nCentral
America and the Caribbean,Saint Kitts and Nevis
,Household,Offline,C,7/11/2010,464626681,7/27/2010,2215,668.27,502.54,148021
8.05,1113126.10,367091.95\r\nMiddle East and North Africa,Tunisia
,Meat,Offline,C,8/9/2015,154119145,9/21/2015,6135,421.89,364.69,2588295.15,2
237373.15,350922.00\r\nCentral America and the
Caribbean,Cuba,Meat,Online,M,10/17/2015,925504004,12/6/2015,6057,421.89,364.
69,2555387.73,2208927.33,346460.40\r\nAustralia and
Oceania,Kiribati,Cereal,Offline,L,7/24/2012,905392587,8/16/2012,4641,205.70,
117.11,954653.70,543507.51,411146.19\r\nAsia,Cambodia,Snacks,Online,C,3/25/2
012,990708720,5/4/2012,1581,152.58,97.44,241228.98,154052.64,87176.34\r\nEur
ope,Moldova ,Baby
Food,Online,M,8/11/2014,798688733,9/18/2014,8600,255.28,159.42,2195408.00,13
71012.00,824396.00\r\nAsia,Uzbekistan,Cereal,Offline,C,10/15/2016,916881453,
11/28/2016,4452,205.70,117.11,915776.40,521373.72,394402.68\r\nAsia,India,Co
smetics,Offline,L,12/3/2016,653148210,1/21/2017,9924,437.20,263.33,4338772.8

```

```

0,2613286.92,1725485.88\r\nEurope,Germany,Personal
Care,Offline,C,6/12/2010,285662829,7/13/2010,2834,81.73,56.67,231622.82,1606
02.78,71020.04\r\nEurope,Austria,Vegetables,Online,C,7/29/2016,612911641,8/3
1/2016,3030,154.06,90.93,466801.80,275517.90,191283.90\r\nEurope,Germany,Off
ice
Supplies,Offline,L,12/9/2013,703693473,1/12/2014,7391,651.21,524.96,4813093.
11,3879979.36,933113.75\r\nAsia,Bhutan,Clothes,Online,M,7/12/2012,147119653,
8/9/2012,4829,109.28,35.84,527713.12,173071.36,354641.76\r\nAsia,Kyrgyzstan,
Fruits,Online,C,1/14/2012,402614009,3/2/2012,1287,9.33,6.92,12007.71,8906.04
,3101.67\r\nMiddle East and North Africa,Somalia,Personal
Care,Online,C,3/20/2013,749912869,4/25/2013,4738,81.73,56.67,387236.74,26850
2.46,118734.28\r\nCentral America and the Caribbean,Saint
Lucia,Household,Online,H,2/10/2014,539065062,3/10/2014,186,668.27,502.54,124
298.22,93472.44,30825.78\r\nEurope,Armenia,Snacks,Offline,H,10/26/2013,54043
1916,11/15/2013,4668,152.58,97.44,712243.44,454849.92,257393.52\r\nNorth
America,Canada,Beverages,Online,C,4/16/2016,694687259,6/2/2016,2252,47.45,31
.79,106857.40,71591.08,35266.32\r\nSub-Saharan
Africa,Burundi,Cosmetics,Offline,H,5/27/2011,562817418,6/2/2011,9036,437.20,
263.33,3950539.20,2379449.88,1571089.32\r\nEurope,Liechtenstein,Cereal,Onlin
e,L,8/4/2016,676121222,9/9/2016,8149,205.70,117.11,1676249.30,954329.39,7219
19.91\r\nMiddle East and North Africa,Tunisia
,Snacks,Offline,L,7/3/2012,286210000,8/5/2012,4754,152.58,97.44,725365.32,46
3229.76,262135.56\r\nMiddle East and North
Africa,Iraq,Beverages,Online,L,12/4/2014,515007579,1/11/2015,1042,47.45,31.7
9,49442.90,33125.18,16317.72\r\nAsia,Indonesia,Cosmetics,Offline,C,5/31/2010
,304750287,6/1/2010,1237,437.20,263.33,540816.40,325739.21,215077.19\r\nAsia
,Kazakhstan,Beverages,Online,H,2/5/2013,467986953,2/17/2013,6594,47.45,31.79
,312885.30,209623.26,103262.04\r\nEurope,Denmark,Beverages,Offline,C,6/15/20
15,537578904,7/9/2015,399,47.45,31.79,18932.55,12684.21,6248.34\r\nEurope,Lu
xembourg,Vegetables,Offline,L,10/21/2014,116699969,11/18/2014,2969,154.06,90
.93,457404.14,269971.17,187432.97\r\nSub-Saharan Africa,Cape
Verde,Snacks,Offline,C,2/13/2017,228836476,3/13/2017,6653,152.58,97.44,10151
14.74,648268.32,366846.42\r\nAustralia and
Oceania,Palau,Vegetables,Offline,C,6/29/2010,167787253,7/16/2010,832,154.06,
90.93,128177.92,75653.76,52524.16\r\nAustralia and
Oceania,Australia,Vegetables,Online,M,5/19/2014,647663629,5/20/2014,6915,154
.06,90.93,1065324.90,628780.95,436543.95\r\nCentral America and the
Caribbean,Nicaragua,Meat,Offline,L,3/21/2015,652889430,4/15/2015,3346,421.89
,364.69,1411643.94,1220252.74,191391.20\r\nAsia,Laos,Office
Supplies,Offline,M,8/31/2015,588200986,10/15/2015,598,651.21,524.96,389423.5
8,313926.08,75497.50\r\nCentral America and the Caribbean,Cuba,Personal
Care,Online,L,8/23/2015,928647124,8/30/2015,6176,81.73,56.67,504764.48,34999
3.92,154770.56\r\nEurope,Moldova
,Cosmetics,Offline,L,2/24/2016,869589173,3/17/2016,9615,437.20,263.33,420367
8.00,2531917.95,1671760.05\r\nMiddle East and North
Africa,Syria,Household,Online,C,6/17/2015,576700961,7/23/2015,7485,668.27,50
2.54,5002000.95,3761511.90,1240489.05\r\nCentral America and the
Caribbean,The
Bahamas,Cereal,Offline,M,11/27/2012,735968816,12/6/2012,8382,205.70,117.11,1
724177.40,981616.02,742561.38\r\nEurope,Belarus,Snacks,Offline,M,9/7/2012,30
3691565,10/19/2012,7938,152.58,97.44,1211180.04,773478.72,437701.32\r\nMiddl
e East and North Africa,United Arab
Emirates,Clothes,Offline,C,6/23/2012,556480538,8/7/2012,3812,109.28,35.84,41
6575.36,136622.08,279953.28\r\nSub-Saharan
Africa,Angola,Beverages,Offline,H,10/23/2014,141259562,11/11/2014,698,47.45,
31.79,33120.10,22189.42,10930.68\r\nCentral America and the
Caribbean,Cuba,Cosmetics,Offline,C,9/18/2015,925264966,10/18/2015,5320,437.2

```



```

0,263.33,2325904.00,1400915.60,924988.40\r\nEurope,Ukraine,Office
Supplies,Online,H,1/24/2016,346045577,2/20/2016,1431,651.21,524.96,931881.51
,751217.76,180663.75\r\nSub-Saharan
Africa,Mozambique,Fruits,Offline,H,3/23/2010,861462724,4/19/2010,4818,9.33,6
.92,44951.94,33340.56,11611.38\r\nEurope,Armenia,Personal
Care,Online,M,7/26/2012,499690234,8/28/2012,8299,81.73,56.67,678277.27,47030
4.33,207972.94\r\nNorth
America,Greenland,Clothes,Online,H,10/20/2015,509214437,11/2/2015,6722,109.2
8,35.84,734580.16,240916.48,493663.68\r\nCentral America and the
Caribbean,Saint Kitts and Nevis ,Office
Supplies,Online,M,6/27/2017,408834159,7/18/2017,1968,651.21,524.96,1281581.2
8,1033121.28,248460.00\r\nEurope,Vatican
City,Beverages,Offline,M,4/4/2016,237660729,4/30/2016,7946,47.45,31.79,37703
7.70,252603.34,124434.36\r\nEurope,Ukraine,Clothes,Online,C,8/5/2015,1051179
76,9/9/2015,5600,109.28,35.84,611968.00,200704.00,411264.00\r\nSub-Saharan
Africa,Niger,Cereal,Offline,L,6/16/2012,640942227,7/4/2012,7903,205.70,117.1
1,1625647.10,925520.33,700126.77\r\nAsia,Myanmar,Cosmetics,Online,L,4/3/2016
,745182311,5/5/2016,4860,437.20,263.33,2124792.00,1279783.80,845008.20\r\nSu
b-Saharan Africa,Guinea,Baby
Food,Offline,C,9/21/2014,738199555,9/21/2014,8508,255.28,159.42,2171922.24,1
356345.36,815576.88\r\nSub-Saharan Africa,Guinea-
Bissau,Snacks,Online,L,8/8/2015,110667788,9/10/2015,7913,152.58,97.44,120736
5.54,771042.72,436322.82\r\nSub-Saharan Africa,South Sudan,Office
Supplies,Online,C,7/19/2012,673573338,7/20/2012,4174,651.21,524.96,2718150.5
4,2191183.04,526967.50\r\nMiddle East and North
Africa,Turkey,Snacks,Offline,H,8/26/2011,708215034,9/13/2011,5421,152.58,97.
44,827136.18,528222.24,298913.94\r\nAustralia and
Oceania,Palau,Household,Online,M,6/23/2012,816204202,7/1/2012,1816,668.27,50
2.54,1213578.32,912612.64,300965.68\r\nEurope,Poland,Beverages,Offline,L,11/
20/2011,769464671,12/24/2011,550,47.45,31.79,26097.50,17484.50,8613.00\r\nAs
ia,Malaysia,Beverages,Offline,L,5/13/2015,860232770,6/4/2015,848,47.45,31.79
,40237.60,26957.92,13279.68\r\nNorth America,United States of
America,Personal
Care,Offline,C,7/16/2010,551057326,8/22/2010,8963,81.73,56.67,732545.99,5079
33.21,224612.78\r\nEurope,Switzerland,Cosmetics,Online,C,5/21/2016,724799668
,5/27/2016,3183,437.20,263.33,1391607.60,838179.39,553428.21\r\nAustralia
and Oceania,Papua New
Guinea,Cosmetics,Offline,H,4/22/2011,534633624,6/8/2011,8825,437.20,263.33,3
858290.00,2323887.25,1534402.75\r\nSub-Saharan
Africa,Namibia,Beverages,Offline,H,8/14/2012,554045522,9/20/2012,3237,47.45,
31.79,153595.65,102904.23,50691.42\r\nEurope,Ireland,Clothes,Online,C,3/17/2
012,526834189,5/2/2012,799,109.28,35.84,87314.72,28636.16,58678.56\r\nSub-
Saharan
Africa,Mozambique,Household,Online,C,8/6/2010,717110955,8/9/2010,7922,668.27
,502.54,5294034.94,3981121.88,1312913.06\r\nSub-Saharan Africa,Democratic
Republic of the Congo,Baby
Food,Offline,L,2/25/2013,559299647,3/26/2013,8049,255.28,159.42,2054748.72,1
283171.58,771577.14\r\nNorth America,United States of
America,Meat,Online,M,3/6/2011,908136594,3/10/2011,6654,421.89,364.69,280725
6.06,2426647.26,380608.80\r\nMiddle East and North Africa,Azerbaijan,Office
Supplies,Offline,M,12/14/2015,888670623,12/16/2015,6240,651.21,524.96,406355
0.40,3275750.40,787800.00\r\nEurope,Belgium,Office
Supplies,Offline,C,2/10/2017,146263062,2/16/2017,1345,651.21,524.96,875877.4
5,706071.20,169806.25\r\nAsia,Taiwan,Office
Supplies,Offline,L,9/20/2016,196587741,10/28/2016,3536,651.21,524.96,2302678
.56,1856258.56,446420.00\r\nCentral America and the Caribbean,Panama,Baby
Food,Online,M,4/30/2010,375630986,6/2/2010,6411,255.28,159.42,1636600.08,102

```

2041.62,614558.46\r\nEurope,Andorra,Beverages,Offline,H,6/25/2013,989691627,7/10/2013,600,47.45,31.79,28470.00,19074.00,9396.00\r\nEurope,Georgia,Household,Offline,H,7/4/2012,165380990,7/27/2012,8765,668.27,502.54,5857386.55,4404763.10,1452623.45\r\nCentral America and the Caribbean,Barbados,Snacks,Online,C,9/21/2013,599622905,10/22/2013,597,152.58,97.44,91090.26,58171.68,32918.58\r\nEurope,Sweden,Personal Care,Offline,C,12/8/2016,109653699,1/6/2017,7821,81.73,56.67,639210.33,443216.07,195994.26\r\nMiddle East and North Africa,Algeria,Meat,Offline,M,9/2/2011,183022201,10/15/2011,9191,421.89,364.69,3877590.99,3351865.79,525725.20\r\nEurope,Italy,Personal Care,Online,L,3/21/2011,127589738,4/2/2011,5494,81.73,56.67,449024.62,311344.98,137679.64\r\nEurope,Russia,Fruits,Offline,L,1/8/2011,221530139,1/26/2011,4546,9.33,6.92,42414.18,31458.32,10955.86\r\nCentral America and the Caribbean,Antigua and Barbuda ,Office Supplies,Offline,M,2/22/2015,363329732,2/22/2015,6197,651.21,524.96,4035548.37,3253177.12,782371.25\r\nMiddle East and North Africa,Jordan,Fruits,Online,C,5/15/2017,521787345,6/25/2017,7325,9.33,6.92,68342.25,50689.00,17653.25\r\nSub-Saharan Africa,Mali,Meat,Online,L,7/14/2012,286014306,8/15/2012,6844,421.89,364.69,2887415.16,2495938.36,391476.80\r\nMiddle East and North Africa,Somalia,Cereal,Offline,C,6/25/2015,215434443,6/30/2015,694,205.70,117.11,142755.80,81274.34,61481.46\r\nMiddle East and North Africa,Kuwait,Snacks,Online,L,10/26/2011,489784085,11/1/2011,6850,152.58,97.44,1045173.00,667464.00,377709.00\r\nSub-Saharan Africa,Liberia,Office Supplies,Offline,C,9/24/2014,459112060,10/12/2014,316,651.21,524.96,205782.36,165887.36,39895.00\r\nAsia,China,Office Supplies,Online,C,9/30/2015,893779695,11/7/2015,8128,651.21,524.96,5293034.88,4266874.88,1026160.00\r\nEurope,Andorra,Meat,Offline,M,3/31/2011,834460818,3/31/2011,4355,421.89,364.69,1837330.95,1588224.95,249106.00\r\nSub-Saharan Africa,Niger,Beverages,Online,C,10/25/2013,742141759,10/28/2013,5093,47.45,31.79,241662.85,161906.47,79756.38\r\nEurope,Hungary,Vegetables,Offline,M,8/12/2010,248121345,9/14/2010,3475,154.06,90.93,535358.50,315981.75,219376.75\r\nEurope,Monaco,Clothes,Offline,M,7/26/2012,404010903,9/4/2012,4659,109.28,35.84,509135.52,166978.56,342156.96\r\nAustralia and Oceania,Tuvalu,Household,Online,L,9/30/2012,531734263,10/12/2012,840,668.27,502.54,561346.80,422133.60,139213.20\r\nSub-Saharan Africa,South Sudan,Baby Food,Online,C,11/9/2012,473527753,12/29/2012,6240,255.28,159.42,1592947.20,994780.80,598166.40\r\nEurope,Cyprus,Cereal,Offline,M,11/19/2011,141940200,1/2/2012,2114,205.70,117.11,434849.80,247570.54,187279.26\r\nEurope,Poland,Household,Offline,M,6/9/2017,869832932,7/25/2017,1749,668.27,502.54,1168804.23,878942.46,289861.77\r\nSub-Saharan Africa,Liberia,Snacks,Online,H,9/17/2011,460379779,11/4/2011,5462,152.58,97.44,833391.96,532217.28,301174.68\r\nSub-Saharan Africa,Cote d'Ivoire,Vegetables,Online,C,10/23/2015,837067067,10/26/2015,5602,154.06,90.93,863044.12,509389.86,353654.26\r\nAsia,Kyrgyzstan,Beverages,Online,C,4/3/2010,393693625,4/9/2010,1547,47.45,31.79,73405.15,49179.13,24226.02\r\nEurope,Slovakia,Vegetables,Offline,L,7/3/2014,744370782,7/14/2014,4711,154.06,90.93,725776.66,428371.23,297405.43\r\nAsia,Malaysia,Cosmetics,Offline,M,12/30/2014,873522365,1/13/2015,3534,437.20,263.33,1545064.80,930608.22,614456.58\r\nSub-Saharan Africa,Liberia,Beverages,Online,L,5/19/2014,285884702,6/10/2014,8491,47.45,31.79,402897.95,269928.89,132969.06\r\nAustralia and Oceania,Vanuatu,Cosmetics,Online,H,8/4/2012,356506621,9/3/2012,7086,437.20,263.33,3097999.20,1865956.38,1232042.82\r\nAustralia and Oceania,Kiribati,Baby Food,Offline,L,8/15/2010,280749452,10/1/2010,8856,255.28,159.42,2260759.68,1411823.52,848936.16\r\nMiddle East and North Africa,Turkey,Baby

```

Food,Online,M,4/26/2014,224287021,5/17/2014,368,255.28,159.42,93943.04,58666
.56,35276.48\r\nEurope,San
Marino,Fruits,Offline,M,6/10/2015,873105657,6/23/2015,221,9.33,6.92,2061.93,
1529.32,532.61\r\nEurope,Vatican
City,Snacks,Offline,C,8/12/2010,283504188,9/2/2010,4044,152.58,97.44,617033.
52,394047.36,222986.16\r\nMiddle East and North
Africa,Morocco,Beverages,Offline,C,8/28/2012,632093942,9/5/2012,9499,47.45,3
1.79,450727.55,301973.21,148754.34\r\nSub-Saharan Africa,Equatorial
Guinea,Meat,Offline,L,7/15/2016,565798747,8/9/2016,1277,421.89,364.69,538753
.53,465709.13,73044.40\r\nMiddle East and North
Africa,Jordan,Vegetables,Online,M,9/14/2010,151854932,10/19/2010,6104,154.06
,90.93,940382.24,555036.72,385345.52\r\nAsia,Kyrgyzstan,Vegetables,Online,H,
3/7/2011,427811324,4/16/2011,7733,154.06,90.93,1191345.98,703161.69,488184.2
9\r\nSub-Saharan Africa,Republic of the
Congo,Fruits,Online,L,4/30/2012,251529252,5/5/2012,1950,9.33,6.92,18193.50,1
3494.00,4699.50\r\nAustralia and Oceania,East
Timor,Snacks,Offline,C,5/18/2013,351182544,6/22/2013,1574,152.58,97.44,24016
0.92,153370.56,86790.36\r\nEurope,Estonia,Meat,Offline,H,8/10/2014,175257527
,9/25/2014,1452,421.89,364.69,612584.28,529529.88,83054.40\r\nAsia,Banglades
h,Snacks,Online,H,7/31/2013,142553031,9/11/2013,3465,152.58,97.44,528689.70,
337629.60,191060.10\r\nSub-Saharan
Africa,Senegal,Fruits,Offline,H,7/2/2016,292180383,8/15/2016,1523,9.33,6.92,
14209.59,10539.16,3670.43\r\nMiddle East and North
Africa,Pakistan,Fruits,Offline,L,9/13/2011,733563411,9/20/2011,6569,9.33,6.9
2,61288.77,45457.48,15831.29\r\nEurope,Czech
Republic,Beverages,Online,H,4/18/2011,296438443,4/19/2011,1578,47.45,31.79,7
4876.10,50164.62,24711.48\r\nSub-Saharan
Africa,Ghana,Meat,Offline,C,2/9/2017,580854308,3/18/2017,6552,421.89,364.69,
2764223.28,2389448.88,374774.40\r\nAsia,Japan,Cosmetics,Offline,H,11/20/2016
,107172334,12/23/2016,3530,437.20,263.33,1543316.00,929554.90,613761.10\r\nA
sia,Kazakhstan,Baby
Food,Offline,H,10/16/2010,166066348,12/5/2010,1578,255.28,159.42,402831.84,2
51564.76,151267.08\r\nCentral America and the Caribbean,The
Bahamas,Snacks,Offline,H,3/15/2015,768522679,3/27/2015,1794,152.58,97.44,273
728.52,174807.36,98921.16\r\nSub-Saharan
Africa,Ethiopia,Household,Online,M,6/25/2013,195840156,7/25/2013,2309,668.27
,502.54,1543035.43,1160364.86,382670.57\r\nSub-Saharan Africa,Burkina
Faso,Cosmetics,Offline,L,10/11/2012,849630105,11/9/2012,3284,437.20,263.33,1
435764.80,864775.72,570989.08\r\nSub-Saharan
Africa,Madagascar,Fruits,Offline,M,7/8/2017,701816356,7/30/2017,1910,9.33,6.
92,17820.30,13217.20,4603.10\r\nEurope,Netherlands,Office
Supplies,Online,M,6/22/2010,944635236,7/27/2010,7413,651.21,524.96,4827419.7
3,3891528.48,935891.25\r\nEurope,Greece,Vegetables,Online,L,2/17/2017,140635
573,3/21/2017,6046,154.06,90.93,931446.76,549762.78,381683.98\r\nMiddle East
and North
Africa,Egypt,Meat,Online,C,1/14/2011,972678697,2/25/2011,6096,421.89,364.69,
2571841.44,2223150.24,348691.20\r\nSub-Saharan Africa,South
Sudan,Fruits,Online,M,1/9/2016,793938434,2/8/2016,2880,9.33,6.92,26870.40,19
929.60,6940.80\r\nEurope,Kosovo,Baby
Food,Online,L,7/18/2011,177901113,8/13/2011,3747,255.28,159.42,956534.16,597
346.74,359187.42\r\nAsia,Brunei,Beverages,Online,L,3/16/2012,668365561,5/4/2
012,3077,47.45,31.79,146003.65,97817.83,48185.82\r\nAustralia and
Oceania,Australia,Vegetables,Online,C,12/15/2010,729443109,1/12/2011,7281,15
4.06,90.93,1121710.86,662061.33,459649.53\r\nSub-Saharan Africa,Cape
Verde,Fruits,Offline,H,3/2/2013,695557582,4/3/2013,9800,9.33,6.92,91434.00,6
7816.00,23618.00\r\nSub-Saharan
Africa,Malawi,Household,Offline,H,2/17/2010,521445310,4/4/2010,6110,668.27,5

```

```

02.54,4083129.70,3070519.40,1012610.30\r\nAsia,Philippines,Personal
Care,Online,L,2/19/2013,232155120,3/30/2013,8714,81.73,56.67,712195.22,49382
2.38,218372.84\r\nEurope,Estonia,Beverages,Online,L,10/28/2011,373048341,12/
12/2011,2149,47.45,31.79,101970.05,68316.71,33653.34\r\nCentral America and
the Caribbean,Trinidad and Tobago,Office
Supplies,Offline,L,10/26/2015,659798800,12/2/2015,7982,651.21,524.96,5197958
.22,4190230.72,1007727.50\r\nAsia,Mongolia,Household,Online,C,2/5/2013,42839
2827,2/5/2013,9812,668.27,502.54,6557065.24,4930922.48,1626142.76\r\nAsia,Ja
pan,Personal
Care,Offline,M,8/25/2011,885129249,9/3/2011,8269,81.73,56.67,675825.37,46860
4.23,207221.14\r\nSub-Saharan
Africa,Niger,Meat,Online,C,12/4/2012,156619393,12/5/2012,6014,421.89,364.69,
2537246.46,2193245.66,344000.80\r\nMiddle East and North Africa,Egypt,Baby
Food,Offline,M,9/8/2015,939787089,9/9/2015,2739,255.28,159.42,699211.92,4366
51.38,262560.54\r\nCentral America and the Caribbean,Saint
Lucia,Vegetables,Online,C,3/1/2012,151868665,4/19/2012,168,154.06,90.93,2588
2.08,15276.24,10605.84\r\nMiddle East and North
Africa,Qatar,Cereal,Offline,C,8/5/2014,180412948,8/24/2014,7055,205.70,117.1
1,1451213.50,826211.05,625002.45\r\nSub-Saharan
Africa,Mali,Fruits,Offline,H,7/24/2013,333281266,7/28/2013,4188,9.33,6.92,39
074.04,28980.96,10093.08\r\nCentral America and the Caribbean,Saint
Lucia,Cosmetics,Online,L,1/26/2012,888647449,2/28/2012,9383,437.20,263.33,41
02247.60,2470825.39,1631422.21\r\nSub-Saharan
Africa,Swaziland,Clothes,Offline,M,3/10/2014,844997823,4/26/2014,2488,109.28
,35.84,271888.64,89169.92,182718.72\r\nAsia,Mongolia,Fruits,Online,M,9/12/20
13,171131217,10/8/2013,385,9.33,6.92,3592.05,2664.20,927.85\r\nSub-Saharan
Africa,Botswana,Office
Supplies,Online,C,1/5/2013,256158959,1/18/2013,1983,651.21,524.96,1291349.43
,1040995.68,250353.75\r\nEurope,San
Marino,Cosmetics,Online,H,10/28/2011,759504878,12/8/2011,3226,437.20,263.33,
1410407.20,849502.58,560904.62\r\nMiddle East and North
Africa,Oman,Fruits,Online,C,11/21/2010,960905301,11/25/2010,2087,9.33,6.92,1
9471.71,14442.04,5029.67\r\nAsia,Bangladesh,Office
Supplies,Offline,C,10/28/2015,210409057,12/4/2015,3570,651.21,524.96,2324819
.70,1874107.20,450712.50\r\nSub-Saharan
Africa,Namibia,Cosmetics,Offline,C,7/29/2010,178377473,9/1/2010,4713,437.20,
263.33,2060523.60,1241074.29,819449.31\r\nAsia,Mongolia,Fruits,Online,M,3/30
/2014,805484378,5/1/2014,9582,9.33,6.92,89400.06,66307.44,23092.62\r\nAsia,N
orth
Korea,Beverages,Online,C,9/14/2016,752716100,10/12/2016,4276,47.45,31.79,202
896.20,135934.04,66962.16\r\nEurope,Latvia,Clothes,Online,M,9/6/2012,5513714
67,9/15/2012,1925,109.28,35.84,210364.00,68992.00,141372.00\r\nSub-Saharan
Africa,Burundi,Snacks,Offline,M,2/8/2013,353061807,3/5/2013,7689,152.58,97.4
4,1173187.62,749216.16,423971.46\r\nSub-Saharan Africa,Seychelles ,Personal
Care,Online,C,1/17/2011,379710948,1/30/2011,3762,81.73,56.67,307468.26,21319
2.54,94275.72\r\nSub-Saharan
Africa,Kenya,Cereal,Online,H,10/11/2015,473555219,11/6/2015,4368,205.70,117.
11,898497.60,511536.48,386961.12\r\nSub-Saharan Africa,Benin,Office
Supplies,Online,C,2/21/2011,547143447,2/23/2011,760,651.21,524.96,494919.60,
398969.60,95950.00\r\nCentral America and the Caribbean,Saint Lucia,Personal
Care,Online,C,4/29/2012,133336961,6/13/2012,6225,81.73,56.67,508769.25,35277
0.75,155998.50\r\nMiddle East and North
Africa,Qatar,Meat,Online,H,8/30/2016,635309588,10/14/2016,1080,421.89,364.69
,455641.20,393865.20,61776.00\r\nSub-Saharan
Africa,Mozambique,Beverages,Online,C,12/20/2014,376547658,12/26/2014,7675,47
.45,31.79,364178.75,243988.25,120190.50\r\nMiddle East and North
Africa,Pakistan,Clothes,Offline,M,6/28/2010,450849997,7/21/2010,5388,109.28,

```

```

35.84,588800.64,193105.92,395694.72\r\nAsia,Taiwan,Personal
Care,Online,M,2/6/2015,672327935,2/6/2015,5631,81.73,56.67,460221.63,319108.
77,141112.86\r\nCentral America and the
Caribbean,Cuba,Cereal,Offline,L,3/30/2015,925405299,5/1/2015,6847,205.70,117
.11,1408427.90,801852.17,606575.73\r\nCentral America and the
Caribbean,Cuba,Household,Offline,H,7/28/2013,714818418,8/24/2013,9509,668.27
,502.54,6354579.43,4778652.86,1575926.57\r\nEurope,Russia,Beverages,Offline,
C,1/6/2010,515616118,2/5/2010,1122,47.45,31.79,53238.90,35668.38,17570.52\r\
nEurope,Switzerland,Cereal,Offline,C,2/20/2013,423159730,4/11/2013,1222,205.
70,117.11,251365.40,143108.42,108256.98\r\nEurope,Czech Republic,Personal
Care,Offline,H,8/25/2013,603123080,9/29/2013,6377,81.73,56.67,521192.21,3613
84.59,159807.62\r\nEurope,Poland,Meat,Offline,C,11/16/2010,841492497,12/31/2
010,5185,421.89,364.69,2187499.65,1890917.65,296582.00\r\nSub-Saharan
Africa,Mauritius
,Cereal,Offline,L,8/2/2016,994566810,9/1/2016,3275,205.70,117.11,673667.50,3
83535.25,290132.25\r\nMiddle East and North
Africa,Pakistan,Vegetables,Offline,M,4/25/2013,538957345,4/25/2013,8310,154.
06,90.93,1280238.60,755628.30,524610.30\r\nSub-Saharan Africa,South
Africa,Fruits,Offline,L,3/2/2011,821587932,3/11/2011,4981,9.33,6.92,46472.73
,34468.52,12004.21\r\nSub-Saharan Africa,Seychelles
,Household,Online,C,9/28/2013,109694898,10/16/2013,13,668.27,502.54,8687.51,
6533.02,2154.49\r\nSub-Saharan
Africa,Benin,Meat,Offline,M,5/5/2014,340827071,6/5/2014,7159,421.89,364.69,3
020310.51,2610815.71,409494.80\r\nSub-Saharan
Africa,Benin,Meat,Offline,L,11/28/2014,372845780,12/9/2014,2207,421.89,364.6
9,931111.23,804870.83,126240.40\r\nCentral America and the
Caribbean,Nicaragua,Fruits,Online,M,8/30/2014,933924853,9/13/2014,7973,9.33,
6.92,74388.09,55173.16,19214.93\r\nMiddle East and North
Africa,Lebanon,Office
Supplies,Online,L,11/20/2013,572550618,11/25/2013,9306,651.21,524.96,6060160
.26,4885277.76,1174882.50\r\nEurope,Moldova
,Meat,Online,M,2/23/2010,607521903,4/5/2010,8086,421.89,364.69,3411402.54,29
48883.34,462519.20\r\nMiddle East and North Africa,Tunisia
,Snacks,Online,H,3/20/2017,177950036,4/29/2017,8225,152.58,97.44,1254970.50,
801444.00,453526.50\r\nAustralia and
Oceania,Vanuatu,Beverages,Offline,M,11/8/2015,293258845,11/14/2015,664,47.45
,31.79,31506.80,21108.56,10398.24\r\nSub-Saharan Africa,South
Sudan,Beverages,Online,C,8/9/2010,683184659,8/23/2010,8377,47.45,31.79,39748
8.65,266304.83,131183.82\r\nEurope,Sweden,Cereal,Online,L,11/12/2010,2477763
05,11/30/2010,1370,205.70,117.11,281809.00,160440.70,121368.30\r\nEurope,Ire
land,Meat,Offline,C,12/14/2011,207395112,1/26/2012,1677,421.89,364.69,707509
.53,611585.13,95924.40\r\nEurope,Italy,Vegetables,Offline,L,1/10/2014,952714
908,2/25/2014,8367,154.06,90.93,1289020.02,760811.31,528208.71\r\nEurope,Bos
nia and
Herzegovina,Vegetables,Online,C,10/3/2010,694722020,10/3/2010,2539,154.06,90
.93,391158.34,230871.27,160287.07\r\nEurope,Bosnia and
Herzegovina,Household,Online,H,10/15/2015,414715278,11/4/2015,2321,668.27,50
2.54,1551054.67,1166395.34,384659.33\r\nEurope,Poland,Snacks,Offline,M,7/20/
2013,714306008,8/17/2013,7876,152.58,97.44,1201720.08,767437.44,434282.64\r\
nMiddle East and North
Africa,Kuwait,Snacks,Online,H,1/19/2016,465418040,2/26/2016,6396,152.58,97.4
4,975901.68,623226.24,352675.44\r\nSub-Saharan
Africa,Sudan,Cereal,Online,M,8/17/2013,860287702,9/11/2013,7103,205.70,117.1
1,1461087.10,831832.33,629254.77\r\nMiddle East and North Africa,Saudi
Arabia,Vegetables,Online,C,8/16/2016,461463820,8/20/2016,6254,154.06,90.93,9
63491.24,568676.22,394815.02\r\nSub-Saharan Africa,Swaziland,Baby
Food,Offline,M,9/25/2013,151807725,9/29/2013,2134,255.28,159.42,544767.52,34

```

```

0202.28,204565.24\r\nSub-Saharan
Africa,Rwanda,Meat,Offline,C,8/21/2013,884493243,10/2/2013,61,421.89,364.69,
25735.29,22246.09,3489.20\r\nAsia,Cambodia,Cosmetics,Offline,M,12/10/2010,53
3006703,1/23/2011,7383,437.20,263.33,3227847.60,1944165.39,1283682.21\r\nSub
-Saharan Africa,Central African
Republic,Vegetables,Online,C,9/4/2012,641146934,10/4/2012,8480,154.06,90.93,
1306428.80,771086.40,535342.40\r\nAsia,Maldives,Cosmetics,Offline,M,10/15/20
11,573025262,11/14/2011,9764,437.20,263.33,4268820.80,2571154.12,1697666.68\r
\r\nSub-Saharan
Africa,Djibouti,Household,Offline,C,8/26/2013,663065516,9/9/2013,4676,668.27
,502.54,3124830.52,2349877.04,774953.48\r\nAsia,Tajikistan,Beverages,Online,
M,2/17/2017,866004025,3/4/2017,8691,47.45,31.79,412387.95,276286.89,136101.0
6\r\nAsia,Sri Lanka,Baby
Food,Online,H,10/19/2010,306889617,10/21/2010,4312,255.28,159.42,1100767.36,
687419.04,413348.32\r\nEurope,Montenegro,Personal
Care,Online,M,7/30/2014,431083619,8/10/2014,6077,81.73,56.67,496673.21,34438
3.59,152289.62\r\nMiddle East and North Africa,United Arab Emirates,Personal
Care,Online,H,5/21/2015,954259860,6/4/2015,5553,81.73,56.67,453846.69,314688
.51,139158.18\r\nCentral America and the Caribbean,Dominican
Republic,Personal
Care,Offline,C,5/5/2016,312404668,6/21/2016,6338,81.73,56.67,518004.74,35917
4.46,158830.28\r\nSub-Saharan Africa,Seychelles ,Office
Supplies,Offline,C,4/14/2010,611816871,5/16/2010,9063,651.21,524.96,5901916.
23,4757712.48,1144203.75\r\nEurope,Iceland,Office
Supplies,Online,C,10/26/2013,879107797,11/2/2013,6388,651.21,524.96,4159929.
48,3353444.48,806485.00\r\nSub-Saharan
Africa,Nigeria,Vegetables,Offline,C,7/21/2010,211201274,9/9/2010,8005,154.06
,90.93,1233250.30,727894.65,505355.65\r\nSub-Saharan
Africa,Rwanda,Fruits,Online,H,6/9/2015,925333631,7/25/2015,5639,9.33,6.92,52
611.87,39021.88,13589.99\r\nEurope,Hungary,Snacks,Offline,C,6/3/2010,9090536
95,6/27/2010,8044,152.58,97.44,1227353.52,783807.36,443546.16\r\nEurope,Bela
rus,Baby
Food,Online,L,5/17/2016,370222795,6/11/2016,6007,255.28,159.42,1533466.96,95
7635.94,575831.02\r\nSub-Saharan Africa,South
Sudan,Cosmetics,Offline,H,8/3/2013,487014758,8/30/2013,7344,437.20,263.33,32
10796.80,1933895.52,1276901.28\r\nEurope,Andorra,Vegetables,Online,M,9/15/20
13,257915914,10/6/2013,1905,154.06,90.93,293484.30,173221.65,120262.65\r\nAs
ia,Japan,Meat,Offline,M,7/2/2010,551725089,8/10/2010,6569,421.89,364.69,2771
395.41,2395648.61,375746.80\r\nCentral America and the Caribbean,El
Salvador,Meat,Offline,L,12/18/2013,957553613,1/10/2014,248,421.89,364.69,104
628.72,90443.12,14185.60\r\nSub-Saharan Africa,Kenya,Office
Supplies,Offline,L,3/19/2016,234825313,3/23/2016,8883,651.21,524.96,5784698.
43,4663219.68,1121478.75\r\nEurope,Bosnia and
Herzegovina,Snacks,Online,M,6/9/2010,363276517,7/9/2010,449,152.58,97.44,685
08.42,43750.56,24757.86\r\nEurope,Andorra,Personal
Care,Online,M,5/21/2017,692956054,6/23/2017,9950,81.73,56.67,813213.50,56386
6.50,249347.00\r\nSub-Saharan Africa,Cape
Verde,Cosmetics,Online,H,6/12/2013,194225251,6/19/2013,4423,437.20,263.33,19
33735.60,1164708.59,769027.01\r\nAustralia and
Oceania,Nauru,Fruits,Online,L,2/22/2010,607757937,4/5/2010,7934,9.33,6.92,74
024.22,54903.28,19120.94\r\nEurope,Czech
Republic,Cereal,Offline,H,6/29/2012,594540441,7/30/2012,6583,205.70,117.11,1
354123.10,770935.13,583187.97\r\nEurope,Serbia,Vegetables,Online,L,3/23/2015
,685871589,4/5/2015,3500,154.06,90.93,539210.00,318255.00,220955.00\r\nAustr
alia and
Oceania,Tuvalu,Cereal,Offline,C,2/12/2014,133362710,3/23/2014,3844,205.70,11
7.11,790710.80,450170.84,340539.96\r\nSub-Saharan

```

```

Africa,Madagascar,Clothes,Offline,H,5/22/2017,958937633,7/5/2017,9810,109.28
,35.84,1072036.80,351590.40,720446.40\r\nSub-Saharan
Africa,Ethiopia,Vegetables,Online,C,8/6/2011,304832684,9/6/2011,5620,154.06,
90.93,865817.20,511026.60,354790.60\r\nAsia,Malaysia,Baby
Food,Offline,L,11/14/2010,783596694,12/24/2010,2530,255.28,159.42,645858.40,
403332.60,242525.80\r\nSub-Saharan
Africa,Tanzania,Household,Offline,C,4/5/2015,128090989,4/27/2015,3825,668.27
,502.54,2556132.75,1922215.50,633917.25\r\nSub-Saharan Africa,Cote
d'Ivoire,Vegetables,Offline,M,7/8/2014,641489398,7/28/2014,9823,154.06,90.93
,1513331.38,893205.39,620125.99\r\nAustralia and Oceania,Solomon
Islands,Clothes,Offline,M,8/9/2014,647278249,9/16/2014,2873,109.28,35.84,313
961.44,102968.32,210993.12\r\nEurope,Netherlands,Clothes,Online,H,3/13/2011,
339256370,3/31/2011,2354,109.28,35.84,257245.12,84367.36,172877.76\r\nSub-
Saharan Africa,Mali,Baby
Food,Offline,M,3/2/2016,431535089,3/19/2016,9677,255.28,159.42,2470344.56,15
42707.34,927637.22\r\nMiddle East and North
Africa,Afghanistan,Cereal,Offline,C,12/19/2015,808538234,1/16/2016,3286,205.
70,117.11,675930.20,384823.46,291106.74\r\nEurope,Moldova ,Personal
Care,Online,C,3/28/2013,975002133,4/7/2013,3653,81.73,56.67,298559.69,207015
.51,91544.18\r\nAsia,Bhutan,Snacks,Online,H,6/27/2015,505975615,7/4/2015,828
3,152.58,97.44,1263820.14,807095.52,456724.62\r\nAsia,Vietnam,Vegetables,Off
line,M,2/18/2016,396820008,3/20/2016,6714,154.06,90.93,1034358.84,610504.02,
423854.82\r\nEurope,Portugal,Vegetables,Online,H,6/14/2013,813209140,7/10/20
13,5511,154.06,90.93,849024.66,501115.23,347909.43\r\nEurope,Spain,Baby
Food,Online,M,5/10/2014,641129338,5/14/2014,3273,255.28,159.42,835531.44,521
781.66,313749.78\r\nMiddle East and North
Africa,Egypt,Meat,Offline,C,6/17/2015,636879432,7/3/2015,5632,421.89,364.69,
2376084.48,2053934.08,322150.40\r\nEurope,Belgium,Snacks,Online,C,6/11/2014,
277070748,7/2/2014,246,152.58,97.44,37534.68,23970.24,13564.44\r\nAsia,Malay
sia,Cosmetics,Offline,H,11/3/2013,908627116,11/24/2013,1810,437.20,263.33,79
1332.00,476627.30,314704.70\r\nCentral America and the Caribbean,Dominican
Republic,Cosmetics,Offline,C,4/2/2017,798784863,5/2/2017,7047,437.20,263.33,
3080948.40,1855686.51,1225261.89\r\nEurope,Estonia,Beverages,Offline,H,5/28/
2010,985092818,7/17/2010,9711,47.45,31.79,460786.95,308712.69,152074.26\r\nS
ub-Saharan
Africa,Burundi,Snacks,Offline,C,5/22/2010,325412309,7/7/2010,5588,152.58,97.
44,852617.04,544494.72,308122.32\r\nEurope,Latvia,Beverages,Online,M,6/2/201
6,447917163,6/24/2016,7497,47.45,31.79,355732.65,238329.63,117403.02\r\nAsia
,Tajikistan,Meat,Offline,M,8/22/2013,801093709,10/5/2013,285,421.89,364.69,1
20238.65,103936.65,16302.00\r\nSub-Saharan
Africa,Zimbabwe,Fruits,Offline,H,10/5/2014,903740775,10/23/2014,5833,9.33,6.
92,54421.89,40364.36,14057.53\r\nSub-Saharan
Africa,Comoros,Meat,Offline,L,10/31/2010,794969689,11/13/2010,8052,421.89,36
4.69,3397058.28,2936483.88,460574.40\r\nSub-Saharan
Africa,Namibia,Clothes,Online,L,11/27/2012,584204280,1/1/2013,7884,109.28,35
.84,861563.52,282562.56,579000.96\r\nEurope,Slovenia,Cereal,Offline,H,5/22/2
010,901180875,5/26/2010,8302,205.70,117.11,1707721.40,972247.22,735474.18\r\
nEurope,Bulgaria,Snacks,Offline,L,8/31/2012,645948302,9/29/2012,9312,152.58,
97.44,1420824.96,907361.28,513463.68\r\nSub-Saharan Africa,Guinea-
Bissau,Cereal,Online,L,1/23/2015,138867890,2/22/2015,2950,205.70,117.11,6068
15.00,345474.50,261340.50\r\nSub-Saharan
Africa,Lesotho,Beverages,Offline,L,3/2/2010,670613467,3/21/2010,8282,47.45,3
1.79,392980.90,263284.78,129696.12\r\nAsia,Sri
Lanka,Cosmetics,Offline,L,5/9/2014,452171361,5/27/2014,6409,437.20,263.33,28
02014.80,1687681.97,1114332.83\r\nAustralia and Oceania,East
Timor,Snacks,Online,H,12/28/2010,464840400,2/5/2011,5459,152.58,97.44,832934
.22,531924.96,301009.26\r\nEurope,Belarus,Household,Online,M,10/19/2014,4102

```

```

31912,10/24/2014,5594,668.27,502.54,3738302.38,2811208.76,927093.62\r\nSub-
Saharan
Africa,Benin,Meat,Offline,H,1/21/2015,960269725,2/22/2015,4006,421.89,364.69
,1690091.34,1460948.14,229143.20\r\nEurope,Ireland,Beverages,Offline,L,4/4/2
017,607190167,5/18/2017,9919,47.45,31.79,470656.55,315325.01,155331.54\r\nMi
ddle East and North
Africa,Iran,Meat,Offline,H,8/5/2016,613542068,8/11/2016,9587,421.89,364.69,4
044659.43,3496283.03,548376.40\r\nSub-Saharan
Africa,Benin,Household,Offline,C,11/26/2016,962186753,1/12/2017,1297,668.27,
502.54,866746.19,651794.38,214951.81\r\nSub-Saharan Africa,South
Sudan,Beverages,Online,L,9/23/2011,806298053,10/24/2011,366,47.45,31.79,1736
6.70,11635.14,5731.56\r\nSub-Saharan Africa,Comoros,Personal
Care,Online,L,10/24/2010,719362294,12/3/2010,4144,81.73,56.67,338689.12,2348
40.48,103848.64\r\nEurope,Poland,Baby
Food,Online,H,8/14/2013,445178306,9/22/2013,7008,255.28,159.42,1789002.24,11
17215.36,671786.88\r\nEurope,Bosnia and
Herzegovina,Cosmetics,Offline,C,2/4/2013,247857415,2/15/2013,5372,437.20,263
.33,2348638.40,1414608.76,934029.64\r\nSub-Saharan
Africa,Namibia,Vegetables,Offline,C,8/11/2014,461823451,9/4/2014,2677,154.06
,90.93,412418.62,243419.61,168999.01\r\nEurope,Spain,Office
Supplies,Offline,C,1/12/2017,141812741,1/24/2017,4396,651.21,524.96,2862719.
16,2307724.16,554995.00\r\nMiddle East and North
Africa,Iran,Meat,Online,C,7/3/2016,212874114,8/17/2016,3036,421.89,364.69,12
80858.04,1107198.84,173659.20\r\nCentral America and the
Caribbean,Guatemala,Office
Supplies,Offline,C,3/27/2010,320368897,4/2/2010,3131,651.21,524.96,2038938.5
1,1643649.76,395288.75\r\nAustralia and Oceania,East
Timor,Beverages,Online,C,6/5/2015,179970920,6/25/2015,6249,47.45,31.79,29651
5.05,198655.71,97859.34\r\nMiddle East and North
Africa,Bahrain,Household,Online,H,6/27/2012,927666509,7/17/2012,5990,668.27,
502.54,4002937.30,3010214.60,992722.70\r\nSub-Saharan Africa,Ethiopia,Office
Supplies,Online,L,12/19/2016,169754493,1/20/2017,2982,651.21,524.96,1941908.
22,1565430.72,376477.50\r\nAustralia and Oceania,Solomon Islands,Personal
Care,Offline,M,3/9/2015,532846200,4/20/2015,9886,81.73,56.67,807982.78,56023
9.62,247743.16\r\nCentral America and the
Caribbean,Belize,Snacks,Online,C,6/18/2013,213865458,7/13/2013,6397,152.58,9
7.44,976054.26,623323.68,352730.58\r\nAsia,Sri Lanka,Office
Supplies,Online,C,8/12/2011,630048596,9/3/2011,4236,651.21,524.96,2758525.56
,2223730.56,534795.00\r\nCentral America and the Caribbean,Costa
Rica,Clothes,Offline,H,4/17/2014,568944442,4/24/2014,2158,109.28,35.84,23582
6.24,77342.72,158483.52\r\nSub-Saharan Africa,Nigeria,Baby
Food,Online,L,2/3/2012,238414323,2/27/2012,951,255.28,159.42,242771.28,15160
8.42,91162.86\r\nMiddle East and North Africa,Iran,Office
Supplies,Online,L,8/1/2015,816632068,9/19/2015,8431,651.21,524.96,5490351.51
,4425937.76,1064413.75\r\nSub-Saharan Africa,Djibouti,Baby
Food,Online,C,9/11/2013,402084004,10/5/2013,4447,255.28,159.42,1135230.16,70
8940.74,426289.42\r\nAsia,South
Korea,Snacks,Online,L,5/8/2015,763568961,6/7/2015,5879,152.58,97.44,897017.8
2,572849.76,324168.06\r\nCentral America and the
Caribbean,Dominica,Snacks,Offline,L,5/22/2015,590198266,6/1/2015,1637,152.58
,97.44,249773.46,159509.28,90264.18\r\nAsia,Vietnam,Snacks,Online,L,7/3/2013
,441395747,8/19/2013,7665,152.58,97.44,1169525.70,746877.60,422648.10\r\nEur
ope,Norway,Personal
Care,Offline,H,6/16/2010,496897733,7/21/2010,1936,81.73,56.67,158229.28,1097
13.12,48516.16\r\nCentral America and the
Caribbean,Haiti,Beverages,Online,C,11/8/2011,106753051,11/14/2011,9455,47.45
,31.79,448639.75,300574.45,148065.30\r\nCentral America and the

```



```

Caribbean, Jamaica, Fruits, Offline, L, 10/17/2016, 941323029, 10/27/2016, 7258, 9.33
, 6.92, 67717.14, 50225.36, 17491.78\r\nSub-Saharan
Africa, Sudan, Vegetables, Offline, M, 2/2/2014, 241281497, 3/3/2014, 9412, 154.06, 90
.93, 1450012.72, 855833.16, 594179.56\r\nSub-Saharan
Africa, Angola, Meat, Offline, L, 4/30/2016, 267614781, 5/12/2016, 2016, 421.89, 364.6
9, 850530.24, 735215.04, 115315.20\r\nCentral America and the
Caribbean, Panama, Cosmetics, Offline, M, 8/31/2010, 651621711, 10/16/2010, 8200, 437
.20, 263.33, 3585040.00, 2159306.00, 1425734.00\r\nEurope, Greece, Personal
Care, Online, M, 8/8/2015, 644913613, 9/7/2015, 3124, 81.73, 56.67, 255324.52, 177037.
08, 78287.44\r\nSub-Saharan
Africa, Madagascar, Cereal, Offline, H, 8/18/2016, 469414317, 8/19/2016, 8983, 205.70
, 117.11, 1847803.10, 1051999.13, 795803.97\r\nSub-Saharan Africa, Guinea-
Bissau, Clothes, Online, L, 6/18/2015, 867360150, 7/1/2015, 9998, 109.28, 35.84, 10925
81.44, 358328.32, 734253.12\r\nNorth
America, Greenland, Clothes, Online, L, 1/28/2011, 851299941, 2/1/2011, 7425, 109.28,
35.84, 811404.00, 266112.00, 545292.00\r\nMiddle East and North
Africa, Libya, Beverages, Offline, H, 2/4/2011, 854095017, 3/4/2011, 4550, 47.45, 31.7
9, 215897.50, 144644.50, 71253.00\r\nEurope, Belarus, Vegetables, Offline, C, 11/3/2
012, 478919208, 11/27/2012, 1691, 154.06, 90.93, 260515.46, 153762.63, 106752.83\r\n
Middle East and North
Africa, Lebanon, Clothes, Offline, H, 9/18/2014, 749258840, 11/5/2014, 1196, 109.28, 3
5.84, 130698.88, 42864.64, 87834.24\r\nSub-Saharan Africa, Djibouti, Baby
Food, Offline, H, 5/17/2012, 958912742, 6/28/2012, 2444, 255.28, 159.42, 623904.32, 38
9622.48, 234281.84\r\nCentral America and the Caribbean, Barbados, Personal
Care, Online, C, 2/20/2010, 921992242, 3/4/2010, 6848, 81.73, 56.67, 559687.04, 388076
.16, 171610.88\r\nSub-Saharan Africa, Guinea-
Bissau, Vegetables, Online, L, 2/28/2017, 522921168, 3/2/2017, 2849, 154.06, 90.93, 43
8916.94, 259059.57, 179857.37\r\nEurope, Finland, Personal
Care, Offline, C, 5/17/2013, 166435849, 6/7/2013, 921, 81.73, 56.67, 75273.33, 52193.0
7, 23080.26\r\nCentral America and the Caribbean, Haiti, Baby
Food, Offline, L, 11/9/2013, 327585113, 11/23/2013, 8569, 255.28, 159.42, 2187494.32,
1366069.98, 821424.34\r\nSub-Saharan
Africa, Niger, Clothes, Offline, M, 1/10/2012, 201730287, 2/19/2012, 5330, 109.28, 35.
84, 582462.40, 191027.20, 391435.20\r\nCentral America and the
Caribbean, Trinidad and
Tobago, Fruits, Offline, L, 9/24/2013, 854545199, 11/9/2013, 7769, 9.33, 6.92, 72484.7
7, 53761.48, 18723.29\r\nCentral America and the Caribbean, Grenada, Personal
Care, Online, M, 9/26/2010, 272016179, 11/8/2010, 4487, 81.73, 56.67, 366722.51, 25427
8.29, 112444.22\r\nCentral America and the Caribbean, Dominican
Republic, Cereal, Offline, H, 10/21/2014, 110442054, 11/20/2014, 1113, 205.70, 117.11
, 228944.10, 130343.43, 98600.67\r\nEurope, Monaco, Household, Offline, C, 1/17/2016
, 746434152, 2/5/2016, 5308, 668.27, 502.54, 3547177.16, 2667482.32, 879694.84\r\nEu
rope, Estonia, Vegetables, Online, L, 12/27/2016, 826916301, 1/7/2017, 1764, 154.06, 9
0.93, 271761.84, 160400.52, 111361.32\r\nEurope, Italy, Beverages, Offline, H, 8/30/
2013, 419124829, 9/19/2013, 7206, 47.45, 31.79, 341924.70, 229078.74, 112845.96\r\nA
sia, Malaysia, Office
Supplies, Online, C, 7/7/2014, 560608565, 8/24/2014, 5387, 651.21, 524.96, 3508068.27
, 2827959.52, 680108.75\r\nSub-Saharan Africa, Ghana, Office
Supplies, Offline, L, 11/29/2013, 109228837, 12/7/2013, 2095, 651.21, 524.96, 1364284
.95, 1099791.20, 264493.75\r\nMiddle East and North
Africa, Pakistan, Clothes, Offline, C, 1/29/2011, 693159472, 2/5/2011, 146, 109.28, 35
.84, 15954.88, 5232.64, 10722.24\r\nAsia, Sri
Lanka, Snacks, Offline, L, 11/13/2013, 860886800, 11/23/2013, 4390, 152.58, 97.44, 669
826.20, 427761.60, 242064.60\r\nEurope, Romania, Fruits, Offline, C, 4/9/2012, 13120
9647, 5/3/2012, 6705, 9.33, 6.92, 62557.65, 46398.60, 16159.05\r\nMiddle East and
North Africa, Qatar, Office
Supplies, Online, H, 6/28/2012, 343239343, 7/13/2012, 1004, 651.21, 524.96, 653814.84

```

```
,527059.84,126755.00\r\nSub-Saharan Africa,Cote
d'Ivoire,Clothes,Online,H,6/8/2010,706399714,7/19/2010,8228,109.28,35.84,899
155.84,294891.52,604264.32\r\nMiddle East and North Africa,Egypt,Office
Supplies,Online,M,8/23/2010,950427091,9/14/2010,1352,651.21,524.96,880435.92
,709745.92,170690.00\r\nMiddle East and North
Africa,Iran,Snacks,Offline,H,12/11/2014,875370299,12/28/2014,379,152.58,97.4
4,57827.82,36929.76,20898.06\r\nMiddle East and North
Africa,Somalia,Clothes,Online,C,1/27/2011,801590669,3/15/2011,7347,109.28,35
.84,802880.16,263316.48,539563.68\r\nMiddle East and North
Africa,Syria,Personal
Care,Offline,C,5/7/2014,219762027,5/28/2014,1322,81.73,56.67,108047.06,74917
.74,33129.32\r\nAustralia and Oceania,Solomon
Islands,Cereal,Offline,C,11/8/2010,940870702,11/21/2010,3404,205.70,117.11,7
00202.80,398642.44,301560.36\r\nCentral America and the
Caribbean,Guatemala,Fruits,Online,M,3/30/2014,346215522,5/4/2014,1721,9.33,6
.92,16056.93,11909.32,4147.61\r\nMiddle East and North
Africa,Kuwait,Clothes,Offline,C,7/9/2016,837407815,7/20/2016,6436,109.28,35.
84,703326.08,230666.24,472659.84\r\nMiddle East and North
Africa,Jordan,Meat,Online,L,7/15/2014,386371409,7/19/2014,4741,421.89,364.69
,2000180.49,1728995.29,271185.20\r\nAustralia and Oceania,Marshall
Islands,Beverages,Offline,H,10/14/2013,185342633,11/24/2013,5859,47.45,31.79
,278009.55,186257.61,91751.94\r\nMiddle East and North
Africa,Egypt,Snacks,Offline,C,1/13/2017,596870315,2/18/2017,6045,152.58,97.4
4,922346.10,589024.80,333321.30\r\nEurope,Switzerland,Meat,Online,C,12/22/20
12,703815782,1/7/2013,3585,421.89,364.69,1512475.65,1307413.65,205062.00\r\n
Australia and Oceania,Samoa ,Personal
Care,Online,C,4/16/2010,559352862,6/4/2010,3797,81.73,56.67,310328.81,215175
.99,95152.82\r\nEurope,Portugal,Cosmetics,Offline,C,2/27/2011,736967885,3/12
/2011,4029,437.20,263.33,1761478.80,1060956.57,700522.23\r\nEurope,Albania,C
lothes,Offline,C,1/14/2015,980459678,2/9/2015,8661,109.28,35.84,946474.08,31
0410.24,636063.84\r\nCentral America and the
Caribbean,Dominica,Vegetables,Offline,C,11/16/2014,653939568,12/6/2014,4105,
154.06,90.93,632416.30,373267.65,259148.65\r\nAustralia and
Oceania,Tuvalu,Cosmetics,Online,H,1/17/2016,991831386,1/29/2016,3803,437.20,
263.33,1662671.60,1001443.99,661227.61\r\nAustralia and Oceania,Marshall
Islands,Cereal,Online,L,5/29/2017,148871457,7/6/2017,3227,205.70,117.11,6637
93.90,377913.97,285879.93\r\nEurope,Bulgaria,Fruits,Online,H,2/10/2015,85010
8611,2/25/2015,4884,9.33,6.92,45567.72,33797.28,11770.44\r\nSub-Saharan
Africa,Niger,Office
Supplies,Offline,M,12/8/2013,940904176,1/7/2014,3309,651.21,524.96,2154853.8
9,1737092.64,417761.25\r\nCentral America and the Caribbean,Saint Vincent
and the Grenadines,Office
Supplies,Offline,H,12/10/2015,136931979,12/13/2015,70,651.21,524.96,45584.70
,36747.20,8837.50\r\nSub-Saharan
Africa,Malawi,Beverages,Offline,L,9/4/2016,474178349,9/26/2016,8766,47.45,31
.79,415946.70,278671.14,137275.56\r\nSub-Saharan Africa,Cape Verde,Personal
Care,Offline,L,7/19/2016,458942115,7/29/2016,25,81.73,56.67,2043.25,1416.75,
626.50\r\nCentral America and the Caribbean,Saint Vincent and the
Grenadines,Beverages,Offline,C,1/9/2017,917834603,1/13/2017,6510,47.45,31.79
,308899.50,206952.90,101946.60\r\nEurope,Greece,Personal
Care,Offline,H,11/5/2016,947779643,12/5/2016,7913,81.73,56.67,646729.49,4484
29.71,198299.78\r\nEurope,Monaco,Clothes,Online,L,10/31/2015,166013562,11/26
/2015,5957,109.28,35.84,650980.96,213498.88,437482.08\r\nSub-Saharan
Africa,Nigeria,Beverages,Online,L,1/25/2011,960085189,2/13/2011,9397,47.45,3
1.79,445887.65,298730.63,147157.02\r\nEurope,Norway,Cosmetics,Offline,C,11/8
/2015,837855851,11/8/2015,9020,437.20,263.33,3943544.00,2375236.60,1568307.4
0\r\nNorth
```

```

America,Greenland,Snacks,Offline,L,6/24/2010,977499377,8/12/2010,2643,152.58
,97.44,403268.94,257533.92,145735.02\r\nMiddle East and North Africa,Tunisia
,Beverages,Online,L,2/27/2014,377502095,3/3/2014,114,47.45,31.79,5409.30,362
4.06,1785.24\r\nAsia,Uzbekistan,Meat,Offline,H,2/5/2013,806662833,3/27/2013,
8313,421.89,364.69,3507171.57,3031667.97,475503.60\r\nCentral America and
the Caribbean,Saint Kitts and Nevis
,Vegetables,Online,H,4/15/2017,954092919,5/11/2017,6152,154.06,90.93,947777.
12,559401.36,388375.76\r\nCentral America and the
Caribbean,Belize,Meat,Online,L,4/9/2011,479216182,4/26/2011,9572,421.89,364.
69,4038331.08,3490812.68,547518.40\r\nSub-Saharan Africa,Angola,Personal
Care,Online,H,12/13/2010,461768949,12/30/2010,6548,81.73,56.67,535168.04,371
075.16,164092.88\r\nAsia,Bhutan,Meat,Online,C,5/13/2014,251800048,5/22/2014,
2085,421.89,364.69,879640.65,760378.65,119262.00\r\nCentral America and the
Caribbean,Honduras,Personal
Care,Offline,L,1/16/2013,619670808,2/25/2013,3217,81.73,56.67,262925.41,1823
07.39,80618.02\r\nSub-Saharan Africa,South
Sudan,Household,Offline,H,12/9/2010,606055057,1/23/2011,4014,668.27,502.54,2
682435.78,2017195.56,665240.22\r\nAsia,Kyrgyzstan,Baby
Food,Online,L,1/24/2013,671939122,2/14/2013,573,255.28,159.42,146275.44,9134
7.66,54927.78\r\nSub-Saharan Africa,Sao Tome and
Principe,Cosmetics,Offline,M,2/25/2014,448621833,3/3/2014,6025,437.20,263.33
,2634130.00,1586563.25,1047566.75\r\nSub-Saharan
Africa,Madagascar,Fruits,Online,M,7/23/2017,987714517,9/11/2017,5530,9.33,6.
92,51594.90,38267.60,13327.30\r\nSub-Saharan
Africa,Senegal,Household,Online,H,5/7/2016,711141002,6/14/2016,1280,668.27,5
02.54,855385.60,643251.20,212134.40\r\nSub-Saharan Africa,Sierra
Leone,Office
Supplies,Online,C,3/24/2012,361137616,4/18/2012,7501,651.21,524.96,4884726.2
1,3937724.96,947001.25\r\nAsia,Malaysia,Household,Offline,H,10/3/2011,750253
188,11/21/2011,5446,668.27,502.54,3639398.42,2736832.84,902565.58\r\nCentral
America and the Caribbean,Cuba,Office
Supplies,Online,C,12/22/2014,511349046,1/2/2015,8401,651.21,524.96,5470815.2
1,4410188.96,1060626.25\r\nSub-Saharan Africa,Zimbabwe,Personal
Care,Offline,L,8/23/2011,147599017,8/28/2011,6684,81.73,56.67,546283.32,3787
82.28,167501.04\r\nEurope,Serbia,Beverages,Online,H,4/18/2015,682489430,5/23
/2015,2644,47.45,31.79,125457.80,84052.76,41405.04\r\nAsia,Maldives,Vegetabl
es,Offline,M,1/11/2014,509819114,2/23/2014,5660,154.06,90.93,871979.60,51466
3.80,357315.80\r\nEurope,Ireland,Office
Supplies,Offline,M,2/27/2012,343699395,4/2/2012,7144,651.21,524.96,4652244.2
4,3750314.24,901930.00\r\nEurope,Romania,Clothes,Offline,L,4/4/2011,96855410
3,4/8/2011,5537,109.28,35.84,605083.36,198446.08,406637.28\r\nEurope,Croatia
,Beverages,Online,M,12/26/2013,989119565,1/6/2014,1315,47.45,31.79,62396.75,
41803.85,20592.90\r\nEurope,Albania,Vegetables,Offline,M,9/12/2012,880444610
,10/31/2012,1980,154.06,90.93,305038.80,180041.40,124997.40\r\nCentral
America and the Caribbean,Dominican
Republic,Vegetables,Offline,L,12/19/2015,737890565,1/15/2016,7071,154.06,90.
93,1089358.26,642966.03,446392.23\r\nSub-Saharan
Africa,Zimbabwe,Vegetables,Online,M,6/26/2014,727131259,8/9/2014,3153,154.06
,90.93,485751.18,286702.29,199048.89\r\nSub-Saharan Africa,Ghana,Office
Supplies,Offline,L,3/2/2015,634153020,3/11/2015,8826,651.21,524.96,5747579.4
6,4633296.96,1114282.50\r\nAsia,Laos,Beverages,Online,H,8/17/2014,315254676,
9/8/2014,9719,47.45,31.79,461166.55,308967.01,152199.54\r\nCentral America
and the Caribbean,Panama,Personal
Care,Offline,M,7/19/2010,147047555,9/3/2010,3494,81.73,56.67,285564.62,19800
4.98,87559.64\r\nSub-Saharan
Africa,Angola,Household,Online,L,1/14/2013,576455485,2/13/2013,4843,668.27,5
02.54,3236431.61,2433801.22,802630.39\r\nMiddle East and North

```

```

Africa,Syria,Snacks,Offline,L,7/19/2010,770714795,8/26/2010,490,152.58,97.44
,74764.20,47745.60,27018.60\r\nSub-Saharan Africa,Sierra
Leone,Cosmetics,Offline,H,7/11/2011,867374312,8/3/2011,4189,437.20,263.33,18
31430.80,1103089.37,728341.43\r\nSub-Saharan
Africa,Uganda,Fruits,Online,L,5/4/2010,624295365,6/23/2010,1727,9.33,6.92,16
112.91,11950.84,4162.07\r\nAsia,Taiwan,Clothes,Offline,M,10/13/2013,76965178
2,11/3/2013,5921,109.28,35.84,647046.88,212208.64,434838.24\r\nMiddle East
and North
Africa,Azerbaijan,Vegetables,Online,C,7/27/2014,751929891,8/1/2014,1619,154.
06,90.93,249423.14,147215.67,102207.47\r\nAsia,Maldives,Office
Supplies,Online,H,4/4/2010,989928519,4/11/2010,702,651.21,524.96,457149.42,3
68521.92,88627.50\r\nSub-Saharan
Africa,Mauritania,Meat,Offline,H,8/17/2014,622758996,10/1/2014,7081,421.89,3
64.69,2987403.09,2582369.89,405033.20\r\nSub-Saharan Africa,Burundi,Baby
Food,Online,H,1/4/2016,659845149,1/29/2016,1698,255.28,159.42,433465.44,2706
95.16,162770.28\r\nSub-Saharan Africa,Zambia,Baby
Food,Offline,H,5/17/2011,830923306,6/5/2011,7526,255.28,159.42,1921237.28,11
99794.92,721442.36\r\nAsia,Singapore,Beverages,Offline,L,2/12/2010,936042296
,3/17/2010,4571,47.45,31.79,216893.95,145312.09,71581.86\r\nSub-Saharan
Africa,Ghana,Household,Online,L,2/17/2015,395563447,3/30/2015,4869,668.27,50
2.54,3253806.63,2446867.26,806939.37\r\nSub-Saharan
Africa,Guinea,Meat,Offline,M,6/7/2011,500160586,6/7/2011,7487,421.89,364.69,
3158690.43,2730434.03,428256.40\r\nSub-Saharan
Africa,Zambia,Fruits,Online,L,5/26/2015,360820043,7/2/2015,3524,9.33,6.92,32
878.92,24386.08,8492.84\r\nEurope,Georgia,Snacks,Online,C,5/24/2010,95884064
4,6/2/2010,1109,152.58,97.44,169211.22,108060.96,61150.26\r\nMiddle East and
North Africa,Bahrain,Baby
Food,Online,M,2/21/2011,195833718,4/7/2011,404,255.28,159.42,103133.12,64405
.68,38727.44\r\nSub-Saharan Africa,Lesotho,Personal
Care,Offline,M,6/28/2014,543723094,7/2/2014,8601,81.73,56.67,702959.73,48741
8.67,215541.06\r\nCentral America and the
Caribbean,Barbados,Cosmetics,Offline,H,10/15/2010,494745099,10/30/2010,4924,
437.20,263.33,2152772.80,1296636.92,856135.88\r\nMiddle East and North
Africa,Saudi
Arabia,Vegetables,Online,H,5/18/2010,411448562,6/30/2010,5628,154.06,90.93,8
67049.68,511754.04,355295.64\r\nEurope,Macedonia,Personal
Care,Offline,H,2/2/2017,276694810,2/16/2017,8998,81.73,56.67,735406.54,50991
6.66,225489.88\r\nAsia,Turkmenistan,Office
Supplies,Offline,M,11/24/2016,143657672,1/8/2017,352,651.21,524.96,229225.92
,184785.92,44440.00\r\nEurope,Albania,Baby
Food,Online,L,12/23/2011,585823476,1/6/2012,7040,255.28,159.42,1797171.20,11
22316.80,674854.40\r\nMiddle East and North
Africa,Afghanistan,Clothes,Online,C,12/5/2016,446991050,1/16/2017,3440,109.2
8,35.84,375923.20,123289.60,252633.60\r\nAustralia and
Oceania,Kiribati,Clothes,Offline,M,2/16/2017,891271722,2/22/2017,5963,109.28
,35.84,651636.64,213713.92,437922.72\r\nMiddle East and North
Africa,Morocco,Cosmetics,Online,C,5/18/2010,453089320,6/16/2010,8053,437.20,
263.33,3520771.60,2120596.49,1400175.11\r\nEurope,Norway,Baby
Food,Offline,L,9/7/2010,887180173,10/18/2010,5183,255.28,159.42,1323116.24,8
26273.86,496842.38\r\nEurope,Sweden,Cosmetics,Online,L,3/12/2014,418593108,3
/25/2014,9858,437.20,263.33,4309917.60,2595907.14,1714010.46\r\nAsia,Tajikis
tan,Personal
Care,Online,M,7/16/2010,492689454,8/16/2010,6613,81.73,56.67,540480.49,37475
8.71,165721.78\r\nEurope,Netherlands,Cosmetics,Online,M,12/5/2016,825143039,
12/20/2016,7017,437.20,263.33,3067832.40,1847786.61,1220045.79\r\nEurope,Spa
in,Vegetables,Online,L,10/8/2013,751940190,10/10/2013,4667,154.06,90.93,7189
98.02,424370.31,294627.71\r\nSub-Saharan Africa,Chad,Baby

```

```

Food,Offline,H,8/9/2016,579379737,8/26/2016,194,255.28,159.42,49524.32,30927
.48,18596.84\r\nEurope,Ireland,Meat,Online,C,10/6/2011,234073007,11/20/2011,
6259,421.89,364.69,2640609.51,2282594.71,358014.80\r\nMiddle East and North
Africa,Pakistan,Meat,Online,C,2/17/2013,612943828,3/1/2013,2554,421.89,364.6
9,1077507.06,931418.26,146088.80\r\nSub-Saharan
Africa,Mozambique,Fruits,Online,C,12/14/2012,433228528,12/21/2012,804,9.33,6
.92,7501.32,5563.68,1937.64\r\nMiddle East and North
Africa,Bahrain,Fruits,Online,L,11/20/2015,282475936,11/28/2015,9762,9.33,6.9
2,91079.46,67553.04,23526.42\r\nAsia,Tajikistan,Meat,Online,M,7/7/2012,36854
7379,7/12/2012,214,421.89,364.69,90284.46,78043.66,12240.80\r\nAustralia and
Oceania,New
Zealand,Meat,Offline,H,6/2/2013,969616687,6/28/2013,9980,421.89,364.69,42104
62.20,3639606.20,570856.00\r\nSub-Saharan Africa,Niger,Baby
Food,Online,L,2/4/2015,184170186,2/17/2015,8906,255.28,159.42,2273523.68,141
9794.52,853729.16\r\nEurope,Armenia,Fruits,Online,C,4/28/2011,681006705,6/14
/2011,3872,9.33,6.92,36125.76,26794.24,9331.52\r\nSub-Saharan
Africa,Gabon,Baby
Food,Online,L,2/1/2012,249237573,2/21/2012,3791,255.28,159.42,967766.48,6043
61.22,363405.26\r\nAsia,Kyrgyzstan,Vegetables,Offline,H,1/29/2017,348286616,
2/13/2017,4604,154.06,90.93,709292.24,418641.72,290650.52\r\nAustralia and
Oceania,Fiji,Clothes,Offline,L,12/1/2011,257890164,12/29/2011,4285,109.28,35
.84,468264.80,153574.40,314690.40\r\nEurope,Romania,Cosmetics,Online,H,5/8/2
017,228097045,6/17/2017,7839,437.20,263.33,3427210.80,2064243.87,1362966.93\r
\nSub-Saharan
Africa,Botswana,Cereal,Online,C,5/3/2017,129268586,6/21/2017,2302,205.70,117
.11,473521.40,269587.22,203934.18\r\nAustralia and
Oceania,Fiji,Cosmetics,Online,M,10/31/2015,802078616,12/9/2015,1741,437.20,2
63.33,761165.20,458457.53,302707.67\r\nEurope,Vatican
City,Clothes,Offline,C,4/11/2010,907513463,4/19/2010,2256,109.28,35.84,24653
5.68,80855.04,165680.64\r\nAsia,Thailand,Vegetables,Online,L,7/17/2010,97687
1955,8/31/2010,6975,154.06,90.93,1074568.50,634236.75,440331.75\r\nEurope,Bel
arus,Cereal,Offline,C,5/31/2012,261765420,7/17/2012,1060,205.70,117.11,2180
42.00,124136.60,93905.40\r\nAustralia and Oceania,Solomon
Islands,Vegetables,Offline,L,7/15/2016,784117686,7/17/2016,6703,154.06,90.93
,1032664.18,609503.79,423160.39\r\nAsia,China,Cosmetics,Offline,M,1/14/2017,
586165082,1/27/2017,8128,437.20,263.33,3553561.60,2140346.24,1413215.36\r\nS
ub-Saharan
Africa,Angola,Snacks,Online,C,11/26/2012,480456435,12/16/2012,6591,152.58,97
.44,1005654.78,642227.04,363427.74\r\nAsia,Cambodia,Cereal,Online,L,10/3/201
1,899853074,10/26/2011,5376,205.70,117.11,1105843.20,629583.36,476259.84\r\n
Central America and the Caribbean,Guatemala,Baby
Food,Offline,L,11/30/2010,547528827,12/15/2010,4802,255.28,159.42,1225854.56
,765534.84,460319.72\r\nSub-Saharan
Africa,Namibia,Vegetables,Online,L,5/8/2016,446970021,5/9/2016,7217,154.06,9
0.93,1111851.02,656241.81,455609.21\r\nEurope,Serbia,Beverages,Offline,H,2/2
7/2016,791975486,3/20/2016,2001,47.45,31.79,94947.45,63611.79,31335.66\r\nMi
ddle East and North
Africa,Turkey,Vegetables,Offline,M,8/24/2014,496656548,9/29/2014,564,154.06,
90.93,86889.84,51284.52,35605.32\r\nMiddle East and North
Africa,Pakistan,Vegetables,Offline,C,5/26/2013,345437037,6/30/2013,1351,154.
06,90.93,208135.06,122846.43,85288.63\r\nEurope,Georgia,Vegetables,Offline,H
,1/26/2015,743053281,2/23/2015,4833,154.06,90.93,744571.98,439464.69,305107.
29\r\nAustralia and
Oceania,Vanuatu,Snacks,Offline,L,1/7/2012,364554107,1/18/2012,8516,152.58,97
.44,1299371.28,829799.04,469572.24\r\nEurope,Luxembourg,Cosmetics,Online,H,1
0/21/2012,205300843,12/3/2012,1937,437.20,263.33,846856.40,510070.21,336786.
19\r\nMiddle East and North Africa,Saudi

```

```

Arabia,Household,Online,M,9/16/2015,430967319,10/5/2015,1661,668.27,502.54,1
109996.47,834718.94,275277.53\r\nAustralia and
Oceania,Vanuatu,Meat,Offline,H,6/28/2012,827539861,7/1/2012,6289,421.89,364.
69,2653266.21,2293535.41,359730.80\r\nCentral America and the
Caribbean,Haiti,Snacks,Offline,H,12/2/2014,351317298,1/14/2015,1450,152.58,9
7.44,221241.00,141288.00,79953.00\r\nSub-Saharan
Africa,Tanzania,Household,Offline,C,2/4/2013,278910958,2/4/2013,4805,668.27,
502.54,3211037.35,2414704.70,796332.65\r\nAsia,North
Korea,Cosmetics,Offline,C,2/17/2017,157244670,3/15/2017,1047,437.20,263.33,4
57748.40,275706.51,182041.89\r\nMiddle East and North
Africa,Bahrain,Beverages,Online,L,6/24/2015,953554761,7/28/2015,6899,47.45,3
1.79,327357.55,219319.21,108038.34\r\nSub-Saharan Africa,Cote
d'Ivoire,Cereal,Online,L,7/30/2016,105390059,8/25/2016,6115,205.70,117.11,12
57855.50,716127.65,541727.85\r\nAsia,Singapore,Cosmetics,Offline,L,10/23/201
4,970611894,11/10/2014,4483,437.20,263.33,1959967.60,1180508.39,779459.21\r\
nAsia,Malaysia,Baby
Food,Offline,M,12/16/2016,677394092,12/29/2016,4820,255.28,159.42,1230449.60
,768404.40,462045.20\r\nEurope,Albania,Baby
Food,Online,L,9/28/2015,474222981,10/26/2015,1973,255.28,159.42,503667.44,31
4535.66,189131.78\r\nSub-Saharan
Africa,Gabon,Snacks,Online,L,4/16/2016,779897391,5/5/2016,7824,152.58,97.44,
1193785.92,762370.56,431415.36\r\nEurope,Poland,Snacks,Offline,L,3/27/2017,7
33528649,3/30/2017,6283,152.58,97.44,958660.14,612215.52,346444.62\r\nSub-
Saharan Africa,Chad,Office
Supplies,Online,H,6/25/2014,444540584,8/2/2014,8292,651.21,524.96,5399833.32
,4352968.32,1046865.00\r\nSub-Saharan Africa,Republic of the
Congo,Clothes,Online,H,7/29/2012,542669522,9/11/2012,6826,109.28,35.84,74594
5.28,244643.84,501301.44\r\nAsia,Philippines,Fruits,Offline,L,5/4/2013,82796
4293,6/1/2013,1888,9.33,6.92,17615.04,13064.96,4550.08\r\nEurope,France,Snac
ks,Offline,M,2/2/2012,720786225,2/15/2012,5516,152.58,97.44,841631.28,537479
.04,304152.24\r\nEurope,Germany,Household,Offline,H,7/24/2016,611809146,8/1/
2016,6777,668.27,502.54,4528865.79,3405713.58,1123152.21\r\nSub-Saharan
Africa,Uganda,Personal
Care,Online,M,11/4/2012,512019383,12/12/2012,6769,81.73,56.67,553230.37,3835
99.23,169631.14\r\nMiddle East and North Africa,Bahrain,Personal
Care,Online,C,8/5/2014,502715766,8/17/2014,3621,81.73,56.67,295944.33,205202
.07,90742.26\r\nMiddle East and North Africa,Jordan,Office
Supplies,Offline,M,10/22/2013,285509622,10/28/2013,7497,651.21,524.96,488212
1.37,3935625.12,946496.25\r\nEurope,Montenegro,Beverages,Offline,L,12/1/2014
,532324779,1/3/2015,5586,47.45,31.79,265055.70,177578.94,87476.76\r\nMiddle
East and North Africa,Tunisia ,Personal
Care,Offline,H,11/2/2015,635397565,11/21/2015,7114,81.73,56.67,581427.22,403
150.38,178276.84\r\nEurope,Germany,Cereal,Offline,M,10/7/2011,957276809,11/4
/2011,8335,205.70,117.11,1714509.50,976111.85,738397.65\r\nEurope,Italy,Snac
ks,Online,L,2/9/2014,580823838,3/21/2014,7536,152.58,97.44,1149842.88,734307
.84,415535.04\r\nEurope,France,Office
Supplies,Offline,H,5/29/2016,459212481,6/16/2016,33,651.21,524.96,21489.93,1
7323.68,4166.25\r\nMiddle East and North Africa,Algeria,Baby
Food,Online,L,4/8/2013,265929067,5/23/2013,3175,255.28,159.42,810514.00,5061
58.50,304355.50\r\nAsia,Myanmar,Beverages,Online,H,9/20/2011,644772422,10/26
/2011,1343,47.45,31.79,63725.35,42693.97,21031.38\r\nEurope,France,Vegetable
s,Offline,C,4/12/2012,959853875,5/4/2012,947,154.06,90.93,145894.82,86110.71
,59784.11\r\nEurope,Spain,Vegetables,Online,C,10/2/2012,645597255,10/25/2012
,5429,154.06,90.93,836391.74,493658.97,342732.77\r\nAsia,Cambodia,Baby
Food,Offline,C,1/2/2012,556738889,1/25/2012,264,255.28,159.42,67393.92,42086
.88,25307.04\r\nSub-Saharan Africa,The
Gambia,Vegetables,Online,M,9/30/2013,718327605,11/10/2013,7956,154.06,90.93,

```

```

1225701.36,723439.08,502262.28\r\nEurope,Russia,Baby
Food,Offline,L,3/10/2015,775724732,3/20/2015,3041,255.28,159.42,776306.48,48
4796.22,291510.26\r\nEurope,Belarus,Baby
Food,Offline,H,10/26/2010,444604098,10/31/2010,7088,255.28,159.42,1809424.64
,1129968.96,679455.68\r\nMiddle East and North
Africa,Turkey,Beverages,Online,C,5/2/2011,860952031,5/13/2011,3693,47.45,31.
79,175232.85,117400.47,57832.38\r\nSub-Saharan
Africa,Kenya,Meat,Offline,C,2/19/2017,531067359,2/20/2017,3488,421.89,364.69
,1471552.32,1272038.72,199513.60\r\nMiddle East and North
Africa,Iran,Cosmetics,Offline,M,7/20/2014,281561410,8/2/2014,9133,437.20,263
.33,3992947.60,2404992.89,1587954.71\r\nAsia,Vietnam,Personal
Care,Offline,L,6/18/2017,109358012,7/10/2017,321,81.73,56.67,26235.33,18191.
07,8044.26\r\nEurope,Albania,Beverages,Online,L,7/19/2010,531693494,8/6/2010
,8775,47.45,31.79,416373.75,278957.25,137416.50\r\nCentral America and the
Caribbean,Antigua and Barbuda
,Clothes,Offline,C,8/9/2013,336116683,9/4/2013,3251,109.28,35.84,355269.28,1
16515.84,238753.44\r\nSub-Saharan
Africa,Senegal,Cereal,Offline,M,3/23/2017,630488908,4/30/2017,4534,205.70,11
7.11,932643.80,530976.74,401667.06\r\nEurope,Netherlands,Fruits,Online,L,10/
31/2016,792983996,12/2/2016,441,9.33,6.92,4114.53,3051.72,1062.81\r\nEurope,
Russia,Fruits,Online,C,7/3/2016,722088277,7/3/2016,822,9.33,6.92,7669.26,568
8.24,1981.02\r\nEurope,Slovakia,Cosmetics,Online,H,1/7/2010,386600577,1/29/2
010,2557,437.20,263.33,1117920.40,673334.81,444585.59\r\nAustralia and
Oceania,East
Timor,Beverages,Offline,L,12/6/2014,275632226,1/18/2015,4556,47.45,31.79,216
182.20,144835.24,71346.96\r\nCentral America and the
Caribbean,Haiti,Vegetables,Offline,H,10/16/2015,948607051,11/27/2015,2761,15
4.06,90.93,425359.66,251057.73,174301.93\r\nMiddle East and North
Africa,Yemen,Cereal,Online,H,5/10/2013,785261380,5/26/2013,5147,205.70,117.1
1,1058737.90,602765.17,455972.73\r\nAustralia and
Oceania,Tuvalu,Cereal,Online,C,3/27/2013,935644042,5/15/2013,6719,205.70,117
.11,1382098.30,786862.09,595236.21\r\nSub-Saharan
Africa,Liberia,Snacks,Offline,M,7/13/2014,370116364,8/17/2014,4512,152.58,97
.44,688440.96,439649.28,248791.68\r\nAsia,North
Korea,Beverages,Online,C,8/16/2013,829352176,8/26/2013,2594,47.45,31.79,1230
85.30,82463.26,40622.04\r\nAsia,North
Korea,Household,Online,M,7/14/2015,974337804,8/7/2015,7063,668.27,502.54,471
9991.01,3549440.02,1170550.99\r\nEurope,Romania,Household,Online,H,12/23/201
4,436372077,1/3/2015,1050,668.27,502.54,701683.50,527667.00,174016.50\r\nSub
-Saharan Africa,Sao Tome and
Principe,Cereal,Offline,M,4/27/2015,267066323,5/19/2015,9715,205.70,117.11,1
998375.50,1137723.65,860651.85\r\nMiddle East and North
Africa,Bahrain,Fruits,Online,M,3/14/2017,688344371,4/28/2017,5251,9.33,6.92,
48991.83,36336.92,12654.91\r\nMiddle East and North
Africa,Somalia,Cosmetics,Offline,M,10/2/2014,642442548,11/2/2014,1881,437.20
,263.33,822373.20,495323.73,327049.47\r\nEurope,Cyprus,Cereal,Online,C,7/13/
2014,941909682,8/1/2014,861,205.70,117.11,177107.70,100831.71,76275.99\r\nEu
rope,United
Kingdom,Beverages,Offline,C,7/31/2016,219607102,8/13/2016,5477,47.45,31.79,2
59883.65,174113.83,85769.82\r\nEurope,Germany,Personal
Care,Offline,H,8/30/2010,778708636,9/2/2010,6045,81.73,56.67,494057.85,34257
0.15,151487.70\r\nMiddle East and North
Africa,Somalia,Beverages,Online,M,4/22/2010,942700612,6/6/2010,4915,47.45,31
.79,233216.75,156247.85,76968.90\r\nAustralia and Oceania,New
Zealand,Household,Offline,H,3/9/2011,905381858,4/8/2011,1466,668.27,502.54,9
79683.82,736723.64,242960.18\r\nMiddle East and North
Africa,Kuwait,Meat,Online,C,1/13/2010,480863702,1/28/2010,7110,421.89,364.69

```

```
,2999637.90,2592945.90,406692.00\r\nAsia,Japan,Cosmetics,Online,L,1/6/2016,4
53569972,2/19/2016,289,437.20,263.33,126350.80,76102.37,50248.43\r\nEurope,N
orway,Baby
Food,Offline,H,8/17/2016,328236997,9/10/2016,1476,255.28,159.42,376793.28,23
5303.92,141489.36\r\nSub-Saharan Africa,Lesotho,Personal
Care,Offline,C,10/22/2011,579913604,10/23/2011,8177,81.73,56.67,668306.21,46
3390.59,204915.62\r\nEurope,Belgium,Cosmetics,Online,H,2/12/2010,403961122,3
/20/2010,9928,437.20,263.33,4340521.60,2614340.24,1726181.36\r\nCentral
America and the
Caribbean,Honduras,Cosmetics,Offline,L,5/29/2017,866053378,6/22/2017,3295,43
7.20,263.33,1440574.00,867672.35,572901.65\r\nEurope,Austria,Household,Offli
ne,C,4/17/2010,852176702,5/13/2010,6878,668.27,502.54,4596361.06,3456470.12,
1139890.94\r\nMiddle East and North Africa,Oman,Baby
Food,Online,M,9/8/2015,218629920,10/20/2015,6307,255.28,159.42,1610050.96,10
05461.94,604589.02\r\nMiddle East and North Africa,Oman,Baby
Food,Offline,C,2/13/2016,242024362,3/17/2016,9242,255.28,159.42,2359297.76,1
473359.64,885938.12\r\nEurope,Spain,Snacks,Offline,M,2/16/2014,469283854,2/1
6/2014,376,152.58,97.44,57370.08,36637.44,20732.64\r\nMiddle East and North
Africa,Afghanistan,Fruits,Online,C,4/22/2013,967644727,4/30/2013,6433,9.33,6
.92,60019.89,44516.36,15503.53\r\nCentral America and the Caribbean,Saint
Vincent and the
Grenadines,Cosmetics,Offline,L,7/15/2014,974655807,7/23/2014,1167,437.20,263
.33,510212.40,307306.11,202906.29\r\nEurope,Iceland,Vegetables,Online,M,3/7/
2012,248178422,3/22/2012,365,154.06,90.93,56231.90,33189.45,23042.45\r\nAsia
,Myanmar,Vegetables,Offline,L,1/14/2013,416386401,2/16/2013,6844,154.06,90.9
3,1054386.64,622324.92,432061.72\r\nEurope,Netherlands,Snacks,Offline,L,5/22
/2017,927766072,6/20/2017,5453,152.58,97.44,832018.74,531340.32,300678.42\r\
nEurope,Slovakia,Vegetables,Online,M,3/23/2015,401116263,3/31/2015,8071,154.
06,90.93,1243418.26,733896.03,509522.23\r\nMiddle East and North
Africa,Bahrain,Fruits,Offline,H,9/28/2012,675548303,11/6/2012,8610,9.33,6.92
,80331.30,59581.20,20750.10\r\nSub-Saharan Africa,Lesotho,Baby
Food,Online,L,6/6/2013,960486018,7/4/2013,8012,255.28,159.42,2045303.36,1277
273.04,768030.32\r\nCentral America and the
Caribbean,Cuba,Clothes,Offline,L,4/2/2014,985665738,5/19/2014,9250,109.28,35
.84,1010840.00,331520.00,679320.00\r\nMiddle East and North
Africa,Afghanistan,Clothes,Offline,C,1/13/2017,551136291,1/13/2017,2331,109.
28,35.84,254731.68,83543.04,171188.64\r\nAustralia and
Oceania,Vanuatu,Cereal,Online,L,2/3/2017,877259004,2/16/2017,9289,205.70,117
.11,1910747.30,1087834.79,822912.51\r\nAsia,Bhutan,Beverages,Offline,M,8/5/2
014,554707705,9/19/2014,9192,47.45,31.79,436160.40,292213.68,143946.72\r\nAu
stralia and
Oceania,Palau,Cereal,Offline,L,10/6/2010,494468724,10/23/2010,3139,205.70,11
7.11,645692.30,367608.29,278084.01\r\nAsia,Indonesia,Personal
Care,Online,H,9/10/2011,777840888,10/23/2011,9259,81.73,56.67,756738.07,5247
07.53,232030.54\r\nEurope,Andorra,Baby
Food,Offline,C,12/12/2016,206435525,1/27/2017,7714,255.28,159.42,1969229.92,
1229765.88,739464.04\r\nMiddle East and North Africa,Algeria,Office
Supplies,Online,C,10/26/2015,352176463,12/5/2015,5696,651.21,524.96,3709292.
16,2990172.16,719120.00\r\nAustralia and
Oceania,Vanuatu,Cosmetics,Offline,C,10/14/2013,607300031,10/14/2013,2429,437
.20,263.33,1061958.80,639628.57,422330.23\r\nNorth America,Mexico,Baby
Food,Online,H,8/17/2013,434355056,9/28/2013,4168,255.28,159.42,1064007.04,66
4462.56,399544.48\r\nEurope,Macedonia,Fruits,Offline,M,8/7/2011,716202867,9/
20/2011,9199,9.33,6.92,85826.67,63657.08,22169.59\r\nCentral America and the
Caribbean,Panama,Personal
Care,Online,H,5/22/2016,606017291,6/12/2016,2838,81.73,56.67,231949.74,16082
9.46,71120.28\r\nAsia,Nepal,Cereal,Offline,H,1/13/2015,677284657,1/15/2015,2
```



```

436,205.70,117.11,501085.20,285279.96,215805.24\r\nAsia,Nepal,Fruits,Online,
C,7/22/2014,673803794,7/29/2014,2371,9.33,6.92,22121.43,16407.32,5714.11\r\n
Sub-Saharan Africa,Mauritius
,Cereal,Offline,C,6/25/2015,859686028,7/10/2015,9055,205.70,117.11,1862613.5
0,1060431.05,802182.45\r\nSub-Saharan Africa,Sao Tome and
Principe,Clothes,Online,H,8/10/2015,669355189,9/26/2015,5930,109.28,35.84,64
8030.40,212531.20,435499.20\r\nCentral America and the Caribbean,Saint
Vincent and the
Grenadines,Fruits,Offline,L,10/19/2013,957547605,11/21/2013,8470,9.33,6.92,7
9025.10,58612.40,20412.70\r\nAsia,Maldives,Personal
Care,Online,M,10/9/2013,849312102,11/23/2013,9180,81.73,56.67,750281.40,5202
30.60,230050.80\r\nSub-Saharan Africa,Swaziland,Personal
Care,Offline,H,9/26/2012,890010011,10/14/2012,2595,81.73,56.67,212089.35,147
058.65,65030.70\r\nMiddle East and North
Africa,Morocco,Cereal,Online,H,10/2/2012,795315158,10/26/2012,284,205.70,117
.11,58418.80,33259.24,25159.56\r\nAsia,Maldives,Clothes,Offline,M,12/11/2016
,801213872,1/28/2017,5844,109.28,35.84,638632.32,209448.96,429183.36\r\nSub-
Saharan
Africa,Zimbabwe,Fruits,Offline,C,7/26/2010,314004981,8/9/2010,9907,9.33,6.92
,92432.31,68556.44,23875.87\r\nAsia,India,Cereal,Online,C,3/27/2010,16029981
3,4/6/2010,5132,205.70,117.11,1055652.40,601008.52,454643.88\r\nAsia,Tajikis
tan,Beverages,Offline,C,9/11/2014,337022197,9/22/2014,1212,47.45,31.79,57509
.40,38529.48,18979.92\r\nSub-Saharan
Africa,Lesotho,Snacks,Offline,M,3/9/2016,461408460,3/15/2016,9872,152.58,97.
44,1506269.76,961927.68,544342.08\r\nAsia,Bhutan,Office
Supplies,Offline,M,10/19/2011,221007430,11/10/2011,9865,651.21,524.96,642418
6.65,5178730.40,1245456.25\r\nCentral America and the Caribbean,Trinidad and
Tobago,Snacks,Offline,L,10/31/2014,723680436,12/20/2014,1978,152.58,97.44,30
1803.24,192736.32,109066.92\r\nAustralia and
Oceania,Tuvalu,Fruits,Offline,L,2/23/2013,447601306,3/11/2013,4028,9.33,6.92
,37581.24,27873.76,9707.48\r\nMiddle East and North
Africa,Iraq,Clothes,Offline,L,10/25/2010,191256368,11/9/2010,5864,109.28,35.
84,640817.92,210165.76,430652.16\r\nSub-Saharan Africa,The
Gambia,Vegetables,Offline,L,10/4/2016,823444449,10/30/2016,4366,154.06,90.93
,672625.96,397000.38,275625.58\r\nMiddle East and North
Africa,Bahrain,Beverages,Online,C,9/4/2010,133276879,10/17/2010,8445,47.45,3
1.79,400715.25,268466.55,132248.70\r\nMiddle East and North
Africa,Qatar,Meat,Online,H,12/22/2014,480177485,2/7/2015,4043,421.89,364.69,
1705701.27,1474441.67,231259.60\r\nSub-Saharan
Africa,Angola,Household,Offline,H,4/8/2015,243882596,5/11/2015,9135,668.27,5
02.54,6104646.45,4590702.90,1513943.55\r\nCentral America and the
Caribbean,Costa
Rica,Cosmetics,Online,L,3/2/2017,574441039,4/6/2017,8724,437.20,263.33,38141
32.80,2297290.92,1516841.88\r\nAustralia and Oceania,Papua New
Guinea,Household,Online,M,3/14/2012,442214143,5/3/2012,9847,668.27,502.54,65
80454.69,4948511.38,1631943.31\r\nMiddle East and North
Africa,Qatar,Clothes,Offline,C,11/22/2011,687875735,12/2/2011,6571,109.28,35
.84,718078.88,235504.64,482574.24\r\nCentral America and the Caribbean,Saint
Kitts and Nevis
,Clothes,Offline,H,9/8/2013,872412145,9/25/2013,4995,109.28,35.84,545853.60,
179020.80,366832.80\r\nSub-Saharan Africa,Sierra
Leone,Fruits,Offline,C,4/23/2012,627122199,4/29/2012,8250,9.33,6.92,76972.50
,57090.00,19882.50\r\nEurope,Russia,Fruits,Online,M,2/14/2011,103617227,3/12
/2011,1495,9.33,6.92,13948.35,10345.40,3602.95\r\nEurope,Lithuania,Vegetable
s,Offline,M,10/6/2010,423821055,10/22/2010,6923,154.06,90.93,1066557.38,6295
08.39,437048.99\r\nEurope,United
Kingdom,Vegetables,Online,L,12/8/2012,529970014,1/3/2013,8759,154.06,90.93,1

```

```

349411.54,796455.87,552955.67\r\nAsia,Indonesia,Personal
Care,Offline,H,8/17/2016,334612929,10/3/2016,8256,81.73,56.67,674762.88,4678
67.52,206895.36\r\nAsia,Mongolia,Beverages,Offline,M,4/23/2014,270611131,5/2
4/2014,8702,47.45,31.79,412909.90,276636.58,136273.32\r\nMiddle East and
North Africa,Egypt,Office
Supplies,Online,C,11/18/2010,841138446,12/8/2010,413,651.21,524.96,268949.73
,216808.48,52141.25\r\nSub-Saharan
Africa,Comoros,Household,Online,M,4/22/2012,369681203,5/9/2012,5738,668.27,5
02.54,3834533.26,2883574.52,950958.74\r\nEurope,Slovenia,Household,Offline,M
,4/5/2014,850038230,4/21/2014,4057,668.27,502.54,2711171.39,2038804.78,67236
6.61\r\nMiddle East and North
Africa,Lebanon,Beverages,Online,M,6/21/2013,296320855,7/13/2013,6781,47.45,3
1.79,321758.45,215567.99,106190.46\r\nAustralia and
Oceania,Australia,Cosmetics,Offline,L,7/13/2011,392952907,8/13/2011,2352,437
.20,263.33,1028294.40,619352.16,408942.24\r\nCentral America and the
Caribbean,Haiti,Snacks,Online,M,2/1/2011,644670712,3/21/2011,1245,152.58,97.
44,189962.10,121312.80,68649.30\r\nCentral America and the Caribbean,Saint
Kitts and Nevis
,Meat,Online,L,1/27/2012,626523101,2/16/2012,963,421.89,364.69,406280.07,351
196.47,55083.60\r\nMiddle East and North
Africa,Syria,Vegetables,Offline,M,6/24/2015,433871400,7/1/2015,1044,154.06,9
0.93,160838.64,94930.92,65907.72\r\nAsia,Laos,Snacks,Offline,M,1/3/2012,2323
89438,1/8/2012,8054,152.58,97.44,1228879.32,784781.76,444097.56\r\nCentral
America and the Caribbean,Saint Kitts and Nevis
,Cereal,Offline,H,2/21/2016,708063542,3/19/2016,592,205.70,117.11,121774.40,
69329.12,52445.28\r\nSub-Saharan
Africa,Sudan,Vegetables,Offline,H,12/4/2016,817192542,12/22/2016,4288,154.06
,90.93,660609.28,389907.84,270701.44\r\nCentral America and the
Caribbean,Guatemala,Meat,Offline,H,1/16/2012,936387765,2/29/2012,6803,421.89
,364.69,2870117.67,2480986.07,389131.60\r\nAsia,Brunei,Clothes,Offline,H,8/6
/2011,612573039,8/9/2011,2830,109.28,35.84,309262.40,101427.20,207835.20\r\n
Middle East and North
Africa,Jordan,Clothes,Online,M,8/12/2011,812984693,8/22/2011,9092,109.28,35.
84,993573.76,325857.28,667716.48\r\nCentral America and the
Caribbean,Panama,Meat,Offline,C,12/18/2012,775171554,1/5/2013,9344,421.89,36
4.69,3942140.16,3407663.36,534476.80\r\nSub-Saharan Africa,Central African
Republic,Household,Online,H,1/10/2010,256994950,2/19/2010,9372,668.27,502.54
,6263026.44,4709804.88,1553221.56\r\nMiddle East and North
Africa,Bahrain,Beverages,Offline,M,2/28/2017,886628711,3/31/2017,1993,47.45,
31.79,94567.85,63357.47,31210.38\r\nSub-Saharan
Africa,Burundi,Beverages,Online,C,12/8/2011,312559163,12/16/2011,2057,47.45,
31.79,97604.65,65392.03,32212.62\r\nEurope,Austria,Cereal,Online,L,8/18/2014
,753585135,9/13/2014,1443,205.70,117.11,296825.10,168989.73,127835.37\r\nAus
tralia and
Oceania,Fiji,Beverages,Offline,H,4/19/2016,448817956,4/22/2016,4062,47.45,31
.79,192741.90,129130.98,63610.92\r\nAustralia and
Oceania,Fiji,Clothes,Offline,M,11/6/2012,407681453,12/24/2012,856,109.28,35.
84,93543.68,30679.04,62864.64\r\nEurope,Switzerland,Snacks,Online,H,5/27/201
6,359911954,6/23/2016,4800,152.58,97.44,732384.00,467712.00,264672.00\r\nMid
dle East and North Africa,Yemen,Office
Supplies,Offline,M,8/18/2013,105558288,8/19/2013,5898,651.21,524.96,3840836.
58,3096214.08,744622.50\r\nSub-Saharan Africa,Comoros,Personal
Care,Offline,H,12/31/2014,864981782,2/11/2015,6186,81.73,56.67,505581.78,350
560.62,155021.16\r\nSub-Saharan Africa,Democratic Republic of the
Congo,Office
Supplies,Online,C,2/4/2013,328856265,2/12/2013,4732,651.21,524.96,3081525.72
,2484110.72,597415.00\r\nAsia,Mongolia,Clothes,Offline,H,9/23/2016,308168065

```

```

,10/18/2016,2633,109.28,35.84,287734.24,94366.72,193367.52\r\nAustralia and
Oceania,Palau,Household,Offline,L,11/1/2016,884216010,11/2/2016,8021,668.27,
502.54,5360193.67,4030873.34,1329320.33\r\nEurope,Monaco,Snacks,Offline,M,12
/27/2012,858611428,1/9/2013,1057,152.58,97.44,161277.06,102994.08,58282.98\r
\nAustralia and
Oceania,Fiji,Cereal,Online,L,3/10/2017,903278148,4/3/2017,8932,205.70,117.11
,1837312.40,1046026.52,791285.88\r\nSub-Saharan
Africa,Mali,Beverages,Online,L,3/17/2012,410452497,3/26/2012,870,47.45,31.79
,41281.50,27657.30,13624.20\r\nSub-Saharan
Africa,Liberia,Cereal,Offline,H,12/17/2015,642683303,1/20/2016,3126,205.70,1
17.11,643018.20,366085.86,276932.34\r\nEurope,Switzerland,Beverages,Offline,
L,2/18/2017,682831895,3/16/2017,3987,47.45,31.79,189183.15,126746.73,62436.4
2\r\nAustralia and Oceania,Samoa,Baby
Food,Online,L,11/5/2016,584072101,11/5/2016,8769,255.28,159.42,2238550.32,13
97953.98,840596.34\r\nAsia,Nepal,Meat,Offline,C,4/9/2017,919890248,5/18/2017
,4821,421.89,364.69,2033931.69,1758170.49,275761.20\r\nMiddle East and North
Africa,Azerbaijan,Snacks,Offline,C,4/18/2010,534085166,4/25/2010,6524,152.58
,97.44,995431.92,635698.56,359733.36\r\nEurope,Georgia,Baby
Food,Offline,H,8/1/2011,590768182,9/7/2011,288,255.28,159.42,73520.64,45912.
96,27607.68\r\nMiddle East and North Africa,United Arab
Emirates,Vegetables,Online,C,5/12/2011,524363124,6/28/2011,9556,154.06,90.93
,1472197.36,868927.08,603270.28\r\nEurope,Finland,Household,Offline,L,1/25/2
016,289606320,2/14/2016,9801,668.27,502.54,6549714.27,4925394.54,1624319.73\r
\nEurope,Portugal,Cereal,Offline,C,4/10/2014,811546599,5/8/2014,3528,205.70
,117.11,725709.60,413164.08,312545.52\r\n";
this.dataSource = GetCSVData(data);
}
}
public List<string[]> GetCSVData(string data)
{
List<string[]> splitted = new List<string[]>();
string[] tempString = data.Split(new[] { "\r\n", "\r", "\n" },
StringSplitOptions.None);
foreach (string item in tempString)
{
if (!string.IsNullOrEmpty(item))
{
splitted.Add(item.Split(','));
}
}
return splitted;
}
}

```

	* Baby Food			* Beverages			* Cereal			* Clothes	
	Total Cost	Total Revenue	Total Profit	Total Cost	Total Revenue	Total Profit	Total Cost	Total Revenue	Total Profit	Total Cost	Total Revenue
* Asia	\$8,097,238	\$14,267,177	\$5,349,947	\$2,883,076	\$4,984,788	\$1,321,794	\$1,625,748	\$6,719,808	\$2,894,856	\$870,554	\$2,84
* Australia and Oceania	\$1,435,182	\$5,000,773	\$2,565,591	\$1,396,310	\$1,536,261	\$540,951	\$5,066,568	\$8,886,212	\$3,825,646	\$699,967	\$1,86
* Central America and t	\$5,390,128	\$8,061,894	\$1,364,974	\$1,886,085	\$2,899,777	\$889,880	\$1,915,222	\$6,878,962	\$2,961,741	\$1,280,394	\$3,93
* Europe	\$20,565,688	\$32,031,888	\$12,366,228	\$3,858,141	\$6,937,562	\$1,949,011	\$11,161,214	\$19,666,764	\$8,435,640	\$3,669,586	\$11,18
* Middle East and North	\$8,192,684	\$13,116,838	\$4,926,245	\$2,267,962	\$3,385,178	\$1,117,216	\$5,686,582	\$10,865,965	\$4,389,263	\$2,819,533	\$8,59
* North America	\$2,686,968	\$4,142,428	\$1,455,520	\$228,045	\$329,636	\$100,798				\$1,031,547	\$3,14
* Sub-Saharan Africa	\$20,331,948	\$32,567,646	\$12,225,697	\$4,818,610	\$6,998,960	\$1,979,548	\$15,761,952	\$27,686,369	\$11,923,417	\$3,295,886	\$8,92

In the meantime, the CSV data from the local *. csv file type can also be connected to the pivot table. Here, the file can be read by the **StreamReader** option, which will give the result in the string form. And

the resulting string needs to be converted to string array that can be assigned to the [DataSource](#) property under [PivotViewDataSourceSettings](#).

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
@using System;
@using System.Collections.Generic;
@using System.IO;
@using System.Net;

<SfPivotView TValue="string[]" Width="1500" Height="300">
  <PivotViewDataSourceSettings TValue="string[]" DataSource="@dataSource"
  ExpandAll=false EnableSorting=true Type=DataSourceType.CSV>
    <PivotViewColumns>
      <PivotViewColumn Name="Item Type"></PivotViewColumn>
      <PivotViewColumn Name="Sales Channel"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Region"></PivotViewRow>
      <PivotViewRow Name="Country"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Total Cost"></PivotViewValue>
      <PivotViewValue Name="Total Revenue"></PivotViewValue>
      <PivotViewValue Name="Total Profit"></PivotViewValue>
    </PivotViewValues>
    <PivotViewFormatSettings>
      <PivotViewFormatSetting Name="Total Cost" Format="C0"
      UseGrouping=true></PivotViewFormatSetting>
      <PivotViewFormatSetting Name="Total Revenue" Format="C0"
      UseGrouping=true></PivotViewFormatSetting>
      <PivotViewFormatSetting Name="Total Profit" Format="C0"
      UseGrouping=true></PivotViewFormatSetting>
    </PivotViewFormatSettings>
  </PivotViewDataSourceSettings>
  <PivotViewGridSettings ColumnWidth="120"></PivotViewGridSettings>
</SfPivotView>

@code{
public List<string[]> dataSource { get; set; }
protected override void OnInitialized()
{
  // Put appropriate file path here
  string url = AppDomain.CurrentDomain.BaseDirectory + "sales.csv";
  WebClient myWebClient = new WebClient();
  Stream myStream = myWebClient.OpenRead(url);
  StreamReader stream = new StreamReader(myStream);
  string result = stream.ReadToEnd();
  stream.Close();
  this.dataSource = GetCSVData(result);
}
public List<string[]> GetCSVData(string data)
{
  List<string[]> splitted = new List<string[]>();
  string[] tempString = data.Split(new[] { "\r\n", "\r", "\n" },
  StringSplitOptions.None);
  foreach (string item in tempString)
  {
```

```

if (!string.IsNullOrEmpty(item))
{
    splitted.Add(item.Split(','));
}
}
return splitted;
}
}

```

	* Baby Food			* Beverages			* Cereal			* Clothes	
	Total Cost	Total Revenue	Total Profit	Total Cost	Total Revenue	Total Profit	Total Cost	Total Revenue	Total Profit	Total Cost	Total Revenue
▶ Asia	\$0,097,230	\$14,267,177	\$5,340,947	\$2,883,076	\$4,984,786	\$1,321,794	\$1,625,749	\$6,719,808	\$2,894,856	\$370,554	\$2,84
▶ Australia and Oceania	\$3,435,162	\$5,000,773	\$2,065,591	\$1,396,310	\$1,636,261	\$540,251	\$5,060,669	\$8,885,212	\$3,829,646	\$999,997	\$1,86
▶ Central America and L.	\$5,599,120	\$8,981,394	\$3,384,974	\$1,886,085	\$2,999,777	\$809,890	\$1,915,222	\$6,876,962	\$2,961,741	\$1,290,384	\$3,93
▶ Europe	\$20,505,650	\$32,931,888	\$12,366,228	\$3,858,141	\$6,907,962	\$1,949,811	\$11,161,314	\$39,666,763	\$20,438,640	\$3,669,586	\$11,18
▶ Middle East and North	\$8,192,584	\$13,116,839	\$4,926,245	\$2,267,962	\$3,385,176	\$1,117,216	\$5,666,582	\$10,885,965	\$4,389,263	\$2,819,533	\$6,59
▶ North America	\$2,686,960	\$4,142,428	\$1,555,520	\$229,045	\$329,636	\$100,798				\$1,031,547	\$3,14
▶ Sub-Saharan Africa	\$20,331,949	\$32,567,545	\$12,225,697	\$4,918,510	\$6,998,860	\$1,979,549	\$15,761,952	\$27,686,369	\$11,923,417	\$3,255,686	\$9,92

Binding CSV data via remote

In-order to bind the remote CSV data, mention the endpoint [Url](#) under [PivotViewDataSourceSettings](#) property. The [Url](#) property supports both direct downloadable file (*.csv) and web service URL.

ASPX-CS

```

@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="PivotProductDetails" Width="1500" Height="300">
<PivotViewDataSourceSettings TValue="PivotProductDetails" ExpandAll=false
EnableSorting=true Url="https://bi.syncfusion.com/productservice/api/sales"
Type=DataSourceType.CSV>
<PivotViewColumns>
<PivotViewColumn Name="Item Type"></PivotViewColumn>
<PivotViewColumn Name="Sales Channel"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Region"></PivotViewRow>
<PivotViewRow Name="Country"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Total Cost"></PivotViewValue>
<PivotViewValue Name="Total Revenue"></PivotViewValue>
<PivotViewValue Name="Total Profit"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="Total Cost" Format="C0"
UseGrouping=true></PivotViewFormatSetting>
<PivotViewFormatSetting Name="Total Revenue" Format="C0"
UseGrouping=true></PivotViewFormatSetting>
<PivotViewFormatSetting Name="Total Profit" Format="C0"
UseGrouping=true></PivotViewFormatSetting>
</PivotViewFormatSettings>
</PivotViewDataSourceSettings>
<PivotViewGridSettings ColumnWidth="120"></PivotViewGridSettings>
</SfPivotView>
@code{
public class PivotProductDetails
{

```

```

public int Quantity { get; set; }
public double UnitPrice { get; set; }
public string ProductName { get; set; }
public string ShipCountry { get; set; }
public string ShipCity { get; set; }
}

```

	* Baby Food			* Beverages			* Cereal			* Clothes	
	Total Cost	Total Revenue	Total Profit	Total Cost	Total Revenue	Total Profit	Total Cost	Total Revenue	Total Profit	Total Cost	Total Revenue
Asia	\$8,097,238	\$14,267,177	\$5,349,947	\$2,883,076	\$4,984,788	\$1,321,794	\$1,625,749	\$6,719,808	\$2,894,856	\$878,554	\$2,84
Australia and Oceania	\$3,435,182	\$5,000,773	\$2,865,591	\$1,396,310	\$1,636,261	\$540,951	\$5,066,969	\$8,885,212	\$3,825,646	\$999,997	\$1,86
Central America and L	\$5,595,128	\$8,981,894	\$3,384,974	\$1,888,085	\$2,895,777	\$889,880	\$1,915,222	\$6,878,962	\$2,961,741	\$1,290,384	\$3,93
Europe	\$20,505,658	\$32,931,888	\$12,366,228	\$3,858,141	\$6,907,562	\$1,949,811	\$11,151,214	\$19,666,763	\$8,435,640	\$3,669,586	\$11,18
Middle East and North	\$8,192,584	\$13,118,838	\$4,926,245	\$2,267,962	\$3,385,178	\$1,117,216	\$5,686,582	\$10,855,955	\$4,389,263	\$2,819,533	\$6,59
North America	\$2,586,968	\$4,142,428	\$1,555,459	\$228,045	\$329,836	\$100,798				\$1,031,547	\$3,14
Sub-Saharan Africa	\$820,334,949	\$32,957,545	\$12,229,697	\$4,818,510	\$6,998,960	\$1,979,549	\$15,761,952	\$27,685,369	\$11,923,417	\$3,295,586	\$8,92

Remote Data Binding

To interact with remote data source, provide the endpoint [Url](#) within [SfDataManager](#) class along with appropriate [Adaptor](#). By default, [SfDataManager](#) uses [ODataAdaptor](#) for remote data-binding.

When using [SfDataManager](#) for data binding then the TValue must be provided explicitly in the [PivotViewDataSourceSettings](#) class.

Binding with OData services

OData is a standardized protocol for creating and consuming data. User can retrieve data from OData service using the [SfDataManager](#) class.

ASPX-CS

```

@using Syncfusion.Blazor.PivotView
@using Syncfusion.Blazor.Data
<SfPivotView TValue="OrderDetails" Width="800" Height="340">
  <PivotViewDataSourceSettings TValue="OrderDetails">
    <SfDataManager
      Url="https://js.syncfusion.com/ejServices/Wcf/Northwind.svc/Orders"
      Adaptor="Syncfusion.Blazor.Adaptors.ODataAdaptor"></SfDataManager>
    <PivotViewColumns>
      <PivotViewColumn Name="OrderDate"></PivotViewColumn>
      <PivotViewColumn Name="ShipCity"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="OrderID"></PivotViewRow>
      <PivotViewRow Name="CustomerID"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Freight"></PivotViewValue>
    </PivotViewValues>
  </PivotViewDataSourceSettings>
</SfPivotView>
@code{
  public class OrderDetails
  {
    public int OrderID { get; set; }
    public string OrderDate { get; set; }
    public string CustomerID { get; set; }
  }
}

```

```
public string ShipCity { get; set; }
public double Freight { get; set; }
}
```

Binding with OData V4 services

The OData V4 is an improved version of OData protocols, and the [SfDataManager](#) class can be used to retrieve and consume OData V4 services. For more details on OData V4 services, refer to the [OData documentation](#). To bind OData V4 service, use the [ODataV4Adaptor](#).

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
@using Syncfusion.Blazor.Data
<SfPivotView TValue="OrderDetails" Width="800" Height="340">
  <PivotViewDataSourceSettings TValue="OrderDetails">
    <SfDataManager
      Url="http://services.odata.org/V4/Northwind/Northwind.svc/Orders/"
      Adaptor="Syncfusion.Blazor.Adaptors.ODataV4Adaptor"></SfDataManager>
    <PivotViewColumns>
      <PivotViewColumn Name="OrderDate"></PivotViewColumn>
      <PivotViewColumn Name="ShipCity"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="OrderID"></PivotViewRow>
      <PivotViewRow Name="CustomerID"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Freight"></PivotViewValue>
    </PivotViewValues>
  </PivotViewDataSourceSettings>
</SfPivotView>
@code{
public class OrderDetails
{
  public int OrderID { get; set; }
  public string OrderDate { get; set; }
  public string CustomerID { get; set; }
  public string ShipCity { get; set; }
  public double Freight { get; set; }
}
```

Web API

User can use [WebApiAdaptor](#) to bind the pivot table with Web API created using OData endpoint.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
@using Syncfusion.Blazor.Data
<SfPivotView TValue="ProductDetails">
  <PivotViewDataSourceSettings TValue="ProductDetails" ExpandAll="true"
    EnableSorting="true">
    <SfDataManager Url="https://bi.syncfusion.com/northwindservice/api/orders"
      Adaptor="Syncfusion.Blazor.Adaptors.WebApiAdaptor"></SfDataManager>
  </PivotViewDataSourceSettings>
</SfPivotView>
```

```

<PivotViewColumns>
<PivotViewColumn Name="ProductName"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="ShipCountry"></PivotViewRow>
<PivotViewRow Name="ShipCity"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Quantity" Caption="Quantity"></PivotViewValue>
<PivotViewValue Name="UnitPrice" Caption="Unit Price"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="UnitPrice"
Format="C"></PivotViewFormatSetting>
</PivotViewFormatSettings>
</PivotViewDataSourceSettings>
</SfPivotView>
@code{
public class ProductDetails
{
public int Quantity { get; set; }
public double UnitPrice { get; set; }
public string ProductName { get; set; }
public string ShipCountry { get; set; }
public string ShipCity { get; set; }
}
}

```

List binding

ExpandoObject

The [Blazor Pivot Table](#) supports **ExpandoObject** data source when the model type is unknown. This can be set in [DataSource](#) property under [PivotViewDataSourceSettings](#) class. Moreover, the data source supports all the available features in the component.

ASPX-CS

```

@using Syncfusion.Blazor.PivotView
@using System.Dynamic
<SfPivotView TValue="ExpandoObject" Height="560" Width="100%">
<PivotViewDataSourceSettings DataSource="@Orders">
<PivotViewColumns>
<PivotViewColumn Name="OrderID"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="CustomerID"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="OrderDate"></PivotViewValue>
<PivotViewValue Name="Freight" Type="SummaryTypes.Max"></PivotViewValue>
</PivotViewValues>
</PivotViewDataSourceSettings>
</SfPivotView>
@code{
public List<ExpandoObject> Orders { get; set; } = new List<ExpandoObject>();
protected override void OnInitialized()

```



```

{
    Orders = Enumerable.Range(1, 75).Select((x) =>
    {
        dynamic d = new ExpandoObject();
        d.OrderID = 1000 + x;
        d.CustomerID = (new string[] { "ALFKI", "ANANTR", "ANTON", "BLONP", "BOLID"
        })[new Random().Next(5)];
        d.Freight = (new double[] { 2, 1, 4, 5, 3 })[new Random().Next(5)] * x;
        d.OrderDate = (new DateTime[] { new DateTime(2010, 11, 5), new
        DateTime(2018, 10, 3), new DateTime(1995, 9, 9), new DateTime(2012, 8, 2),
        new DateTime(2015, 4, 11) })[new Random().Next(5)];
        d.ShipCountry = (new string[] { "USA", "UK" })[new Random().Next(2)];
        d.Verified = (new bool[] { true, false })[new Random().Next(2)];
        return d;
    }).Cast<ExpandoObject>().ToList<ExpandoObject>();
}
}

```

	1001		1002		1003		1004		1005		1006		1007
	OrderDate	Freight	OrderDate	Freight	OrderDate	Freight	OrderDate	Freight	OrderDate	Freight	OrderDate	Freight	OrderDate
ALFKI													
ANANTR									1	10			
ANTON	1	4											
BLONP					1	5					1	8	
BOLID			1	6			1	20					
Grand Total	1	4	1	6	1	5	1	20	1	10	1	8	

DynamicObject

The [Blazor Pivot Table](#) supports **DynamicObject** data source when the model type is unknown. This can be set in [DataSource](#) property under [PivotViewDataSourceSettings](#) class. Moreover, the data source supports all the available features in the component.

ASPX-CS

```

@using Syncfusion.Blazor.PivotView
@using System.Dynamic
<SfPivotView TValue="DynamicDictionary" Height="560" Width="100%">
    <PivotViewDataSourceSettings DataSource="@Orders">
        <PivotViewColumns>
            <PivotViewColumn Name="OrderID"></PivotViewColumn>
        </PivotViewColumns>
        <PivotViewRows>
            <PivotViewRow Name="CustomerID"></PivotViewRow>
        </PivotViewRows>
        <PivotViewValues>
            <PivotViewValue Name="OrderDate"></PivotViewValue>
            <PivotViewValue Name="Freight" Type="SummaryTypes.Max"></PivotViewValue>
        </PivotViewValues>
    </PivotViewDataSourceSettings>
</SfPivotView>
@code{
    private List<string> ToolbarItems = new List<string>() { "Add", "Edit",
    "Delete", "Update", "Cancel" };
    public List<DynamicDictionary> Orders = new List<DynamicDictionary>() { };
    protected override void OnInitialized()
    {
        Orders = Enumerable.Range(1, 1075).Select((x) =>

```

```

{
dynamic d = new DynamicDictionary();
d.OrderID = 1000 + x;
d.CustomerID = (new string[] { "ALFKI", "ANANTR", "ANTON", "BLONP", "BOLID"
})[new Random().Next(5)];
d.Freight = (new double[] { 2, 1, 4, 5, 3 })[new Random().Next(5)] * x;
d.OrderDate = DateTime.Now.AddDays(-x);
return d;
}).Cast<DynamicDictionary>().ToList<DynamicDictionary>();
}
public class DynamicDictionary : DynamicObject
{
Dictionary<string, object> dictionary = new Dictionary<string, object>();
public override bool TryGetMember(GetMemberBinder binder, out object result)
{
string name = binder.Name;
return dictionary.TryGetValue(name, out result);
}
public override bool TrySetMember(SetMemberBinder binder, object value)
{
dictionary[binder.Name] = value;
return true;
}
public override System.Collections.Generic.IEnumerable<string>
GetDynamicMemberNames()
{
return this.dictionary?.Keys;
}
}
}

```

	1001		1002		1003		1004		1005		1006		1007
	OrderDate	Freight	OrderDate	Freight	OrderDate	Freight	OrderDate	Freight	OrderDate	Freight	OrderDate	Freight	OrderDate
ALFKI							1	12					
ANANTR					1	12					1	12	
ANTON									1	10			
BLONP													
BOLID	1	2	1	3									
Grand Total	1	2	1	3	1	12	1	12	1	10	1	12	

Mapping

One can define field information like alias name (caption), data type, aggregation type, show and hide subtotals etc. using the [FieldMapping](#) property under [PivotViewDataSourceSettings](#) class. The available options are,

- [Name](#) - It is to specify the appropriate field name.
- [Caption](#) - It is to set the alias name (caption) to the specific field. Instead of actual field name, the alias name (caption) will be set in the UI of the pivot table.
- [Type](#) - It is to display values in the pivot table with appropriate aggregation such as sum, product, count, average, minimum, maximum, etc. Its default value is **sum**. This option is applicable only for relational data source.
- [Axis](#) - It will help to display the field in specified axis such as row/column/value/filter axis of the pivot table.

- [ShowNoDataItems](#) - It is to show all the members of a specific field to the pivot table, even if there are no data in the intersection of the row and column. The default value is **false**. This option is applicable only for relational data source.
- [BaseField](#) - For the aggregate types like "DifferenceFrom" or "PercentageOfDifferenceFrom" or "PercentageOfParentTotal", selective field is assigned for comparison via this property.
- [BaseItem](#) For the aggregate types like "DifferenceFrom" or "PercentageOfDifferenceFrom" or "PercentageOfParentTotal", selective member in a field is assigned for comparison via this property.
- [ShowSubTotals](#) - It is to show or hide sub-totals of a specific field in row and column axis of the pivot table. The default value is **true**.
- [IsNamedSet](#) - It is to set whether the specified field is named set or not. In general, the named set is a set of dimension members or a set expression (MDX query) to be created as a dimension in the SSAS OLAP cube itself. The default value is **false** and this option is applicable only for OLAP data source.
- [IsCalculatedField](#) - It is to set whether the specified field is a calculated field or not. In general, a calculated field is created from the bound data source or using simple formula with basic arithmetic operators in the pivot table. The default value is **false** and this option is applicable only for OLAP data source.
- [ShowFilterIcon](#) - It is to show or hide the filter icon of a specific field which will be displayed on the button of the grouping bar and field list UI. This filter icon is used to filter the members of a specified field at runtime in the pivot table. The default value is **true**.
- [ShowSortIcon](#) - It is to show or hide the sort icon of a specific field which will be displayed on the button of the grouping bar and field list UI. This sort icon is used to order members of a specified field either in ascending or descending at runtime. The default value is **true**.
- [ShowRemoveIcon](#) - It is to show or hide the remove icon of a specific field which will be displayed on the button of the grouping bar and field list UI. This remove icon is used to remove the specified field during runtime. The default value is **true**.
- [ShowValueTypeIcon](#) - It is to show or hide the value type icon of a specific field which will be displayed on the button of the grouping bar and field list UI. This value type icon helps to select the appropriate aggregation type to specified value field at runtime. The default value is **true**.
- [ShowEditIcon](#) - It is to show or hide the edit icon of a specific field which will be displayed on the button of the grouping bar and field list UI. This edit icon is used to modify caption, formula, and format of a specified calculated field at runtime. The default value is **true**.
- [AllowDragAndDrop](#) - It is to restrict specific field's button from being dragged on runtime in the grouping bar and field list UI. This will prevent from altering the current report. The default value is **true**.
- [DataType](#) - It is to specify the type of the field like 'string', 'number', 'datetime', 'date', and 'boolean'.

The main purpose of these mapping options is to configure each field that is not a part of the initial pivot report. Even if any field that is a part of this mapping is defined here, the value set in the initial report will have the highest preceding.

This option is applicable only for relational data source.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails" ShowGroupingBar="true"
ShowFieldList="true">
```

```

<PivotViewDataSourceSettings DataSource="@dataSource">
  <PivotViewColumns>
    <PivotViewColumn Name="Year"></PivotViewColumn>
  </PivotViewColumns>
  <PivotViewRows>
    <PivotViewRow Name="Country"></PivotViewRow>
  </PivotViewRows>
  <PivotViewValues>
    <PivotViewValue Name="Sold" Caption="Units Sold"></PivotViewValue>
  </PivotViewValues>
  <PivotViewFormatSettings>
    <PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
  </PivotViewFormatSettings>
  <PivotViewFieldMapping>
    <PivotViewField Name="Quarter" ShowSortIcon="false"></PivotViewField>
    <PivotViewField Name="Products" ShowFilterIcon="false"
    ShowRemoveIcon="false"></PivotViewField>
    <PivotViewField Name="Amount" ShowValueTypeIcon="false" Caption="Sold
    Amount"></PivotViewField>
  </PivotViewFieldMapping>
</PivotViewDataSourceSettings>
</SfPivotView>
@code{
public List<ProductDetails> dataSource { get; set; }
protected override void OnInitialized()
{
this.dataSource = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
}

```

The screenshot shows a Blazor Pivot Table component. At the top, there are two filter buttons: "Sales of Units Sold" and "Year". Below these, there are two more filter buttons: "Country" and "Products". The main table displays data for the years FY 2015, FY 2016, FY 2017, FY 2018, and a Grand Total column. The rows represent different countries: France, Germany, and United States, along with a Grand Total row. The data values are displayed in the table cells.

	FY 2015	FY 2016	FY 2017	FY 2018	Grand Total
France	458	528		592	1584
Germany	448	495		372	1404
United States	546	636		731	1909
Grand Total	1452	1659		1695	4806

Entity Framework

Entity Framework acts as a modern object-database mapper for .NET. This section explains how to consume data from the Microsoft SQL Server database and bind it to the pivot table component.

Create DbContext class

The first step is to create a DbContext class called **OrderContext** for establishing connection to a Microsoft SQL Server database.

C#

```

using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

```

```

using EFPivot.Data;
namespace EFPivot.Data
{
    public class OrderContext : DbContext
    {
        public virtual DbSet<Order> Orders { get; set; }
        protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
        {
            if (!optionsBuilder.IsConfigured)
            {
                // Configures the context to connect to a Microsoft SQL Serve database
                optionsBuilder.UseSqlServer(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename='D:\blazor\EFPivot\App_Data\N
ORTHWND.MDF';Integrated Security=True;Connect Timeout=30");
            }
        }
    }
    public class Order
    {
        [Key]
        public int? SoldAmount { get; set; }
        [Required]
        public string Year { get; set; }
        [Required]
        public string Products { get; set; }
    }
}

```

Create data access layer to perform data operation

Now, there is a need to create a class called **OrderDataAccessLayer**, which acts as data access layer to retrieve the records from the database table.

C#

```

using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using EFPivot.Data;
namespace EFPivot.Data
{
    public class OrderDataAccessLayer
    {
        OrderContext db = new OrderContext();
        //To get all order details
        public DbSet<Order> GetAllOrders()
        {
            try
            {
                return db.Orders;
            }
            catch
            {
                throw;
            }
        }
    }
}

```

```

}
}
}
}

```

Creating Web API controller

A Web API controller must be created which allows the pivot table to directly consume data from the Entity Framework.

C#

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Primitives;
using EFPivot.Data;
namespace EFPivot.Controller
{
    [Route("api/[controller]")]
    [ApiController]
    public class DefaultController : ControllerBase
    {
        OrderDataAccessLayer db = new OrderDataAccessLayer();
        [HttpGet]
        public object Get()
        {
            IQueryable<Order> data = db.GetAllOrders().AsQueryable();
            return new { Items = data, Count = data.Count() };
        }
    }
}

```

Add Web API controller services in Startup.cs

Open the **Startup.cs** file and add services and endpoints required for Web API controller.

C#

```

using Newtonsoft.Json.Serialization;
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void ConfigureServices(IServiceCollection services)
        {
            ....
            ....
            services.AddSingleton<OrderDataAccessLayer>();
            // Adds services for controllers to the specified
            Microsoft.Extensions.DependencyInjection.IServiceCollection.
            services.AddControllers().AddNewtonsoftJson(options =>

```

```

{
options.SerializerSettings.ContractResolver = new DefaultContractResolver();
});
}
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
....
....
app.UseEndpoints(endpoints =>
{
// Adds endpoints for controller actions to the
Microsoft.AspNetCore.Routing.IEndpointRouteBuilder
endpoints.MapDefaultControllerRoute();
.....
.....
});
}
}
}
}

```

Configure pivot table component

Configure the pivot table to bind data either by using the [DataSource](#) property or [SfDataManager](#) class under the [PivotViewDataSourceSettings](#) class.

For instance, to bind data directly from the data access layer class **OrderDataAccessLayer**, assign the [DataSource](#) property to be **OrderData.GetAllOrders()** under [PivotViewDataSourceSettings](#) in the pivot table.

ASPX-CS

```

@inject OrderDataAccessLayer OrderData
@using EFPivot.Data
@using Syncfusion.Blazor.PivotView
<SfPivotView Height="340" TValue="Order">
<PivotViewDataSourceSettings DataSource="@OrderData.GetAllOrders()"
ExpandAll=false EnableSorting=true>
<PivotViewColumns>
<PivotViewColumn Name="Year"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Products"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="SoldAmount" Caption="Sold Amount"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="SoldAmount"
Format="C"></PivotViewFormatSetting>
</PivotViewFormatSettings>
</PivotViewDataSourceSettings>
</SfPivotView>

```

On the other hand, to configure the pivot table using Web API, provide the appropriate endpoint [Url](#) within the [SfDataManager](#) along with [Adaptor](#) type under [PivotViewDataSourceSettings](#) class. Here,

there is a need to use [WebApiAdaptor](#) in-order to interact with the Web API to consume data from the Entity Framework appropriately.

ASPX-CS

```
@using EFPivot.Data
@using Syncfusion.Blazor
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="Order" Height="340">
  <PivotViewDataSourceSettings TValue="Order" ExpandAll=false
  EnableSorting=true>
    <SfDataManager Url="api/Default"
    Adaptor="Adaptors.WebApiAdaptor"></SfDataManager>
  </PivotViewDataSourceSettings>
  <PivotViewColumns>
    <PivotViewColumn Name="Year"></PivotViewColumn>
  </PivotViewColumns>
  <PivotViewRows>
    <PivotViewRow Name="Products"></PivotViewRow>
  </PivotViewRows>
  <PivotViewValues>
    <PivotViewValue Name="SoldAmount" Caption="Sold Amount"></PivotViewValue>
  </PivotViewValues>
  <PivotViewFormatSettings>
    <PivotViewFormatSetting Name="SoldAmount"
    Format="C"></PivotViewFormatSetting>
  </PivotViewFormatSettings>
</PivotViewDataSourceSettings>
</SfPivotView>
```

Values in row axis

By default, the value fields are plotted in column axis. To plot those fields in row axis, use the [ValueAxis](#) property by setting its value as **row**. By default, it holds the value **column**.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
  <PivotViewDataSourceSettings DataSource="@data" ValueAxis="row">
    <PivotViewColumns>
      <PivotViewColumn Name="Year"></PivotViewColumn>
      <PivotViewColumn Name="Quarter"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Country"></PivotViewRow>
      <PivotViewRow Name="Products"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
      <PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
    </PivotViewValues>
    <PivotViewFormatSettings>
      <PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
    </PivotViewFormatSettings>
  </PivotViewDataSourceSettings>
</SfPivotView>
@code{
```



```

public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
    this.data = ProductDetails.GetProductData().ToList();
    //Bind the data source collection here. Refer "Assigning sample data to the
    pivot table" section in getting started for more details.
}
}

```

	FY 2015	FY 2016	FY 2017	FY 2018	Grand Total
France					
Units Sold	450	526	592	16	
Sold Amount	\$714,955	\$1,542,104	\$2,903,308	\$27,264	\$
Germany					
Units Sold	440	496	372	96	
Sold Amount	\$563,515	\$1,772,104	\$1,634,808	\$77,264	\$
United States					

Show 'no data' items

By default, the pivot table only shows the field item if it has data in its row or column combination. To show all items that do not have data in row and column combination in the pivot table, use the [ShowNoDataItems](#) property by setting its value to **true** for the desired fields. In the following code sample, rows of the "Country" and "Products" fields do not have data in all combination with "Year" and "Quarter" column field.

ASPX-CS

```

@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
  <PivotViewDataSourceSettings DataSource="@data" ExpandAll="true">
    <PivotViewColumns>
      <PivotViewColumn Name="Year" ShowNoDataItems="true"></PivotViewColumn>
      <PivotViewColumn Name="Quarter" ShowNoDataItems="true"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Country" ShowNoDataItems="true"></PivotViewRow>
      <PivotViewRow Name="Products" ShowNoDataItems="true"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
      <PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
    </PivotViewValues>
    <PivotViewFormatSettings>
      <PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
    </PivotViewFormatSettings>
  </PivotViewDataSourceSettings>
</SfPivotView>
@code{

```

```

public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
    this.data = ProductDetails.GetProductData().ToList();
    //Bind the data source collection here. Refer "Assigning sample data to the
    pivot table" section in getting started for more details.
}
}

```

	FY 2018				
	Q1		Q2		Q3
	Units Sold	Sold Amount	Units Sold	Sold Amount	Units
France	16	\$27,264			
Mountain Bikes	16	\$27,264			
Road Bikes					
Germany	96	\$77,264			
Mountain Bikes	96	\$77,264			

Show value headers always

To show value header always in pivot table, even if it holds a single value, use the [AlwaysShowValueHeader](#) property by setting its value as **true**.

ASPX-CS

```

@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
  <PivotViewDataSourceSettings DataSource="@data"
  AlwaysShowValueHeader="true">
    <PivotViewColumns>
      <PivotViewColumn Name="Year"></PivotViewColumn>
      <PivotViewColumn Name="Quarter"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Country"></PivotViewRow>
      <PivotViewRow Name="Products"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
    </PivotViewValues>
    <PivotViewFormatSettings>
      <PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
    </PivotViewFormatSettings>
  </PivotViewDataSourceSettings>
</SfPivotView>
@code{
    public List<ProductDetails> data { get; set; }
    protected override void OnInitialized()
    {
        this.data = ProductDetails.GetProductData().ToList();
    }
}

```

```
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
}
```

	► FY 2015	► FY 2016	► FY 2017	► FY 2018	Grand Total
	Sold Amount	Sold Amount	Sold Amount	Sold Amount	Sold Amount
► France	\$714,955	\$1,542,104	\$2,903,308	\$27,264	\$5,1
► Germany	\$563,515	\$1,772,104	\$1,634,808	\$77,264	\$4,0
► United States	\$754,515	\$2,263,104	\$3,387,308	\$97,264	\$6,5
Grand Total	\$2,032,985	\$5,577,312	\$7,925,424	\$201,792	\$15,7

Customize empty value cells

User can show custom string in empty value cells using the [EmptyCellsTextContent](#) property in [PivotViewDataSourceSettings](#) class of the pivot table. Since the property is of string data type, user can fill empty value cells with any value like "0", "-", "*", "(blank)", etc. Its common for all value fields and can be configured through code behind.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
  <PivotViewDataSourceSettings DataSource="@data" EmptyCellsTextContent="*"
  ExpandAll="true">
    <PivotViewColumns>
      <PivotViewColumn Name="Year"></PivotViewColumn>
      <PivotViewColumn Name="Quarter"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Country"></PivotViewRow>
      <PivotViewRow Name="Products"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Sold" Caption="Units Sold"></PivotViewValue>
      <PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
    </PivotViewValues>
  </PivotViewDataSourceSettings>
</SfPivotView>

@code{
  public List<ProductDetails> data { get; set; }
  protected override void OnInitialized()
  {
    this.data = ProductDetails.GetProductData().ToList();
    //Bind the data source collection here. Refer "Assigning sample data to the
    pivot table" section in getting started for more details.
  }
}
```

	FY 2015				
	Q1		Q2		Q3
	Units Sold	Sold Amount	Units Sold	Sold Amount	Units Sold
France	114	\$177,248	**	**	110
Mountain Bikes	31	\$52,824	**	**	90
Road Bikes	83	\$124,422	**	**	20
Germany	74	\$117,248	128	\$152,352	140
Mountain Bikes	51	\$92,824	61	\$76,904	70

Event

The event [OnLoad](#) fires before initiate rendering of pivot table. It holds following parameters like [DataSourceSettings](#), [FieldsType](#) and [PivotView](#). In this event user can customize data source settings before initiating pivot table render module.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails" ShowGroupingBar="true">
  <PivotViewDataSourceSettings DataSource="@data" EmptyCellsTextContent="*"
  ExpandAll="true">
    <PivotViewColumns>
      <PivotViewColumn Name="Year"></PivotViewColumn>
      <PivotViewColumn Name="Quarter"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Country"></PivotViewRow>
      <PivotViewRow Name="Products"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Sold" Caption="Units Sold"></PivotViewValue>
      <PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
    </PivotViewValues>
    <PivotViewFormatSettings>
      <PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
    </PivotViewFormatSettings>
    <PivotViewEvents TValue="ProductDetails" OnLoad="load"></PivotViewEvents>
  </PivotViewDataSourceSettings>
</SfPivotView>

@code{
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
  this.data = ProductDetails.GetProductData().ToList();
  //Bind the data source collection here. Refer "Assigning sample data to the
  pivot table" section in getting started for more details.
}
public void load(LoadEventArgs<ProductDetails> args)
{
  args.DataSourceSettings.EmptyCellsTextContent = "###";
  args.DataSourceSettings.Columns[0].Caption = "Full Year";
  args.DataSourceSettings.ExpandAll = false;
}
```

```
}
}
```

Sum of Units Sold	Drop filter here			
Sum of Sold Amount	Year Quarter			
Country	FY 2015		FY 2016	
Products	Units Sold	Sold Amount	Units Sold	Sold Amount
France	450	\$714,955	526	\$1,542,104
Germany	440	\$563,515	496	\$1,772,104
United States	546	\$754,515	636	\$2,263,104
Grand Total	1436	\$2,032,985	1658	\$5,577,312

You can also explore our [Blazor Pivot Table example](#) to know how to render and configure the pivot table.

```
<!-- markdownlint-disable MD024 -->
```

```
<!-- markdownlint-disable MD012 -->
```

OLAP in Blazor Pivot Table Component

Getting Started

This section briefly explains about how to create a simple [Blazor Pivot Table](#) with OLAP data source. You can refer [Getting Started with Syncfusion Blazor for Client-Side in Visual Studio 2019 Preview](#) page for the introduction and configuring the common specifications.

Importing Syncfusion Blazor component package in the application

Install **Syncfusion.Blazor** NuGet package to the application by using the **NuGet Package Manager**. Please ensure to check the Include **prerelease** option.

Initializing pivot table component in an application

The Syncfusion Pivot Table component can be initialized in any razor page inside `~/Pages` folder. Here, the pivot table component is initialized inside `~/Pages/Index.razor` page. In a new application, if `Index.razor` page has any default content template, then those content can be completely removed and following code can be added.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails"></SfPivotView>
```

Assigning olap data to the pivot table

The [Blazor Pivot Table](#) component further needs to be populated with an appropriate data source. For illustration purpose, a collection of objects mentioning the sales details of certain products over a period and region has been prepared. This sample data is assigned to the pivot table component through [PivotViewDataSourceSettings](#) class.

Refer [here](#) to know the more details about OLAP data binding.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
<PivotViewDataSourceSettings TValue="ProductDetails"
ProviderType="ProviderType.SSAS" Catalog="Adventure Works DW 2008
SE"Cube="Adventure Works" Url="https://bi.syncfusion.com/olap/msmdpump.dll"
LocaleIdentifier="1033"EnableSorting="true"></PivotViewDataSourceSettings>
</SfPivotView>
```

Adding OLAP cube elements to row, column, value and filter axes

Now that pivot table is initialized and assigned with sample OLAP data source, will further move to showcase the component by organizing appropriate [OLAP cube elements](#) in [PivotViewRows](#), [PivotViewColumns](#), [PivotViewValues](#) and [PivotViewFilters](#) axes.

In [PivotViewDataSourceSettings](#) class, four major axes - [PivotViewRows](#), [PivotViewColumns](#), [PivotViewValues](#) and [PivotViewFilters](#) plays a vital role in defining and organizing [OLAP cube elements](#) from the bound data source, to render the entire pivot table component in a desired format.

[PivotViewRows](#) – – Collection of [OLAP cube elements](#) (such as Hierarchies, NamedSet, Calculated Members etc.,) that needs to be displayed in row axis of the pivot table.

[PivotViewColumns](#) – – Collection of [OLAP cube elements](#) (such as Hierarchies, NamedSet, Calculated Members etc.,) that needs to be displayed in column axis of the pivot table.

[PivotViewValues](#) – – Collection of [OLAP cube elements](#) (such as Measures, Calculated Measures) that needs to be displayed as aggregated numeric values in the pivot table.

[PivotViewFilters](#) – – Collection of [OLAP cube elements](#) (such as Hierarchies and Calculated Members) that would act as master filter over the data bound in row, column and value axes of the pivot table.

In-order to define each [OLAP cube element](#) in the respective axis, the following basic properties should be set.

- [Name](#): It allows to set the unique name of the hierarchies, named set, measures, calculated members etc., from the bound OLAP data source. It's casing should match exactly like in the data source and if not set properly, the pivot table will be rendered as empty.
- [Caption](#): It allows to set the caption, which is the alias name of the unique name that needs to be displayed in the pivot table. If not provided, unique name will be displayed.

In this sample, "Product Categories" is added in column, "Customer Geography" in row, and "Customer Count" and "Internet Sales Amount" in value axes respectively.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails" Width="800" Height="350" >
<PivotViewDataSourceSettings TValue="ProductDetails"
ProviderType="ProviderType.SSAS" Catalog="Adventure Works DW 2008 SE"
Cube="Adventure Works" Url="https://bi.syncfusion.com/olap/msmdpump.dll"
LocaleIdentifier="1033" EnableSorting="true">
<PivotViewColumns>
<PivotViewColumn Name="[Product].[Product Categories]" Caption="Product
Category"></PivotViewColumn>
<PivotViewColumn Name="[Measures]" Caption="Measure"></PivotViewColumn>
```

```

</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="[Customer].[Customer Geography]" Caption="Customer
Geography"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="[Measures].[Customer Count]" Caption="Customer
Count"></PivotViewValue>
<PivotViewValue Name="[Measures].[Internet Sales Amount]" Caption="Internet
Sales Amount"></PivotViewValue>
</PivotViewValues>
</PivotViewDataSourceSettings>
<PivotViewGridSettings ColumnWidth="160"></PivotViewGridSettings>
</SfPivotView>
<style>
.e-pivotview {
min-height: 200px;
}
</style>
@code{
public class ProductDetails
{
public int Sold { get; set; }
public double Amount { get; set; }
public string Country { get; set; }
public string Products { get; set; }
public string Year { get; set; }
public string Quarter { get; set; }
}
}

```

Applying formatting to measures

Formatting defines a way in which values should be displayed in pivot table. For example, format “C0” denotes the values should be displayed in currency pattern without decimal points. To do so, define the [PivotViewFormatSetting](#) with its [Name](#) and [Format](#) properties. In this sample, the [Name](#) property is set as “[Measures].[Internet Sales Amount]”, a measure from value axis and its [Format](#) is set as “C0”. Likewise, we can set format for other measures as well.

Only measures from [PivotViewValues](#) axis, which is in the form of numeric data values are applicable for formatting.

ASPX-CS

```

@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails" Width="800" Height="350" >
<PivotViewDataSourceSettings TValue="ProductDetails"
ProviderType="ProviderType.SSAS" Catalog="Adventure Works DW 2008 SE"
Cube="Adventure Works" Url="https://bi.syncfusion.com/olap/msmdpump.dll"
LocaleIdentifier="1033" EnableSorting="true">
<PivotViewColumns>
<PivotViewColumn Name="[Product].[Product Categories]" Caption="Product
Category"></PivotViewColumn>
<PivotViewColumn Name="[Measures]" Caption="Measure"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>

```



```

<PivotViewRow Name="[Customer].[Customer Geography]" Caption="Customer
Geography"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="[Measures].[Customer Count]" Caption="Customer
Count"></PivotViewValue>
<PivotViewValue Name="[Measures].[Internet Sales Amount]" Caption="Internet
Sales Amount"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="[Measures].[Internet Sales Amount]"
Format="C0"></PivotViewFormatSetting>
</PivotViewFormatSettings>
</PivotViewDataSourceSettings>
<PivotViewGridSettings ColumnWidth="160"></PivotViewGridSettings>
</SfPivotView>
<style>
.e-pivotview {
min-height: 200px;
}
</style>
@code{
public class ProductDetails
{
public int Sold { get; set; }
public double Amount { get; set; }
public string Country { get; set; }
public string Products { get; set; }
public string Year { get; set; }
public string Quarter { get; set; }
}
}

```

After successful compilation of the application, simply press F5 to run the same. The pivot table component will render in the default web browser like below.

	Accessories		Bikes	
	Customer Count	Internet Sales Amount	Customer Count	Internet Sales Amount
Australia	2,905	\$138,690.63	2,155	\$8,852,0
Canada	1,230	\$103,377.85	656	\$1,821,3
France	1,505	\$63,406.78	816	\$2,553,5
Germany	1,535	\$62,232.59	904	\$2,808,5
United Kingdom	1,677	\$76,630.04	1,032	\$3,282,8
United States	6,262	\$256,422.07	3,569	\$8,999,8
Grand Total	15,114	\$700,759.96	9,132	\$28,318,1

Enable pivot field list

The component provides a built-in Field List similar to Microsoft Excel. It allows you to add or remove [OLAP cube elements](#) and also rearrange the [OLAP cube elements](#) between different axes, including

[PivotViewRows](#), [PivotViewColumns](#), [PivotViewValues](#) and [PivotViewFilters](#) along with filter and sort options dynamically at runtime. It can be enabled by setting the [ShowFieldList](#) property to **true** as follows. To know more about field list, [refer](#) here.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails" ShowFieldList="true" Width="800"
Height="350" >
  <PivotViewDataSourceSettings TValue="ProductDetails"
ProviderType="ProviderType.SSAS" Catalog="Adventure Works DW 2008 SE"
Cube="Adventure Works" Url="https://bi.syncfusion.com/olap/msmdpump.dll"
LocaleIdentifier="1033" EnableSorting="true">
    <PivotViewColumns>
      <PivotViewColumn Name="[Product].[Product Categories]" Caption="Product
Category"></PivotViewColumn>
      <PivotViewColumn Name="[Measures]" Caption="Measure"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="[Customer].[Customer Geography]" Caption="Customer
Geography"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="[Measures].[Customer Count]" Caption="Customer
Count"></PivotViewValue>
      <PivotViewValue Name="[Measures].[Internet Sales Amount]" Caption="Internet
Sales Amount"></PivotViewValue>
    </PivotViewValues>
    <PivotViewFormatSettings>
      <PivotViewFormatSetting Name="[Measures].[Internet Sales Amount]"
Format="C0"></PivotViewFormatSetting>
    </PivotViewFormatSettings>
  </PivotViewDataSourceSettings>
  <PivotViewGridSettings ColumnWidth="160"></PivotViewGridSettings>
</SfPivotView>
<style>
.e-pivotview {
min-height: 200px;
}
</style>
@code{
public class ProductDetails
{
    public int Sold { get; set; }
    public double Amount { get; set; }
    public string Country { get; set; }
    public string Products { get; set; }
    public string Year { get; set; }
    public string Quarter { get; set; }
}
```

	Accessories		Bikes	
	Customer Count	Internet Sales Amount	Customer Count	Internet Sales Amount
Australia	2,905	\$138,691	2,155	\$8,852
Canada	1,230	\$103,378	656	\$1,821
France	1,505	\$63,407	816	\$2,550
Germany	1,535	\$62,233	904	\$2,808
United Kingdom	1,677	\$76,630	1,032	\$3,282
United States	6,262	\$256,422	3,569	\$8,990
Grand Total	15,114	\$700,760	9,132	\$28,318

Field List

All Fields

Σ Measures

Account

Customer

Date

Delivery Date

Department

Destination Currency

Filters

Drop filter here

Columns

Product Category

Measures

Rows

Customer Geography

Values

Customer Count

Internet Sales Amount

Close

Enable grouping bar

The Grouping Bar feature automatically populates [OLAP cube elements](#) from the bound data source and allows end users to drag [OLAP cube elements](#) between different axes such as [PivotViewRows](#), [PivotViewColumns](#), [PivotViewValues](#) and [PivotViewFilters](#) along with filter and sort options dynamically at runtime. It can be enabled by setting the [ShowGroupingBar](#) property to **true** as follows. To know more about grouping bar, [refer](#) here.

ASPX-CS

```

@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails" ShowGroupingBar="true" Width="800"
Height="350" >
<PivotViewDataSourceSettings TValue="ProductDetails"
ProviderType="ProviderType.SSAS" Catalog="Adventure Works DW 2008 SE"
Cube="Adventure Works" Url="https://bi.syncfusion.com/olap/msmdpump.dll"
LocaleIdentifier="1033" EnableSorting="true">
<PivotViewColumns>
<PivotViewColumn Name="[Product].[Product Categories]" Caption="Product
Category"></PivotViewColumn>
<PivotViewColumn Name="[Measures]" Caption="Measure"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="[Customer].[Customer Geography]" Caption="Customer
Geography"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="[Measures].[Customer Count]" Caption="Customer
Count"></PivotViewValue>
<PivotViewValue Name="[Measures].[Internet Sales Amount]" Caption="Internet
Sales Amount"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="[Measures].[Internet Sales Amount]"
Format="C0"></PivotViewFormatSetting>
</PivotViewFormatSettings>
</PivotViewDataSourceSettings>
<PivotViewGridSettings ColumnWidth="160"></PivotViewGridSettings>
</SfPivotView>
<style>
.e-pivotview {
min-height: 200px;
}
</style>
@code{
public class ProductDetails
{
public int Sold { get; set; }
public double Amount { get; set; }
public string Country { get; set; }
public string Products { get; set; }
public string Year { get; set; }
public string Quarter { get; set; }
}
}

```

Customer Count ✕	Drop filter here			
Internet Sales Amount ✕	Product Categories ↑ ↓ ✕		Measures ✕	
Customer Geo... ↑ ↓ ✕	Accessories		Bikes	
	Customer Count	Internet Sales Amount	Customer Count	Internet Sa
▶ Australia	2,905	\$138,691	2,155	\$8,8
▶ Canada	1,230	\$103,378	656	\$1,8
▶ France	1,505	\$63,407	816	\$2,5
▶ Germany	1,535	\$62,233	904	\$2,8
▶ United Kingdom	1,677	\$76,630	1,032	\$3,2

Exploring filter axis

The filter axis contains collection of [OLAP cube elements](#) such as hierarchies and calculated members that would act as master filter over the data bound in [PivotViewRows](#), [PivotViewColumns](#), and [PivotViewValues](#) axes of the pivot table. The [OLAP cube elements](#) along with filter members could be set to filter axis either through report via code behind or by dragging and dropping [OLAP cube elements](#) from other axes to filter axis via grouping bar or field list at runtime.

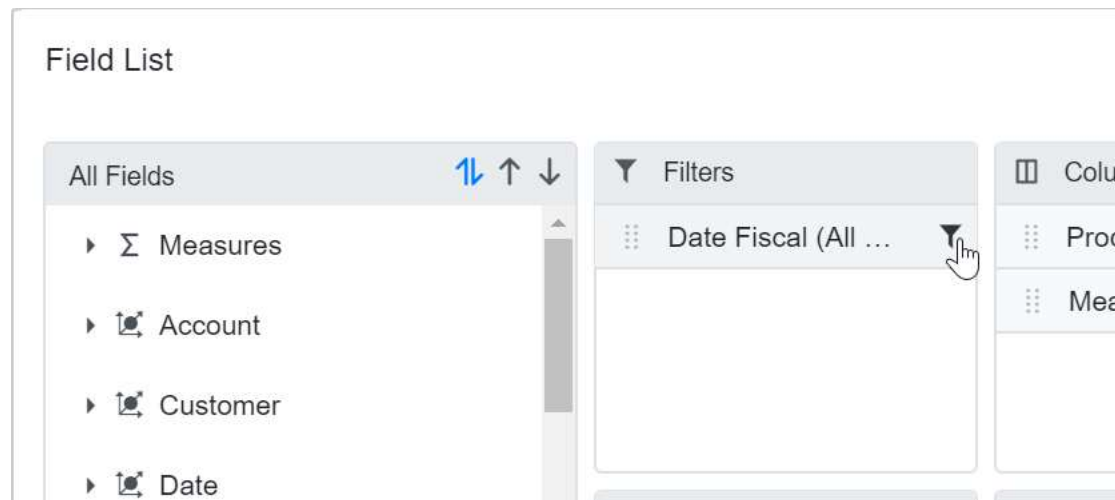
ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails" ShowGroupingBar="true"
ShowFieldList="true">
  <PivotViewDataSourceSettings TValue="ProductDetails" Catalog="Adventure
Works DW 2008 SE" Cube="Adventure Works"
url="https://bi.syncfusion.com/olap/msmdpump.dll" ProviderType="SSAS"
EnableSorting=true>
    <PivotViewColumns>
      <PivotViewColumn Name="[Product].[Product Categories]" Caption="Product
Categories"></PivotViewColumn>
      <PivotViewColumn Name="[Measures]" Caption="Measures"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="[Customer].[Customer Geography]" Caption="Customer
Geography"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="[Measures].[Customer Count]" Caption="Customer
Count"></PivotViewValue>
      <PivotViewValue Name="[Measures].[Internet Sales Amount]" Caption="Internet
Sales Amount"></PivotViewValue>
    </PivotViewValues>
    <PivotViewFilters>
      <PivotViewFilter Name="[Date].[Fiscal]" Caption="Date
Fiscal"></PivotViewFilter>
    </PivotViewFilters>
  </PivotViewDataSourceSettings>
</SfPivotView>
@code{
public class ProductDetails
{
```

```

public int Sold { get; set; }
public double Amount { get; set; }
public string Country { get; set; }
public string Products { get; set; }
public string Year { get; set; }
public string Quarter { get; set; }
}
}

```



Customer Count x	Date Fiscal (All Periods) x	
Internet Sales Amount x	Product Categories ↑ x	Measures x
Customer Geo... ↑ x	Accessories	
	Customer Count	Internet Sales Amount
▶ Australia	2,905	\$138,691

Calculated Field

The calculated field allows user to insert or add a new calculated field based on the available [OLAP cube elements](#) from the bound data source. Calculated fields are nothing but customized dimensions or measures that are newly created based on the user-defined expression.

The two types of calculated fields are as follows:

- **Calculated Measure** – Creates a new measure through user-defined expression.
- **Calculated Dimension** – Creates a new dimension through user-defined expression.

It can be customized using the [PivotViewCalculatedFieldSetting](#) property through code behind. The setting required for calculate field feature at code behind are:

- [Name](#): It allows to set the unique name for new calculated field.
- [Formula](#): It allows to set the user-defined expression.
- [HierarchyUniqueName](#) : It allows to specify dimension unique name whose hierarchies alone should be used in the expression. This will be applicable only for calculated dimension.
- [FormatString](#) : It allows to set the format string for the resultant calculated field.

You need to set [IsCalculatedField](#) property to true, while adding calculated fields to respective axis through code behind.

Also calculated fields can be added at run time through the built-in dialog. The dialog can be enabled by setting the [AllowCalculatedField](#) property to **true** as follows. You will see a button enabled in the Field List UI automatically to invoke the calculated field dialog and perform necessary operation.

Calculated measure can be added only in value axis.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails" ShowFieldList="true"
AllowCalculatedField="true" Width="800" Height="350" >
@*<PivotViewDisplayOption Primary=Primary.Table
View=View.Both></PivotViewDisplayOption>*@
<PivotViewDataSourceSettings TValue="ProductDetails"
ProviderType="ProviderType.SSAS" Catalog="Adventure Works DW 2008 SE"
Cube="Adventure Works" Url="https://bi.syncfusion.com/olap/msmdpump.dll"
LocaleIdentifier="1033" EnableSorting="true">
<PivotViewColumns>
<PivotViewColumn Name="[Product].[Product Categories]" Caption="Product
Categories"></PivotViewColumn>
<PivotViewColumn Name="[Measures]" Caption="Measures"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="[Customer].[Customer Geography]" Caption="Customer
Geography"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="[Measures].[Customer Count]" Caption="Customer
Count"></PivotViewValue>
<PivotViewValue Name="[Measures].[Internet Sales Amount]" Caption="Internet
Sales Amount"></PivotViewValue>
<PivotViewValue Name="Order on Discount"
IsCalculatedField="true"></PivotViewValue>
</PivotViewValues>
<PivotViewFilters>
<PivotViewFilter Name="[Date].[Fiscal]" Caption="Date
Fiscal"></PivotViewFilter>
</PivotViewFilters>
<PivotViewCalculatedFieldSettings>
<PivotViewCalculatedFieldSetting Name="BikeAndComponents"
Formula="([Product].[Product Categories].[Category].[Bikes] +
[Product].[Product Categories].[Category].[Components])"
HierarchyUniqueName="[Product].[Product Categories]"
FormatString="Standard"></PivotViewCalculatedFieldSetting>
<PivotViewCalculatedFieldSetting Name="Order on Discount"
Formula="[Measures].[Order Quantity] + ([Measures].[Order Quantity] * 0.10)"
FormatString="Currency"></PivotViewCalculatedFieldSetting>
```

```

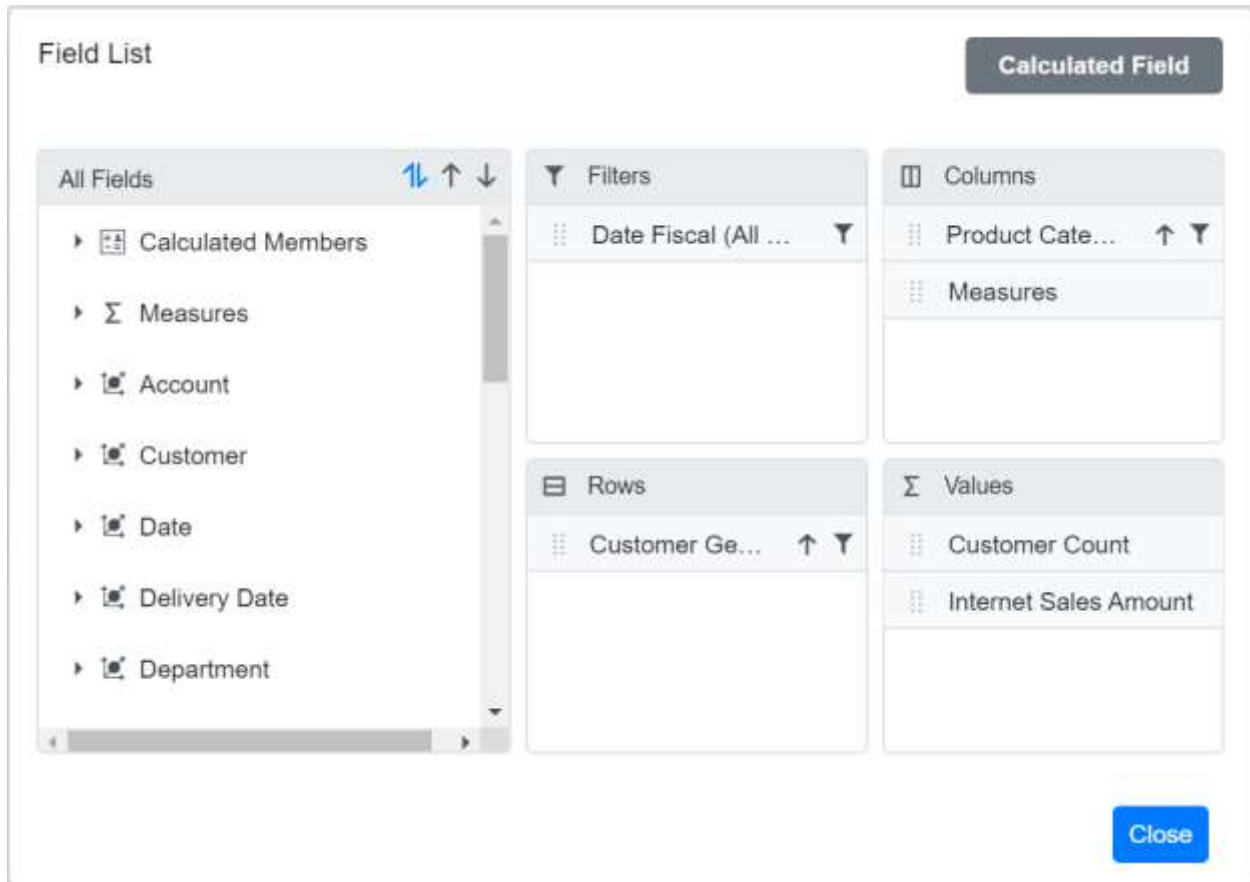
</PivotViewCalculatedFieldSettings>
</PivotViewDataSourceSettings>
<PivotViewGridSettings ColumnWidth="160"></PivotViewGridSettings>
</SfPivotView>
@code{
public class ProductDetails
{
public int Sold { get; set; }
public double Amount { get; set; }
public string Country { get; set; }
public string Products { get; set; }
public string Year { get; set; }
public string Quarter { get; set; }
}
}

```

	Accessories		
	Customer Count	Internet Sales Amount	Order on Discount
▶ Australia	2,905	\$138,690.63	\$68,124.10
▶ Canada	1,230	\$103,377.85	\$68,124.10
▶ France	1,505	\$63,406.78	\$68,124.10
▶ Germany	1,535	\$62,232.59	\$68,124.10

Users can add a calculated field at runtime through the built-in dialog by using the following steps.

Step 1: Click the "Calculated Field" button in the field list dialog positioned at the top right corner. The calculated field dialog will be opened now. Enter the name of the calculated field to be created.



Create Calculated Field

All Fields

Calculated Members

Measures

Account

Customer

Date

Delivery Date

Department

Destination Currency

Employee

Geography

Internet Sales Order Details

Organization

Product

Promotion

Field Name

BikeAndComponents

Expression

Example: [Measures].[Order Quantity] + ([Measures].[Order Quantity] * 0.10)

Field Type

Measure

Parent Hierarchy

Account

Format

Standard

Enter custom format string

Clear

OK

Cancel

Step 2: Frame the expression by dragging and dropping the fields from the tree view on the left side of the dialog using simple arithmetic operators. **Example:** "IIF([Measures].[Internet Sales Amount]^0.5 > 100, [Measures].[Internet Sales Amount]*100, [Measures].[Internet Sales Amount]/100)". Please refer here to learn more about the supported [operators](#) and [functions](#) to frame the expression.

Create Calculated Field

All Fields

Financial

History

Large Photo

Model Name

Product

Product Categories

Product Key

Product Line

Product Model Categories

Product Model Lines

Style

Subcategory

Promotion

Reseller

Field Name

BikeAndComponents

Expression

Example: [Measures].[Order Quantity] + ([Measures].[Order Quantity] * 0.10)

Product Categories

Field Type

Measure

Parent Hierarchy

Account

Format

Standard

Enter custom format string

Clear

OK

Cancel

Step 3: Confirm the type of the field to be created - calculated measure or calculated dimension.

Create Calculated Field ✕

All Fields i

- Calculated Members
 - BikeAndComponents
 - Order on Discount
- Measures
- Account
- Customer
- Date
- Delivery Date
- Department
- Destination Currency
- Employee
- Geography
- Internet Sales Order Details
- Organization

Field Name

BikeAndComponents

Expression

```
([Product].[Product Categories].[Category].[Bikes] + [Product].[Product Categories].[Category].[Components] )
```

Field Type

Dimension

Measure

Dimension

Format

Standard

Enter custom format string

Clear

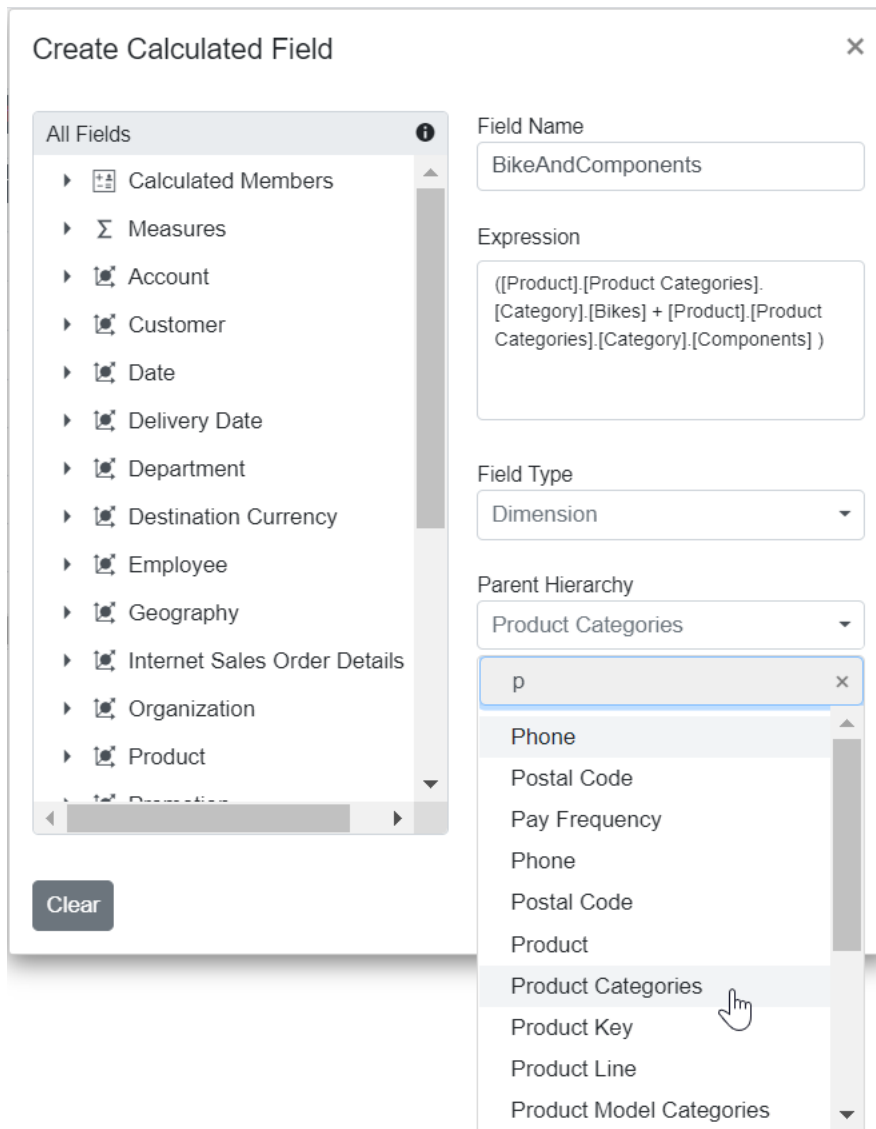
OK

Cancel

Step 4: Choose the parent hierarchy of the calculated field. NOTE: It is only applicable to the calculated dimension.

Copyright © 2001 - 2022 Syncfusion Inc.

335



Step 5: Then select the format string from the drop-down list and finally click "OK".

Create Calculated Field

All Fields

- Calculated Members
- Measures
- Account
- Customer
- Date
- Delivery Date
- Department
- Destination Currency
- Employee
- Geography
- Internet Sales Order Details
- Organization
- Product
- Promotion

Field Name

BikeAndComponents

Expression

`([Product].[Product Categories].[Category].[Bikes] + [Product].[Product Categories].[Category].[Components])`

Field Type

Dimension

Parent Hierarchy

Product Categories

Format

Standard

- Standard
- Currency
- Percent
- Custom

Clear

Format String

Allows you to specify the required format string while creating new calculated field. Supported format strings are:

- **Standard** - Denotes the numeric type.
- **Currency** - Denotes the currency type.
- **Percent** - Denotes the percentage type.
- **Custom** - Denotes the custom format. For example: "###0.##0#". This shows the value "9584.3" as "9584.300."

By default, **Standard** will be selected from the drop down list.

Create Calculated Field

All Fields

Calculated Members

Measures

Account

Customer

Date

Delivery Date

Department

Destination Currency

Employee

Geography

Internet Sales Order Details

Organization

Product

Promotion

Field Name

Enter the field name

Expression

Example: [Measures].[Order Quantity] + ([Measures].[Order Quantity] * 0.10)

Field Type

Measure

Parent Hierarchy

Account

Format

Standard

Standard

Currency

Percent

Custom

Clear


Renaming the existing calculated field

Existing calculated field can be renamed only through the UI at runtime. To do so, open the calculated field dialog, click the target field. User can now see the existing name getting displayed in the text box at the top of the dialog. Now, change the name based on user requirement and click "OK".

<!-- markdownlint-disable MD012 -->


Create Calculated Field ✕


All Fields i


▼  Calculated Members


Σ Order of Discount


▶ Σ Measures


▶  Account


▶  Customer


▶  Date


▶  Delivery Date


▶  Department


▶  Destination Currency

▶  Employee

▶  Geography

▶  Internet Sales Order Details

▶  Organization

▶  Product

Field Name

Enter the field name

Expression

Example: [Measures].[Order Quantity] + ([Measures].[Order Quantity] * 0.10)

Field Type

Measure ▼

Parent Hierarchy

Account ▼

Format

Standard ▼

Enter custom format string

Clear

OK

Cancel

Create Calculated Field

All Fields

Calculated Members

Order on Discount

Measures

Account

Customer

Date

Delivery Date

Department

Destination Currency

Employee

Geography

Internet Sales Order Details

Organization

Product

Field Caption

Order Quantity on Discount

Expression

$$[\text{Measures}].[Order\ Quantity] + ([\text{Measures}].[Order\ Quantity] * 0.10)$$

Field Type

Measure

Parent Hierarchy

Account

Format

Standard

Enter custom format string

Clear

OK


Cancel

[Editing the existing calculated field formula](#)

Existing calculated field formula can be edited only through the UI at runtime. To do so, open the calculated field dialog, click the target field. User can now see the existing expression getting displayed in a "Expression" section. Now, change the expression based on user requirement and click "OK".


Create Calculated Field ✕


All Fields i


▼  Calculated Members


Σ Order of Discount


▶ Σ Measures


▶  Account


▶  Customer


▶  Date


▶  Delivery Date


▶  Department


▶  Destination Currency

▶  Employee

▶  Geography

▶  Internet Sales Order Details

▶  Organization

▶  Product

Field Name

Enter the field name

Expression

Example: [Measures].[Order Quantity] + ([Measures].[Order Quantity] * 0.10)

Field Type

Measure ▼

Parent Hierarchy

Account ▼

Format

Standard ▼

Enter custom format string

Clear

OK

Cancel

Create Calculated Field ✕

All Fields i

Calculated Members

Order on Discount

Measures

Account

Customer

Date

Delivery Date

Department

Destination Currency

Employee

Geography

Internet Sales Order Details

Organization

Product

Field Caption

Order Quantity on Discount

Expression

$$[\text{Measures}].[Order\ Quantity] + ([\text{Measures}].[Order\ Quantity] * 0.50)$$

Field Type

Measure

Parent Hierarchy

Account

Format

Standard

Enter custom format string

Clear

OK


Cancel

[Reusing the existing formula in a new calculate field](#)

While creating a new calculated field, if user wants to add the formula of an existing calculated field, it can be done easily. To do so, simply drag-and-drop the existing calculated field to the "Expression" section.


Create Calculated Field ✕


All Fields i


▼  Calculated Members


Σ Order of Discount


▶ Σ Measures


▶  Account


▶  Customer


▶  Date


▶  Delivery Date


▶  Department


▶  Destination Currency

▶  Employee

▶  Geography

▶  Internet Sales Order Details

▶  Organization

▶  Product

Field Name

Enter the field name

Expression

Example: [Measures].[Order Quantity] + ([Measures].[Order Quantity] * 0.10)

Field Type

Measure ▼

Parent Hierarchy

Account ▼

Format

Standard ▼

Enter custom format string

Clear

OK

Cancel

Create Calculated Field ✕

All Fields i

Calculated Members

Σ Order on Discount

Σ Measures

Account

Customer

Date

Delivery Date

Department

Destination Currency

Employee

Geography

Internet Sales Order Details

Organization

Product

Field Name

Discount Quantity

Expression

Example: [Measures].[Order Quantity] +
([Measures].[Order Quantity] * 0.10)

Order on Discount

Field Type

Measure

Parent Hierarchy

Account

Format

Standard

Enter custom format string

Clear

OK

Cancel

Create Calculated Field ✕

All Fields i

Calculated Members

Order on Discount

Measures

Account

Customer

Date

Delivery Date

Department

Destination Currency

Employee

Geography

Internet Sales Order Details

Organization

Product

Field Caption

Discount Quantity

Expression

[Measures].[Order on Discount]

Field Type

Measure

Parent Hierarchy

Account

Format

Standard

Enter custom format string

Clear

OK


Cancel

Modifying the existing format string

Existing calculated field's format string can be modified only through the UI at runtime. To do so, open the calculated field dialog and click the target calculated field. User can now see the format string for the existing calculated field getting displayed in a drop-down list. Change the format string based on the requirement and finally click "OK".


Create Calculated Field ✕


All Fields i


▼  Calculated Members


Σ Order of Discount


▶ Σ Measures


▶  Account


▶  Customer


▶  Date


▶  Delivery Date


▶  Department


▶  Destination Currency

▶  Employee

▶  Geography

▶  Internet Sales Order Details

▶  Organization

▶  Product

Field Name

Enter the field name

Expression

Example: [Measures].[Order Quantity] + ([Measures].[Order Quantity] * 0.10)

Field Type

Measure ▼

Parent Hierarchy

Account ▼

Format

Standard ▼

Enter custom format string

Clear

OK

Cancel

Create Calculated Field

All Fields

Calculated Members

Order on Discount

Measures

Account

Customer

Date

Delivery Date

Department

Destination Currency

Employee

Geography

Internet Sales Order Details

Organization

Product

Field Caption

Order on Discount

Expression

$$[\text{Measures}].[Order\ Quantity] + ([\text{Measures}].[Order\ Quantity] * 0.10)$$

Field Type

Measure

Parent Hierarchy

Account

Format

Currency

Standard

Currency

Percent

Custom

Clear

[Clearing the changes while editing the calculated field](#)

Previous changes can be cleared by using the "Clear" option while performing operations such as creating and editing the calculated field. To do so, click the "Clear" button in the bottom left corner of the dialog.

Create Calculated Field

All Fields

Calculated Members

Order on Discount

BikeAndComponents

Measures

Account

Customer

Date

Delivery Date

Department

Destination Currency

Employee

Geography

Internet Sales Order Details

Organization

Field Caption

Order on Discount

Expression

[Measures].[Order Quantity] +
([Measures].[Order Quantity] * 0.10)

Field Type

Measure

Parent Hierarchy

Account

Format

Currency

Enter custom format string

Clear

OK

Cancel

Virtual Scrolling

Allows large amounts of data to be loaded without any performance degradation by rendering rows and columns in relation to the current viewport. Rest of the data will be brought into the viewport dynamically based on vertical or horizontal scroll position. This feature can be enabled by setting the [EnableVirtualization](#) property in [SfPivotView](#) class to **true**.

To use the virtual scrolling feature, inject the `VirtualScroll` module into the pivot table.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails" Width="800" Height="350"
EnableVirtualization="true">
<PivotViewDataSourceSettings TValue="ProductDetails"
ProviderType="ProviderType.SSAS" Catalog="Adventure Works DW 2008 SE"
```



```

Cube="Adventure Works" Url="https://bi.syncfusion.com/olap/msmdpump.dll"
LocaleIdentifier="1033" EnableSorting="true">
<PivotViewColumns>
<PivotViewColumn Name="[Product].[Product Categories]" Caption="Product
Category"></PivotViewColumn>
<PivotViewColumn Name="[Measures]" Caption="Measure"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="[Customer].[Customer]"
Caption="Customer"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="[Measures].[Customer Count]" Caption="Customer
Count"></PivotViewValue>
<PivotViewValue Name="[Measures].[Internet Sales Amount]" Caption="Internet
Sales Amount"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="[Measures].[Internet Sales Amount]"
Format="C0"></PivotViewFormatSetting>
</PivotViewFormatSettings>
</PivotViewDataSourceSettings>
<PivotViewGridSettings ColumnWidth="160"></PivotViewGridSettings>
</SfPivotView>
<style>
.e-pivotview {
min-height: 200px;
}
</style>
@code{
public class ProductDetails
{
public int Sold { get; set; }
public double Amount { get; set; }
public string Country { get; set; }
public string Products { get; set; }
public string Year { get; set; }
public string Quarter { get; set; }
}
}

```

The screenshot shows a Blazor Pivot Table component. On the left, there are filter buttons for 'Customer Count', 'Internet Sales Amount', and 'Customer Group'. The main area displays a data grid with columns for 'Hydration Packs', 'Tees and Tanks', and 'Bikes'. Each of these category columns is further divided into 'Internet Sales Amount' and 'Customer Count'. The rows list customer names and their corresponding sales data.

	Hydration Packs		Tees and Tanks		Bikes	
	Internet Sales Amount	Customer Count	Internet Sales Amount	Customer Count	Internet Sales Amount	Customer Count
Aaron A. Allen						1
Aaron A. Hayes						1
Aaron A. Zhang	\$34.99			1	\$25.48	1
Aaron Alexander						
Aaron B. Adams	\$34.99			1	\$28.96	
Aaron Bryant	\$34.99			1	\$35.99	

Limitations for virtual scrolling

- The [ColumnWidth](#) property in [GridSettings](#) should be in pixels. The percentage value is not accepted.
- Resizing columns and setting the width of individual columns will affect scrolling and is therefore not recommended.
- The grand totals option is not supported by virtual scrolling.

Data Binding

To bind OLAP datasource to the pivot table, you need to specify following properties under [PivotViewDataSourceSettings](#) option.

Properties	Description
Cube	Points the respective cube name from OLAP database.
ProviderType	Points the provider type for pivot table to identify the type of data source.
Url	Contains the cube URL for establishing the connection (online).
Catalog	Contains the database name (catalog name) to fetch the data.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails" Width="800" Height="350" >
  <PivotViewDataSourceSettings TValue="ProductDetails"
    ProviderType="ProviderType.SSAS" Catalog="Adventure Works DW 2008 SE"
    Cube="Adventure Works" Url="https://bi.syncfusion.com/olap/msmdpump.dll"
    LocaleIdentifier="1033" EnableSorting="true">
    <PivotViewColumns>
      <PivotViewColumn Name="[Product].[Product Categories]" Caption="Product
      Category"></PivotViewColumn>
      <PivotViewColumn Name="[Measures]" Caption="Measure"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="[Customer].[Customer Geography]" Caption="Customer
      Geography"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="[Measures].[Customer Count]" Caption="Customer
      Count"></PivotViewValue>
      <PivotViewValue Name="[Measures].[Internet Sales Amount]" Caption="Internet
      Sales Amount"></PivotViewValue>
    </PivotViewValues>
    <PivotViewFormatSettings>
      <PivotViewFormatSetting Name="[Measures].[Internet Sales Amount]"
      Format="C0"></PivotViewFormatSetting>
    </PivotViewFormatSettings>
  </PivotViewDataSourceSettings>
  <PivotViewGridSettings ColumnWidth="160"></PivotViewGridSettings>
</SfPivotView>
<style>
.e-pivotview {
min-height: 200px;
```

```

}
</style>
@code{
public class ProductDetails
{
public int Sold { get; set; }
public double Amount { get; set; }
public string Country { get; set; }
public string Products { get; set; }
public string Year { get; set; }
public string Quarter { get; set; }
}
}

```

	Accessories		Bikes	
	Customer Count	Internet Sales Amount	Customer Count	Internet Sales Amount
Australia	2,905	\$138,690.63	2,155	\$8,852,0
Canada	1,230	\$103,377.85	656	\$1,821,3
France	1,505	\$63,406.78	816	\$2,553,5
Germany	1,535	\$62,232.59	904	\$2,808,5
United Kingdom	1,677	\$76,630.04	1,032	\$3,282,8
United States	6,262	\$256,422.07	3,569	\$8,999,8
Grand Total	15,114	\$700,759.96	9,132	\$28,318,1

Fields

Measures in row axis

By default, the measures are plotted in column axis. You can place measures in row axis either through code behind or UI. To plot those measures in row axis, place the **Measures** field in the row axis as follows.

ASPX-CS

```

@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails" Width="800" Height="350" >
<PivotViewDataSourceSettings TValue="ProductDetails"
ProviderType="ProviderType.SSAS" Catalog="Adventure Works DW 2008 SE"
Cube="Adventure Works" Url="https://bi.syncfusion.com/olap/msmdpump.dll"
LocaleIdentifier="1033" EnableSorting="true">
<PivotViewColumns>
<PivotViewColumn Name="[Product].[Product Categories]" Caption="Product
Category"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="[Customer].[Customer Geography]" Caption="Customer
Geography"></PivotViewRow>
<PivotViewRow Name="[Measures]" Caption="Measures"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>

```

```

<PivotViewValue Name="[Measures].[Customer Count]" Caption="Customer
Count"></PivotViewValue>
<PivotViewValue Name="[Measures].[Internet Sales Amount]" Caption="Internet
Sales Amount"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="[Measures].[Internet Sales Amount]"
Format="C0"></PivotViewFormatSetting>
</PivotViewFormatSettings>
</PivotViewDataSourceSettings>
<PivotViewGridSettings ColumnWidth="160"></PivotViewGridSettings>
</SfPivotView>
<style>
.e-pivotview {
min-height: 200px;
}
</style>
@code{
public class ProductDetails
{
public int Sold { get; set; }
public double Amount { get; set; }
public string Country { get; set; }
public string Products { get; set; }
public string Year { get; set; }
public string Quarter { get; set; }
}
}

```

	Accessories	Bikes	Clothing	Grand Total
Australia				
Customer Count	2,905	2,155	1,436	
Internet Sales Amount	\$138,691	\$8,852,050	\$70,260	
Canada				
Customer Count	1,230	656	715	
Internet Sales Amount	\$103,378	\$1,821,302	\$53,165	
France				
Customer Count	1,505	816	624	

Named set

Named set is a multidimensional expression (MDX) that returns a set of dimension members, which can be created by combining the cube data, arithmetic operators, numbers, and functions.

You can bind the named sets in the pivot table by setting its unique name in the [Name](#) property either in row or column axis and [IsNamedSet](#) boolean property to **true**. In this sample, we have added "Core Product Group" named set in the column axis.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
```

```

<SfPivotView TValue="ProductDetails" Width="800" Height="350" >
  <PivotViewDataSourceSettings TValue="ProductDetails"
    ProviderType="ProviderType.SSAS" Catalog="Adventure Works DW 2008 SE"
    Cube="Adventure Works" Url="https://bi.syncfusion.com/olap/msmdpump.dll"
    LocaleIdentifier="1033" EnableSorting="true">
    <PivotViewColumns>
      <PivotViewColumn Name="[Core Product Group]" Caption="Core Product Group"
        IsNamedSet="true"></PivotViewColumn>
      <PivotViewColumn Name="[Measures]" Caption="Measures"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="[Customer].[Customer Geography]" Caption="Customer
        Geography"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="[Measures].[Customer Count]" Caption="Customer
        Count"></PivotViewValue>
      <PivotViewValue Name="[Measures].[Internet Sales Amount]" Caption="Internet
        Sales Amount"></PivotViewValue>
    </PivotViewValues>
  </PivotViewDataSourceSettings>
  <PivotViewGridSettings ColumnWidth="160"></PivotViewGridSettings>
</SfPivotView>
<style>
.e-pivotview {
min-height: 200px;
}
</style>
@code{
public class ProductDetails
{
public int Sold { get; set; }
public double Amount { get; set; }
public string Country { get; set; }
public string Products { get; set; }
public string Year { get; set; }
public string Quarter { get; set; }
}
}

```

	Accessories		Bikes	
			Mountain Bikes	
	Customer Count	Internet Sales Amount	Customer Count	Internet Sales Amount
▶ Australia	2,905	\$138,691	741	
▶ Canada	1,230	\$103,378	292	
▶ France	1,505	\$63,407	419	
▶ Germany	1,535	\$62,233	419	

Configuring authentication

Users can configure basic authentication information to access the OLAP cube using the [PivotViewAuthentication](#) property. The settings required to configure are as follows:

- [UserName](#): It allows the user to set a username that recognizes the basic authentication of the IIS.
- [Password](#): It allows to set the appropriate password.

If the user does not configure the authentication, a default popup will appear in the browser to get the authentication information.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails" Width="800" Height="350" >
<PivotViewDataSourceSettings TValue="ProductDetails"
ProviderType="ProviderType.SSAS" Catalog="Adventure Works DW 2008 SE"
Cube="Adventure Works" Url="https://bi.syncfusion.com/olap/msmdpump.dll"
LocaleIdentifier="1033" EnableSorting="true">
<PivotViewColumns>
<PivotViewColumn Name="[Product].[Product Categories]" Caption="Product
Category"></PivotViewColumn>
<PivotViewColumn Name="[Measures]" Caption="Measure"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="[Customer].[Customer Geography]" Caption="Customer
Geography"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="[Measures].[Customer Count]" Caption="Customer
Count"></PivotViewValue>
<PivotViewValue Name="[Measures].[Internet Sales Amount]" Caption="Internet
Sales Amount"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="[Measures].[Internet Sales Amount]"
Format="C0"></PivotViewFormatSetting>
</PivotViewFormatSettings>
<PivotViewAuthentication UserName="UserName"
Password="Password"></PivotViewAuthentication>
</PivotViewDataSourceSettings>
<PivotViewGridSettings ColumnWidth="160"></PivotViewGridSettings>
</SfPivotView>
<style>
.e-pivotview {
min-height: 200px;
}
</style>
@code{
public class ProductDetails
{
public int Sold { get; set; }
public double Amount { get; set; }
public string Country { get; set; }
public string Products { get; set; }
public string Year { get; set; }
public string Quarter { get; set; }
}
}
```

OLAP Cube: Elements

Field list

The field list, aka cube dimension browser, is a tree view like structure that organizes the cube elements such as dimensions, hierarchies, measures, etc., from the selected cube into independent logical groups.

Types of node in field list

- **Display folder:** A folder that contains a set of similar elements.
- **Measure:** Quantity available for analysis.
- **Dimension:** A name given to the parts of the cube that categorizes data.
- **Attribute Hierarchy:** Level of attributes down the hierarchy.
- **User-defined Hierarchy:** Members of a dimension in a hierarchical structure.
- **Level:** Denotes a specific level in the category.
- **Named Set:** A collection of tuples and members, that can be defined and saved as a part of cube definition for later use.

Measure

In a cube, a measure is a set of values that are based on a column in the cube's fact table and are usually numeric. The measures are the central values of a cube that are analyzed. That is, measures are the numeric data of primary interest to users browsing a cube. You can select measures depend on the types of users request. Some common measures are sales, costs, expenditures, and production count.

Dimension

A simple dimension object is composed of basic information such as name, hierarchy, level, and members. You can create a dimension element by specifying its name and providing the hierarchy and level name. The dimension element contains the hierarchical details and information about each included level elements in that hierarchy. A hierarchy can have any number of level elements and the level elements can have any number of members and the member elements can have any number of child members.

Hierarchy

Each element of a dimension can be summarized using a hierarchy. The hierarchy is a series of parent-child relationship, where a parent member represents the consolidation of members which are its children. Parent members can be further aggregated as the children of another parent. For example, May 2005 can be summarized into Second Quarter 2005 which in turn would be summarized in the year 2005.

Level

Level element is the child of hierarchy element which contains a set of members, each of which has the same rank within a hierarchy.

Attribute hierarchy

Attribute hierarchy contains the following levels:

- A leaf level contains distinct attribute member, and each member of the leaf level is known as a leaf member.
- Intermediate levels if the attribute hierarchy is a parent-child hierarchy.
- An optional (all) level contains the aggregated value of the attribute hierarchy's leaf members, with the member of the (all) level also known as the (all) member.

User-defined hierarchy

User-defined hierarchy organizes the members of a dimension into hierarchical structure and provides navigation paths in a cube. For example, take a dimension table that supports three attributes such as year, quarter, and month. The year, quarter, and month attributes are used to construct a user-defined hierarchy, named Calendar, in the time dimension that relates to all levels.

Differentiating user-defined hierarchy and attribute hierarchy

- User-defined hierarchy contains more than one level whereas attribute hierarchy contains only one level.
- User-defined hierarchy provides the navigation path between the levels taken from attribute hierarchies of the same dimension.
- The attribute hierarchy and the user-defined hierarchy are represented in different ways as shown in the following table.

Named set

A named set is a collection of tuples and members, which can be defined and saved as a part of the cube definition. Named set records reside inside the sets folder, which is under a dimension element. These elements can be dragged to [PivotViewRows](#) or [PivotViewColumns](#) axis via grouping bar or field list at runtime. To work with a lengthy, complex, or commonly used expression easier, Multidimensional Expressions (MDX) allows you to define a named set.









Calculated field

The calculated field allows user to insert or add a new calculated field based on the available OLAP cube elements from the bound data source. Calculated fields are nothing but customized dimensions or measures that are newly created based on the user-defined expression.

The two types of calculated fields are as follows:

- **Calculated Measure** – Creates a new measure through user-defined expression.
- **Calculated Dimension** – Creates a new dimension through user-defined expression.

Symbolic representation of the nodes inside field list

Icon	Name	Node type	Is Draggable
-----	-----	-----	-----
	Display Folder	Display Folder	False
	Measure	Measure	False
	Dimension	Dimension	False
	User Defined Hierarchy	Hierarchy	True
	Attribute Hierarchy	Hierarchy	True
  	Levels (in order)	Level Element	True

|  | Named Set | Named Set | True |

You can also explore our [Blazor Pivot Table example](#) to know how to render and configure the pivot table.

Pivot Chart in Blazor Pivot Table Component

In [Blazor Pivot Table](#) component, pivot chart would act as an additional visualization component with its basic and important characteristics like drill down and drill up, 15+ chart types, series customization, axis customization, legend customization, export, print and tooltip. Its main purpose is to show the pivot data in graphical format.

If user prefers, the pivot chart component can also be displayed individually with pivot values and can change the report dynamically with the help of field list and grouping bar. Using the [PivotViewDisplayOption](#) property in [SfPivotView](#) class, user can set the visibility of grid and chart in pivot table component. It holds below properties,

- [View](#): Specifies the pivot table component to display grid alone or chart alone or both.
- [Primary](#): Specifies the pivot table to display either grid or chart as primary component during initial loading. It is applicable only when setting the property [View](#) to [View.Both](#).

The below sample displays the pivot chart component based on the pivot report bound on it.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails" Width="800" Height="500">
  <PivotViewDataSourceSettings DataSource="@data">
    <PivotViewColumns>
      <PivotViewColumn Name="Country"></PivotViewColumn>
      <PivotViewColumn Name="Products"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Year"></PivotViewRow>
      <PivotViewRow Name="Quarter"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
      <PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
    </PivotViewValues>
    <PivotViewFormatSettings>
      <PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
    </PivotViewFormatSettings>
  </PivotViewDataSourceSettings>
  <PivotViewDisplayOption View=View.Chart></PivotViewDisplayOption>
  <PivotChartSettings Title="Sales Analysis">
    <PivotChartSeries Type=ChartSeriesType.Column></PivotChartSeries>
  </PivotChartSettings>
</SfPivotView>
@code{
  public List<ProductDetails> data { get; set; }
  protected override void OnInitialized()
  {
    this.data = ProductDetails.GetProductData().ToList();
  }
}
```

```
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
}
```



Data Binding

End user can bind both local and remote data binding options available in the component to feed the data. The [DataSource](#) property can be assigned either with an instance of [SfDataManager](#) or list of object.

For more information [refer](#) here.

Chart Types

Supports 19 different types of charts as follows,

- Line
- Column
- Area
- Bar
- StepArea
- StackingColumn
- StackingArea
- StackingBar
- StepLine
- Pareto
- Bubble
- Scatter
- Spline
- SplineArea
- StackingColumn100
- StackingBar100
- StackingArea100
- Polar

- Radar

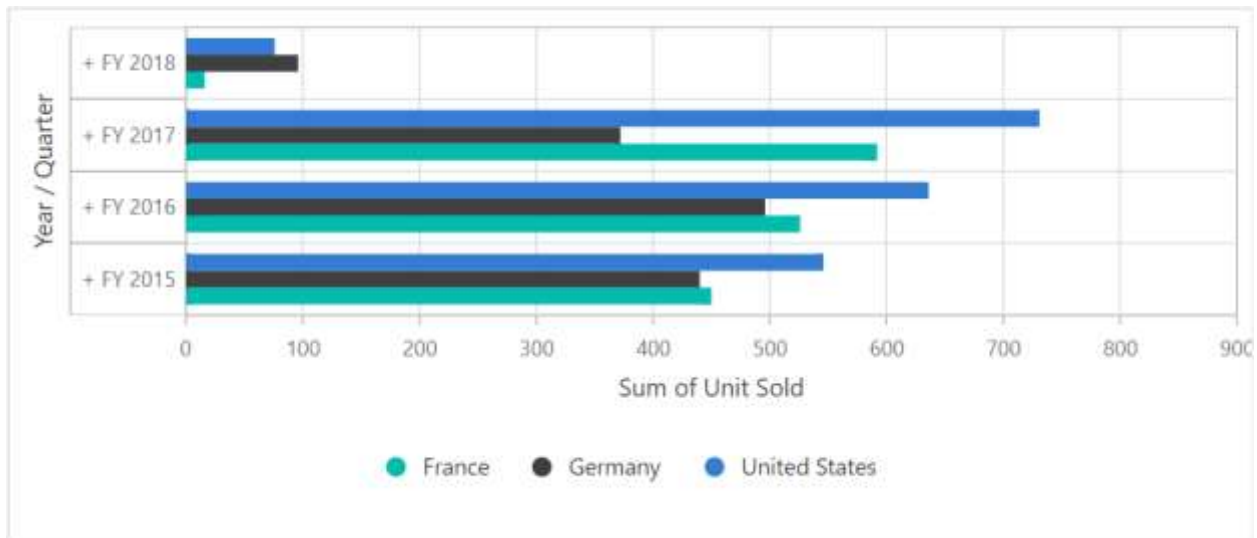
[ChartSeriesType.Line](#) is the default pivot chart type. User can change the pivot chart type by using the property [Type](#) in [PivotChartSeries](#) class.

In the below code sample, the pivot chart type is set as [ChartSeriesType.Bar](#).

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
  <PivotViewDataSourceSettings DataSource="@data">
    <PivotViewColumns>
      <PivotViewColumn Name="Country"></PivotViewColumn>
      <PivotViewColumn Name="Products"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Year"></PivotViewRow>
      <PivotViewRow Name="Quarter"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
    </PivotViewValues>
  </PivotViewDataSourceSettings>
  <PivotViewDisplayOption View=View.Chart></PivotViewDisplayOption>
  <PivotChartSettings>
    <PivotChartSeries Type=ChartSeriesType.Bar></PivotChartSeries>
  </PivotChartSettings>
</SfPivotView>

@code{
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
}
```



Accumulation Charts

Supports 4 different types of accumulation charts as follows,

- Pie
- Doughnut
- Funnel
- Pyramid

As like other chart types it can be changed using the property [Type](#) in [PivotChartSeries](#) class.

In the below code sample, the **Pie** chart is rendered, and the other accumulation charts can be switched using the drop-down list.

ASPX-CS

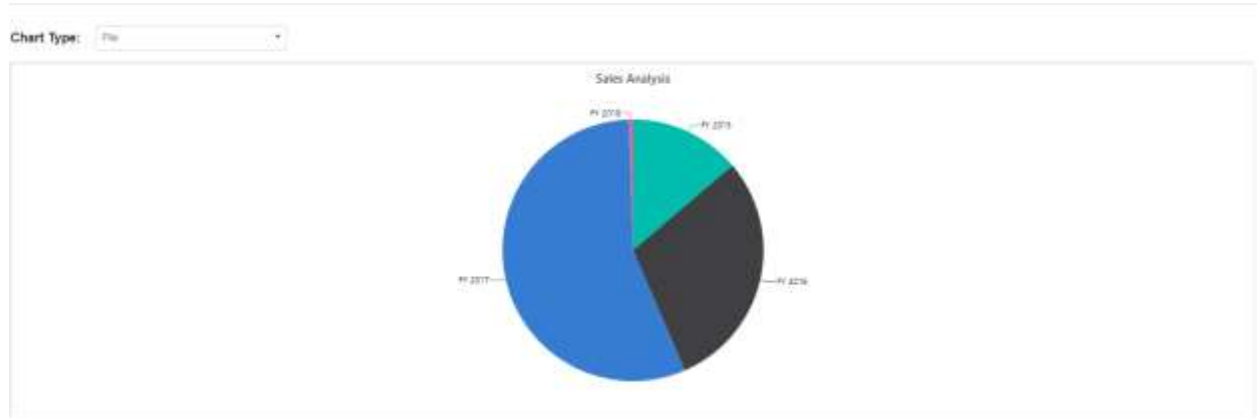
```
@using Syncfusion.Blazor.PivotView
@using Syncfusion.Blazor.DropDowns
<div class="control-section">
<div id="dropdown-control" style="margin-bottom:5px;">
<table style="width: 350px;">
<tbody>
<tr style="height: 50px">
<td>
<div>
<b>Chart Type:</b>
</div>
</td>
<td>
<div>
<SfDropDownList TValue="ChartSeriesType" TItem="DropDownData"
DataSource="@ChartTypes" @bind-Value="@ChartType">
<DropDownListFieldSettings Text="Name"
Value="Value"></DropDownListFieldSettings>
</SfDropDownList>
</div>
</td>
</tr>
</tbody>
</table>
</div>
```

```

</tbody>
</table>
</div>
<div class="content-wrapper">
<SfPivotView TValue="ProductDetails" ShowFieldList=true>
<PivotViewDataSourceSettings DataSource="@Data" ExpandAll=false
EnableSorting=true>
<PivotViewColumns>
<PivotViewColumn Name="Country"></PivotViewColumn>
<PivotViewColumn Name="Products"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Year"></PivotViewRow>
<PivotViewRow Name="Order_Source" Caption="Order Source"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Amount" Caption="Sales Amount"></PivotViewValue>
</PivotViewValues>
<PivotViewDrilledMembers>
<PivotViewDrilledMember Name="Year"
Items="@DrilledMembers"></PivotViewDrilledMember>
</PivotViewDrilledMembers>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="Amount" Format="C"
UseGrouping=true></PivotViewFormatSetting>
</PivotViewFormatSettings>
</PivotViewDataSourceSettings>
<PivotViewDisplayOption View=View.Chart></PivotViewDisplayOption>
<PivotChartSettings Title="Sales Analysis">
<PivotChartSeries Type="@ChartType"></PivotChartSeries>
<PivotChartPrimaryYAxis>
<PivotChartPrimaryYAxisBorder Width="0"></PivotChartPrimaryYAxisBorder>
</PivotChartPrimaryYAxis>
</PivotChartSettings>
</SfPivotView>
</div>
</div>
@code{
public ChartSeriesType ChartType = ChartSeriesType.Pie;
public string[] DrilledMembers = new string[] { "FY 2015" };
public List<ProductDetails> Data { get; set; }
protected override void OnInitialized()
{
this.Data = ProductDetails.GetProductData();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
List<DropDownData> ChartTypes = new List<DropDownData>() {
new DropDownData { Name = "Pie", Value = ChartSeriesType.Pie },
new DropDownData { Name = "Doughnut", Value = ChartSeriesType.Doughnut },
new DropDownData { Name = "Funnel", Value = ChartSeriesType.Funnel },
new DropDownData { Name = "Pyramid", Value = ChartSeriesType.Pyramid }
};
public class DropDownData
{
public string Name { get; set; }
public ChartSeriesType Value { get; set; }
}

```

```
}
}
```



Drill Down/Up

In the accumulation charts, drill down and drill up operations can be performed using the built-in context menu option. It will be shown while clicking on the chart series. The context menu has the following options:

Expand - It is to drill down the corresponding series until the last level.

Collapse - It is to drill up the corresponding series until the first level.

Exit - It is to close the context menu.

The drill operation in accumulation charts can be performed only for row headers.

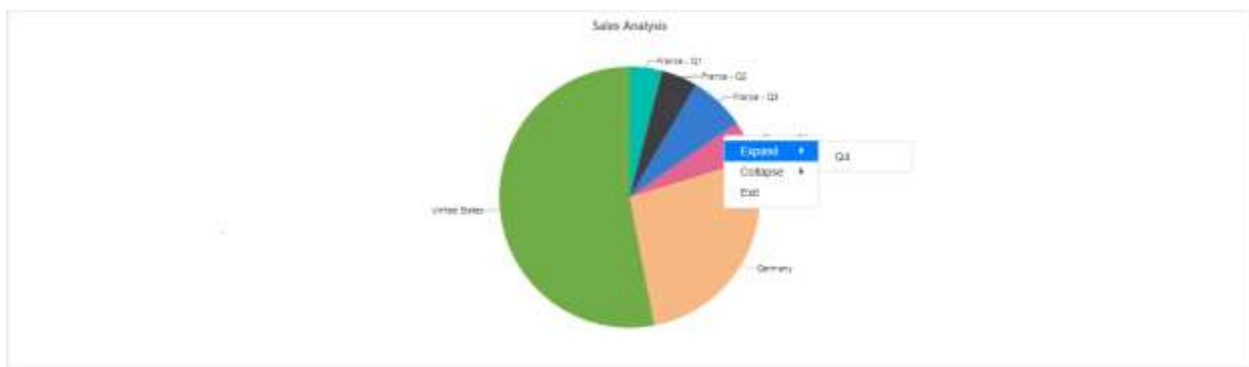
ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
  <PivotViewDataSourceSettings DataSource="@data" ExpandAll=false
  EnableSorting=true>
    <PivotViewColumns>
      <PivotViewColumn Name="Products"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Country"></PivotViewRow>
      <PivotViewRow Name="Quarter"></PivotViewRow>
      <PivotViewRow Name="Year"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Amount" Caption="Sales Amount"></PivotViewValue>
    </PivotViewValues>
    <PivotViewFormatSettings>
      <PivotViewFormatSetting Name="Amount" Format="C"
      UseGrouping=true></PivotViewFormatSetting>
    </PivotViewFormatSettings>
  </PivotViewDataSourceSettings>
  <PivotViewDisplayOption View=View.Chart></PivotViewDisplayOption>
```

```

<PivotChartSettings Title="Sales Analysis">
<PivotChartSeries Type="@ChartType">
</PivotChartSeries>
</PivotChartSettings>
</SfPivotView>
@code{
public ChartSeriesType ChartType = ChartSeriesType.Pie;
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
}

```



Column Headers and Delimiters

Unlike other chart types, the accumulation charts consider the values of a single column from the pivot table to be drawn. Preferably the first column of the pivot table is considered by default. But it can be changed by defining the column headers using the `ColumnHeader` property in [PivotChartSettings](#) class.

If the column has more than one header, then need to mention all the headers separated by the delimiter -, for example, **Germany-Road Bikes**. Using the property `ColumnDelimiter` in [PivotChartSettings](#) class, one can set the desired delimiter to separate the column headers.

ASPX-CS

```

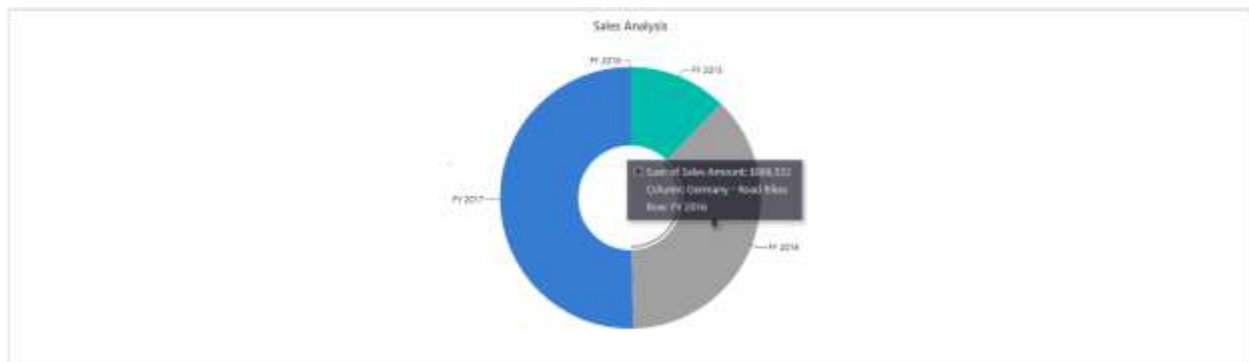
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
<PivotViewDataSourceSettings DataSource="@data" ExpandAll=false
EnableSorting=true>
<PivotViewColumns>
<PivotViewColumn Name="Country"></PivotViewColumn>
<PivotViewColumn Name="Products"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Year"></PivotViewRow>
<PivotViewRow Name="Quarter"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Amount" Caption="Sales Amount"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>

```

```

<PivotViewFormatSetting Name="Amount" Format="C"
UseGrouping=true></PivotViewFormatSetting>
</PivotViewFormatSettings>
<PivotViewDrilledMembers>
<PivotViewDrilledMember Name="Country" Items="@ (new string[] { "Germany"
}) "></PivotViewDrilledMember>
</PivotViewDrilledMembers>
</PivotViewDataSourceSettings>
<PivotViewDisplayOption View=View.Chart></PivotViewDisplayOption>
<PivotChartSettings Title="Sales Analysis" ColumnHeader="Germany-Road Bikes"
ColumnDelimiter="-">
<PivotChartSeries Type="@ChartType"></PivotChartSeries>
</PivotChartSettings>
</SfPivotView>
@code{
public ChartSeriesType ChartType = ChartSeriesType.Doughnut;
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
}

```



Label Customization

The data labels are visible by default showing header name. Its visibility can be modified using the **Visible** boolean property in **DataLabel** Tag. With regard to the label arrangement, the **Smart Labels** options help to arrange labels efficiently without overlapping. It can be disabled by setting the **EnableSmartLabels** property in **PivotChartSettings** class as **false**.

The **position** property in **dataLabel** allows to specify the position of the data label. The available options are,

- **Outside:** Positions the label outside the point. It is the default option.
- **Inside:** Positions the label inside the point.

In the following code sample, the data labels are placed inside.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
```



```

<SfPivotView TValue="ProductDetails">
  <PivotViewDataSourceSettings DataSource="@data" ExpandAll=false
  EnableSorting=true>
    <PivotViewColumns>
      <PivotViewColumn Name="Year"></PivotViewColumn>
      <PivotViewColumn Name="Products"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Country"></PivotViewRow>
      <PivotViewRow Name="Quarter"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Amount" Caption="Sales Amount"></PivotViewValue>
    </PivotViewValues>
    <PivotViewFormatSettings>
      <PivotViewFormatSetting Name="Amount" Format="C"
      UseGrouping=true></PivotViewFormatSetting>
    </PivotViewFormatSettings>
  </PivotViewDataSourceSettings>
  <PivotViewDisplayOption View=View.Chart></PivotViewDisplayOption>
  <PivotChartSettings Title="Sales Analysis" EnableSmartLabels="false">
    <PivotChartSeries Type="@ChartType">
      <PivotChartDataLabel Position="@Position"></PivotChartDataLabel>
    </PivotChartSeries>
  </PivotChartSettings>
</SfPivotView>
@code{
public ChartSeriesType ChartType = ChartSeriesType.Pyramid;
public PivotChartDataLabelPosition Position = PivotChartDataLabelPosition.Inside;
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
  this.data = ProductDetails.GetProductData().ToList();
  //Bind the data source collection here. Refer "Assigning sample data to the
  pivot table" section in getting started for more details.
}
}

```



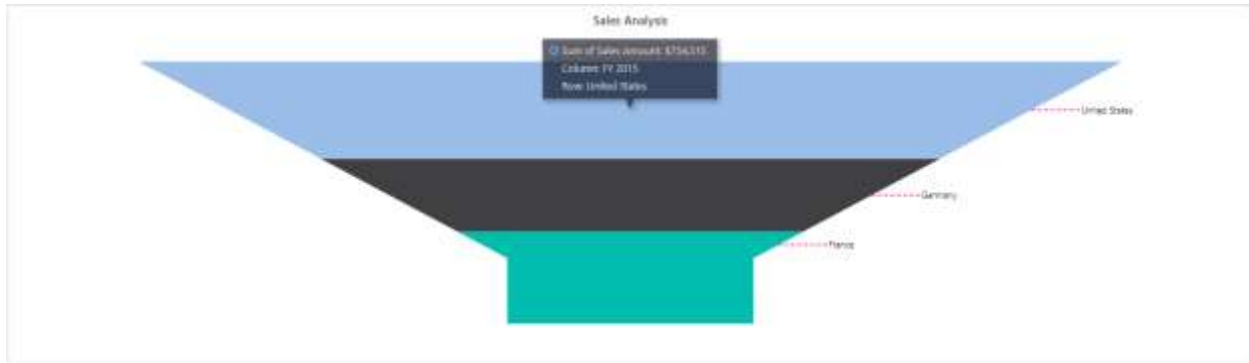
The **Connector Line** will be visible when the data label is placed outside the chart. It can be customized using the **ConnectorStyle** property in **PivotChartDataLabel** class for its color, length, width etc. In the following code sample, the connector line is customized.

ASPX-CS

```

@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
<PivotViewDataSourceSettings DataSource="@data" ExpandAll=false
EnableSorting=true>
<PivotViewColumns>
<PivotViewColumn Name="Year"></PivotViewColumn>
<PivotViewColumn Name="Products"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Country"></PivotViewRow>
<PivotViewRow Name="Quarter"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Amount" Caption="Sales Amount"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="Amount" Format="C"
UseGrouping=true></PivotViewFormatSetting>
</PivotViewFormatSettings>
</PivotViewDataSourceSettings>
<PivotViewDisplayOption View=View.Chart></PivotViewDisplayOption>
<PivotChartSettings Title="Sales Analysis" EnableSmartLabels="true">
<PivotChartSeries Type="@ChartType">
<PivotChartDataLabel Position="@Position">
<PivotChartConnectorStyle Color="#f4429e" Length="50px" Width="2"
DashArray="5,3">
</PivotChartConnectorStyle>
</PivotChartDataLabel>
</PivotChartSeries>
</PivotChartSettings>
</SfPivotView>
@code{
public ChartSeriesType ChartType = ChartSeriesType.Funnel;
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
}

```



Pie and Doughnut Customization

User can draw pie and doughnut charts within the specified range using the `StartAngle` and `EndAngle` properties in `PivotChartSeries` class. The default value of the `StartAngle` property is **0**, and the `EndAngle` property is **360**. By customizing these properties, user can draw semi pie and semi doughnut charts.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
  <PivotViewDataSourceSettings DataSource="@data" ExpandAll=false
  EnableSorting=true>
    <PivotViewColumns>
      <PivotViewColumn Name="Year"></PivotViewColumn>
      <PivotViewColumn Name="Products"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Country"></PivotViewRow>
      <PivotViewRow Name="Quarter"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Amount" Caption="Sales Amount"></PivotViewValue>
    </PivotViewValues>
    <PivotViewFormatSettings>
      <PivotViewFormatSetting Name="Amount" Format="C"
      UseGrouping=true></PivotViewFormatSetting>
    </PivotViewFormatSettings>
  </PivotViewDataSourceSettings>
  <PivotViewDisplayOption View=View.Chart></PivotViewDisplayOption>
  <PivotChartSettings Title="Sales Analysis">
    <PivotChartSeries Type="@ChartType" StartAngle="270"
    EndAngle="90"></PivotChartSeries>
  </PivotChartSettings>
</SfPivotView>

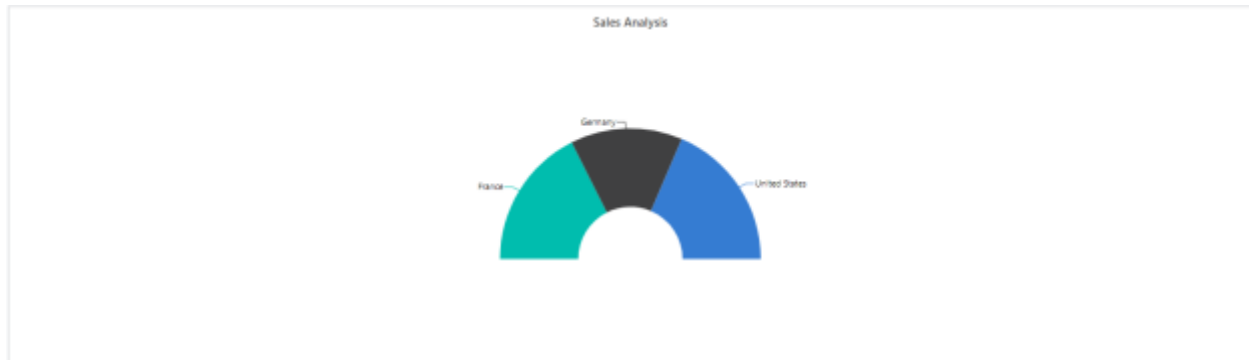
@code{
public ChartSeriesType ChartType = ChartSeriesType.Doughnut;
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{

```

```

this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
}

```



Users can get doughnut chart from pie chart and vice-versa using the `InnerRadius` property in [PivotChartSeries](#) class. If the property is greater than 0 percent, the doughnut chart will appear from the pie chart.

It takes the value only in percentage.

ASPX-CS

```

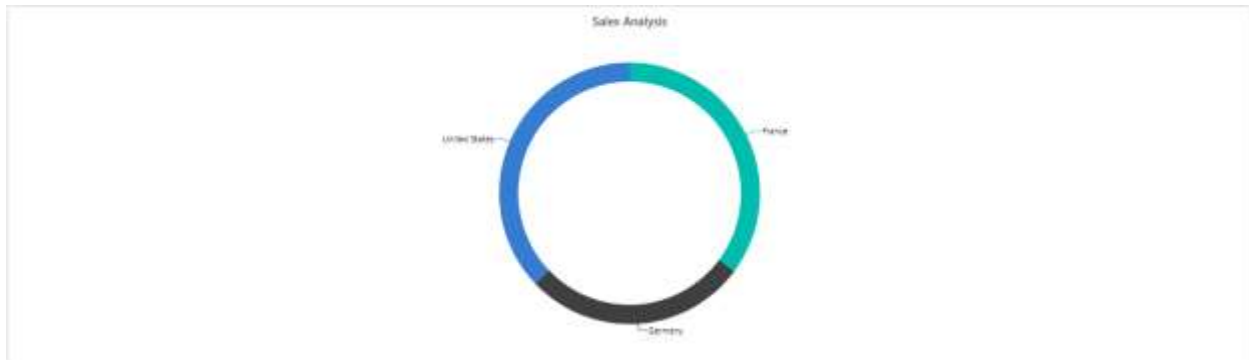
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
  <PivotViewDataSourceSettings DataSource="@data" ExpandAll=false
  EnableSorting=true>
    <PivotViewColumns>
      <PivotViewColumn Name="Year"></PivotViewColumn>
      <PivotViewColumn Name="Products"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Country"></PivotViewRow>
      <PivotViewRow Name="Quarter"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Amount" Caption="Sales Amount"></PivotViewValue>
    </PivotViewValues>
    <PivotViewFormatSettings>
      <PivotViewFormatSetting Name="Amount" Format="C"
      UseGrouping=true></PivotViewFormatSetting>
    </PivotViewFormatSettings>
  </PivotViewDataSourceSettings>
  <PivotViewDisplayOption View=View.Chart></PivotViewDisplayOption>
  <PivotChartSettings Title="Sales Analysis">
    <PivotChartSeries Type="@ChartType" InnerRadius="140">

```

```

</PivotChartSettings>
</SfPivotView>
@code{
public ChartSeriesType ChartType = ChartSeriesType.Pie;
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
}

```



Exploding Series Points

Exploding can be enabled by setting the **Explode** property in [PivotChartSeries](#) class to **true**. The series points will be exploded either on mouse click or touch.

ASPX-CS

```

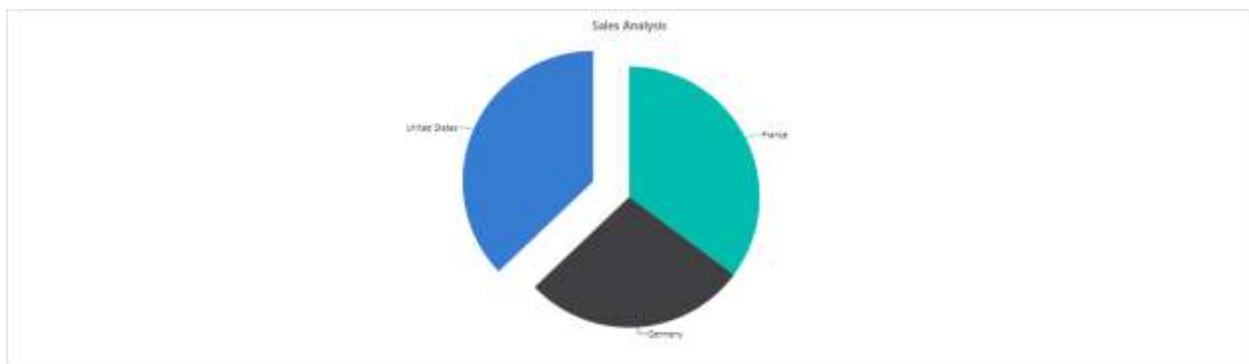
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
<PivotViewDataSourceSettings DataSource="@data" ExpandAll=false
EnableSorting=true>
<PivotViewColumns>
<PivotViewColumn Name="Year"></PivotViewColumn>
<PivotViewColumn Name="Products"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Country"></PivotViewRow>
<PivotViewRow Name="Quarter"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Amount" Caption="Sales Amount"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="Amount" Format="C"
UseGrouping=true></PivotViewFormatSetting>
</PivotViewFormatSettings>

```

```

</PivotViewDataSourceSettings>
<PivotViewDisplayOption View=View.Chart></PivotViewDisplayOption>
<PivotChartSettings Title="Sales Analysis">
<PivotChartSeries Type="@ChartType" Explode="true">
</PivotChartSeries>
</PivotChartSettings>
</SfPivotView>
@code{
public ChartSeriesType ChartType = ChartSeriesType.Pie;
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
}

```



Field List

User can enable the field list by setting the property [ShowFieldList](#) in [SfPivotView](#) class as **true**.

By using this, user can customize the report dynamically and view the result in pivot chart. For more information regarding the field list, refer the [field list](#) topic.

ASPX-CS

```

@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails" ShowFieldList="true">
<PivotViewDataSourceSettings DataSource="@data">
<PivotViewColumns>
<PivotViewColumn Name="Country"></PivotViewColumn>
<PivotViewColumn Name="Products"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Year"></PivotViewRow>
<PivotViewRow Name="Quarter"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>

```

```

<PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
</PivotViewFormatSettings>
</PivotViewDataSourceSettings>
<PivotViewDisplayOption View=View.Chart></PivotViewDisplayOption>
<PivotChartSettings Title="Sales Analysis">
<PivotChartSeries Type=ChartSeriesType.Column></PivotChartSeries>
</PivotChartSettings>
</SfPivotView>
@code{
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
}

```



Grouping Bar

User can enable the grouping bar by setting the property [ShowGroupingBar](#) in [SfPivotView](#) class as **true**. The grouping bar in pivot chart shows a dropdown list in value axis instead of buttons. The dropdown list holds list of value fields bounded in the [PivotViewDataSourceSettings](#) and it can be switched to draw the pivot chart with the selected value field. This has been defined as the default behavior in the pivot chart component. For more information regarding the grouping bar, refer the [grouping bar](#) topic.

For multiple axis support, buttons will be placed in value axis instead of dropdown list.

ASPX-CS

```

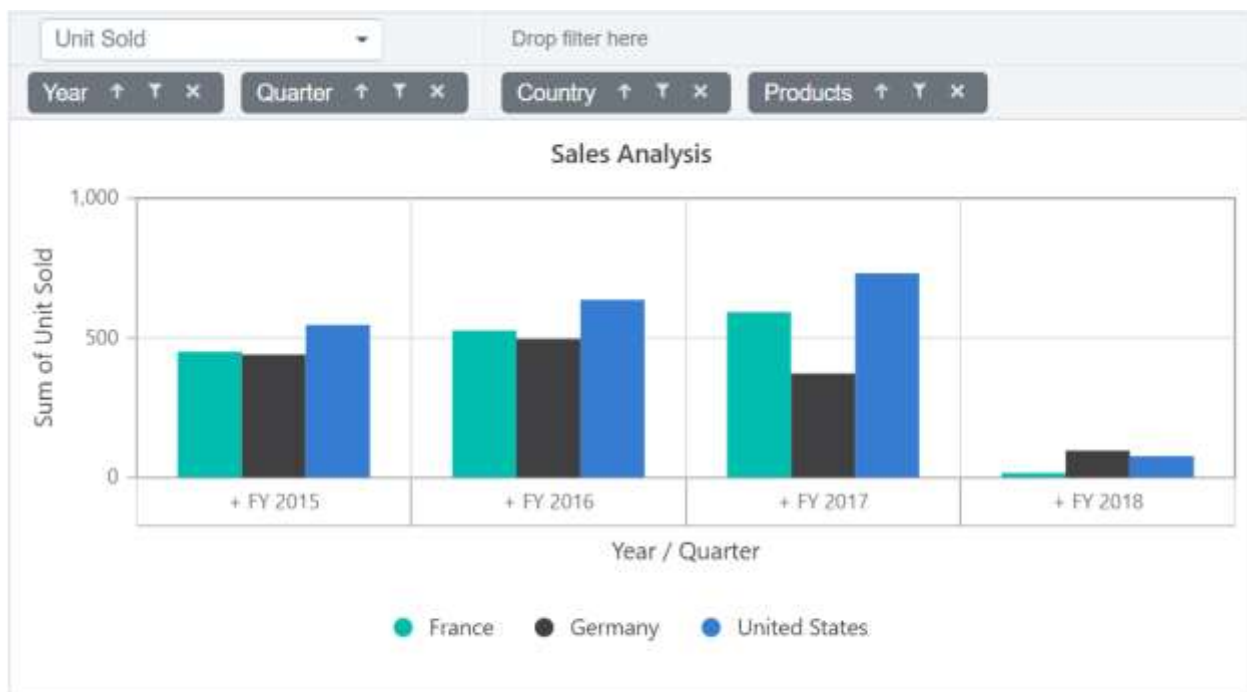
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails" Width="100%" Height="300"
ShowGroupingBar="true">
<PivotViewDataSourceSettings DataSource="@data">
<PivotViewColumns>

```

```

<PivotViewColumn Name="Country"></PivotViewColumn>
<PivotViewColumn Name="Products"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Year"></PivotViewRow>
<PivotViewRow Name="Quarter"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
<PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
</PivotViewFormatSettings>
</PivotViewDataSourceSettings>
<PivotViewDisplayOption View=View.Chart></PivotViewDisplayOption>
<PivotChartSettings Title="Sales Analysis">
<PivotChartSeries Type=ChartSeriesType.Column></PivotChartSeries>
</PivotChartSettings>
</SfPivotView>
@code{
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
}

```



For accumulation charts alone, a drop-down list will be placed in the column axis instead of the buttons. The drop-down list shows the column headers available in the pivot table. Users can dynamically switch

column headers with the help of the drop-down list, and the accumulation chart will be updated accordingly.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails" ShowGroupingBar="true">
  <PivotViewDataSourceSettings DataSource="@data" ExpandAll=false
  EnableSorting=true>
    <PivotViewColumns>
      <PivotViewColumn Name="Year"></PivotViewColumn>
      <PivotViewColumn Name="Products"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Country"></PivotViewRow>
      <PivotViewRow Name="Quarter"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Amount" Caption="Sales Amount"></PivotViewValue>
    </PivotViewValues>
    <PivotViewFormatSettings>
      <PivotViewFormatSetting Name="Amount" Format="C"
      UseGrouping=true></PivotViewFormatSetting>
    </PivotViewFormatSettings>
  </PivotViewDataSourceSettings>
  <PivotViewDisplayOption View=View.Chart></PivotViewDisplayOption>
  <PivotChartSettings Title="Sales Analysis">
    <PivotChartSeries Type="@ChartType">
    </PivotChartSeries>
  </PivotChartSettings>
</SfPivotView>

@code{
public ChartSeriesType ChartType = ChartSeriesType.Pie;
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
  this.data = ProductDetails.GetProductData().ToList();
  //Bind the data source collection here. Refer "Assigning sample data to the
  pivot table" section in getting started for more details.
}
}
```



Single Axis

By default, the pivot chart will be drawn with the value field (measure) which is set first in the report under value axis. But, user can change to specific value field using the property [Value](#) in [PivotChartSettings](#) class.

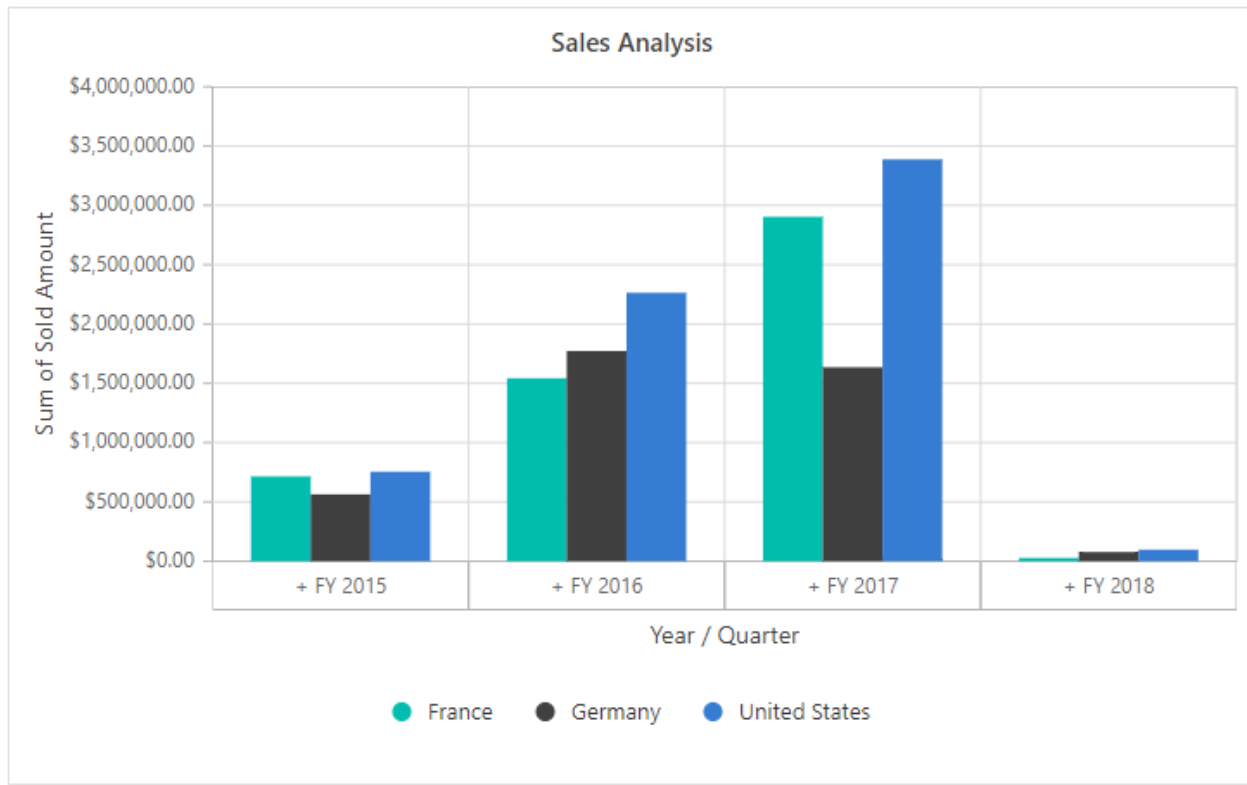
ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
  <PivotViewDataSourceSettings DataSource="@data">
    <PivotViewColumns>
      <PivotViewColumn Name="Country"></PivotViewColumn>
      <PivotViewColumn Name="Products"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Year"></PivotViewRow>
      <PivotViewRow Name="Quarter"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
      <PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
    </PivotViewValues>
    <PivotViewFormatSettings>
      <PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
    </PivotViewFormatSettings>
  </PivotViewDataSourceSettings>
  <PivotViewDisplayOption View=View.Chart></PivotViewDisplayOption>
  <PivotChartSettings Value="Amount" Title="Sales Analysis">
    <PivotChartSeries Type=ChartSeriesType.Column></PivotChartSeries>
  </PivotChartSettings>
</SfPivotView>

@code{
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
  this.data = ProductDetails.GetProductData().ToList();
  //Bind the data source collection here. Refer "Assigning sample data to the
  pivot table" section in getting started for more details.
}
```

```
}

```



Multi Axis

User can draw the pivot chart with multiple value fields by setting the property [EnableMultiAxis](#) in [PivotChartSettings](#) class as **true**. In the below code sample, the pivot chart will be drawn with both value fields "Sold" and "Amount" available in the [PivotViewDataSourceSettings](#).

The multi axis support is not applicable for the accumulation chart types like pie, doughnut, pyramid, and funnel.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
  <PivotViewDataSourceSettings DataSource="@data">
    <PivotViewColumns>
      <PivotViewColumn Name="Country"></PivotViewColumn>
      <PivotViewColumn Name="Products"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Year"></PivotViewRow>
      <PivotViewRow Name="Quarter"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
      <PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
    </PivotViewValues>
  </PivotViewDataSourceSettings>
  <PivotViewDisplayOption View=View.Chart></PivotViewDisplayOption>
</SfPivotView>
```

```

<PivotChartSettings EnableMultiAxis="true"></PivotChartSettings>
</SfPivotView>
@code{
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
}

```



If the user binds more value fields, the result will be multiple pivot charts, and each chart will shrink within the parent container height. To avoid this, set the [EnableScrollOnMultiAxis](#) property in [PivotChartSettings](#) to **true**. By doing so, each pivot chart will only shrink to a minimal "160px" – "180px" height showing a vertical scrollbar for a clear view.

ASPX-CS

```

@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
<PivotViewDataSourceSettings DataSource="@data">
<PivotViewColumns>
<PivotViewColumn Name="Country"></PivotViewColumn>
<PivotViewRows>
<PivotViewRow Name="Year"></PivotViewRow>
<PivotViewRow Name="Quarter"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
<PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
<PivotViewColumn Name="Products" Type="Count"></PivotViewColumn>

```

```

</PivotViewColumns>
</PivotViewValues>
</PivotViewDataSourceSettings>
<PivotViewDisplayOption View=View.Chart></PivotViewDisplayOption>
<PivotChartSettings EnableMultiAxis="true"
EnableScrollOnMultiAxis="true"></PivotChartSettings>
</SfPivotView>
@code{
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
}

```



Meanwhile, there is another way to display multiple values in a chart. In this approach, the series drawn from multiple values are grouped and displayed in a single chart. And, based on the values, multiple Y axis scales will be framed with different ranges. This can be achieved by setting the properties [EnableMultipleAxis](#) as `true` and [MultipleAxisMode](#) as `Grouped` in `PivotChartSettings`.

In the following code sample, the pivot chart can be seen as a single chart with multiple value fields such as `Sold` and `Amount` that are drawn as multiple Y axis.

ASPX-CS

```

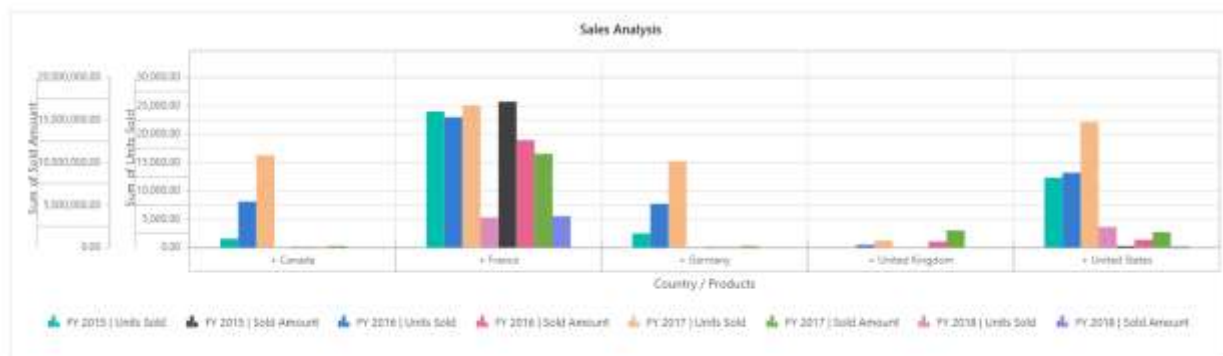
@using Syncfusion.Blazor.PivotView
<SfPivotView ID="PivotView" TValue="ProductDetails" Height="400"
Width="1400">
<PivotViewDisplayOption View="View.Chart"
Primary="Primary.Chart"></PivotViewDisplayOption>
<PivotViewDataSourceSettings DataSource="@pivotData" ExpandAll="false"
AllowLabelFilter="true" AllowMemberFilter="true" AllowValueFilter="true"
EnableSorting=true>

```

```

<PivotViewColumns>
<PivotViewColumn Name="Year"></PivotViewColumn>
<PivotViewColumn Name="Quarter"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Country"></PivotViewRow>
<PivotViewRow Name="Products"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Units Sold"></PivotViewValue>
<PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
</PivotViewValues>
</PivotViewDataSourceSettings>
<PivotChartSettings Title="Sales Analysis"
EnableMultipleAxis="enableMultiaxis"
MultipleAxisMode="MultipleAxisMode.Grouped">
<PivotChartSeries
Type=Syncfusion.Blazor.PivotView.ChartSeriesType.Column></PivotChartSeries>
</PivotChartSettings>
</SfPivotView>
@code{
public List<ProductDetails> pivotData { get; set; }
protected override void OnInitialized()
{
this.pivotData = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
}

```



Series Customization

User can customize series of the pivot chart using [PivotChartSeries](#) in [PivotChartSettings](#) class. The changes handled in the property will be reflected commonly in all chart series.

ASPX-CS

```

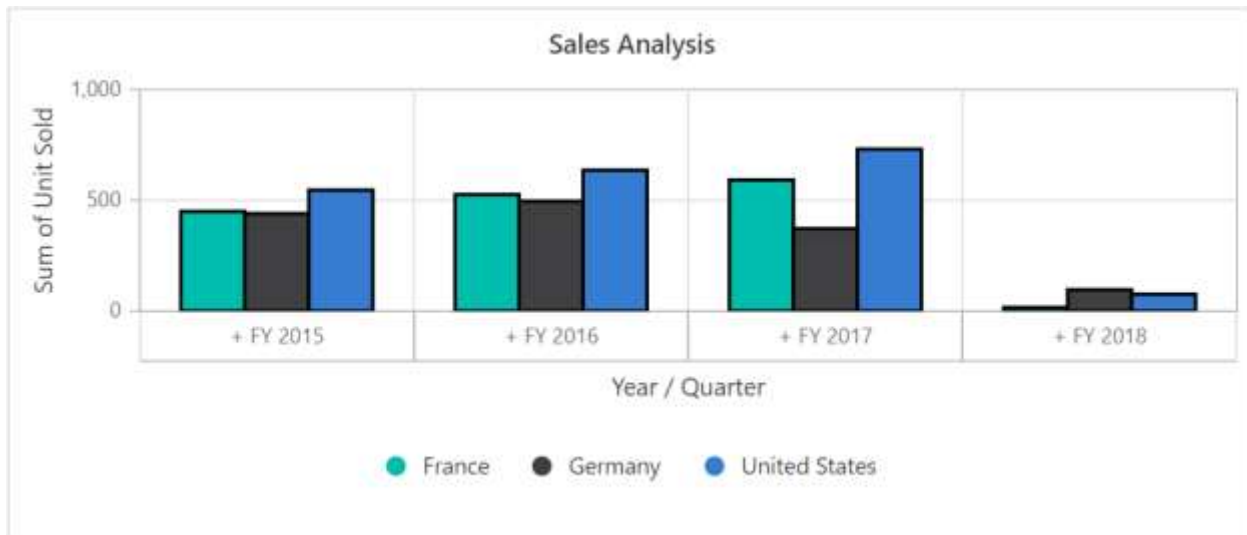
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
<PivotViewDisplayOption View=View.Chart></PivotViewDisplayOption>
<PivotViewDataSourceSettings DataSource="@data">
<PivotViewColumns>
<PivotViewColumn Name="Country"></PivotViewColumn>
<PivotViewColumn Name="Products"></PivotViewColumn>

```

```

</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Year"></PivotViewRow>
<PivotViewRow Name="Quarter"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
</PivotViewValues>
</PivotViewDataSourceSettings>
<PivotViewDisplayOption View=View.Chart></PivotViewDisplayOption>
<PivotChartSettings Title="Sales Analysis">
<PivotChartSeries Type=ChartSeriesType.Column>
<PivotChartSeriesBorder Color="#000" Width=2></PivotChartSeriesBorder>
// Other major options are:
// <PivotChartCornerRadius TopLeft=20 TopRight=20 BottomLeft=10
BottomRight=10></PivotChartCornerRadius>
// <PivotChartSeriesAnimation Enable="true"
Duration=500></PivotChartSeriesAnimation>
</PivotChartSeries>
</PivotChartSettings>
</SfPivotView>
@code{
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
}

```



Data Label Customization

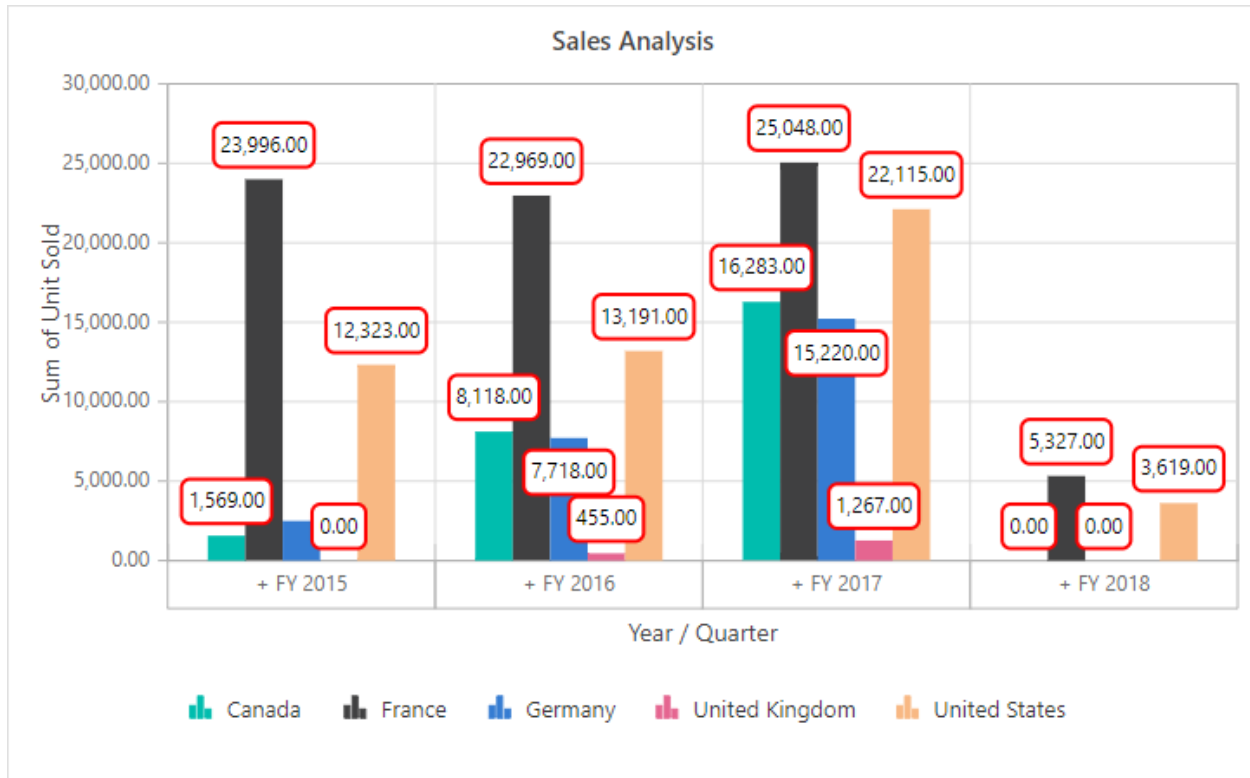
You can customize data label of the pivot chart markers in terms of angle, alignment, border, color, margin, position, visibility, and more using [PivotChartMarkerDataLabel](#) in [PivotChartMarkerSettings](#) class. The changes handled in the property will be reflected commonly in all chart series.

ASPX-CS

```

@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails" Width="800px" Height="450px">
  <PivotViewDisplayOption View=View.Chart></PivotViewDisplayOption>
  <PivotViewDataSourceSettings DataSource="@data">
    <PivotViewColumns>
      <PivotViewColumn Name="Country"></PivotViewColumn>
      <PivotViewColumn Name="Products"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Year"></PivotViewRow>
      <PivotViewRow Name="Quarter"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
      <PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
    </PivotViewValues>
    <PivotViewFormatSettings>
      <PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
    </PivotViewFormatSettings>
  </PivotViewDataSourceSettings>
  <PivotViewDisplayOption View=View.Chart></PivotViewDisplayOption>
  <PivotChartSettings Title="Sales Analysis">
    <PivotChartSeries Type="ChartSeriesType.Column">
      <PivotChartMarkerSettings>
        <PivotChartMarkerDataLabel Visible="true" Fill="white"
        Position="Syncfusion.Blazor.PivotView.LabelPosition.Auto" Rx="5" Ry="5">
          <PivotChartMarkerDataLabelBorder Width="2"
          Color="red"></PivotChartMarkerDataLabelBorder>
          <PivotChartMarkerDataLabelFont Size="12px"></PivotChartMarkerDataLabelFont>
          <PivotChartMarkerDataLabelMargin Top="5" Bottom="5" Right="5"
          Left="5"></PivotChartMarkerDataLabelMargin>
        </PivotChartMarkerDataLabel>
      </PivotChartMarkerSettings>
    </PivotChartSeries>
    <PivotChartPrimaryYAxis>
      <PivotChartPrimaryYAxisBorder Width="0"></PivotChartPrimaryYAxisBorder>
    </PivotChartPrimaryYAxis>
  </PivotChartSettings>
</SfPivotView>
@code{
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
}

```

Axis Customization

User can customize axis of the pivot chart using [PivotChartPrimaryXAxis](#) and [PivotChartPrimaryYAxis](#) properties in [PivotChartSettings](#) class.

Axis customization is not applicable for the accumulation chart types like pie, doughnut, pyramid, and funnel.

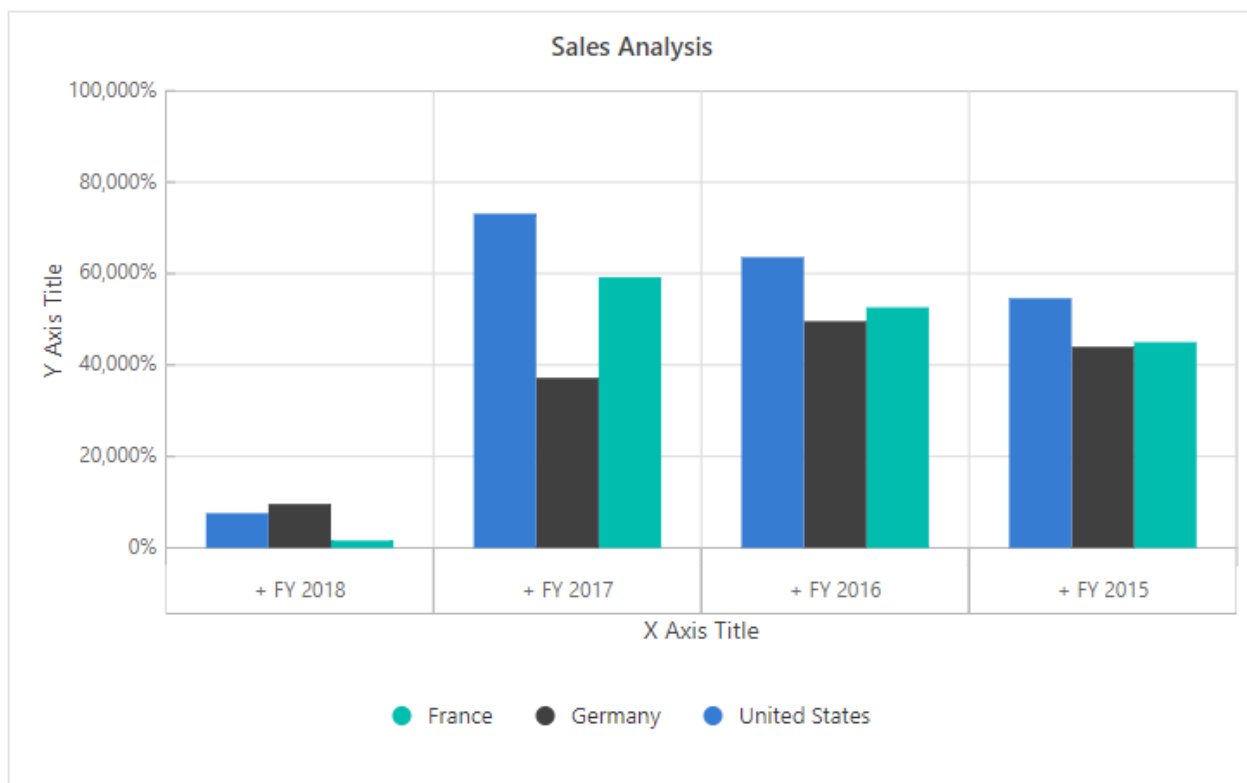
ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
  <PivotViewDataSourceSettings DataSource="@data">
    <PivotViewColumns>
      <PivotViewColumn Name="Country"></PivotViewColumn>
      <PivotViewColumn Name="Products"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Year"></PivotViewRow>
      <PivotViewRow Name="Quarter"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
    </PivotViewValues>
  </PivotViewDataSourceSettings>
  <PivotViewDisplayOption View=View.Chart></PivotViewDisplayOption>
  <PivotChartSettings Title="Sales Analysis">
    <PivotChartSeries Type=ChartSeriesType.Column></PivotChartSeries>
    <PivotChartPrimaryYAxis Title="Y Axis Title" LabelFormat="P"
      LabelPosition="PivotChartAxisPosition.Outside">
    </PivotChartPrimaryYAxis>
```

```

<PivotChartPrimaryXAxis Title="X Axis Title" IsInversed="true"
LabelPosition="PivotChartAxisPosition.Outside">
</PivotChartPrimaryXAxis>
</PivotChartSettings>
</SfPivotView>
@code{
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
}

```



Legend Customization

User can customize legend using [PivotChartLegendSettings](#) in [PivotChartSettings](#) class. By default, legend will be visible and it can be hidden by setting the property [Visible](#) in [PivotChartLegendSettings](#) class as **false**.

The pivot chart support different types of legend shapes as follows,

- Circle
- Rectangle
- VerticalLine
- Pentagon
- InvertedTriangle
- SeriesType

- Triangle
- Diamond
- Cross
- HorizontalLine

Here [PivotChartLegendShape.SeriesType](#) would act as the default shape and it can be changed using the property [LegendShape](#) in [PivotChartSeries](#) class.

Also user can set the position of the legend in pivot chart using the property [Position](#) in [PivotChartLegendSettings](#) class. The available options to set the legend position are as follows,

- Auto: Places the legend based on area type. This is the default.
- Top: Displays the legend at the top of the pivot chart.
- Left: Displays the legend at the left of the pivot chart.
- Bottom: Displays the legend at the bottom of the pivot chart.
- Right: Displays the legend at the right of the pivot chart.
- Custom: Displays the legend based on the given x and y values.

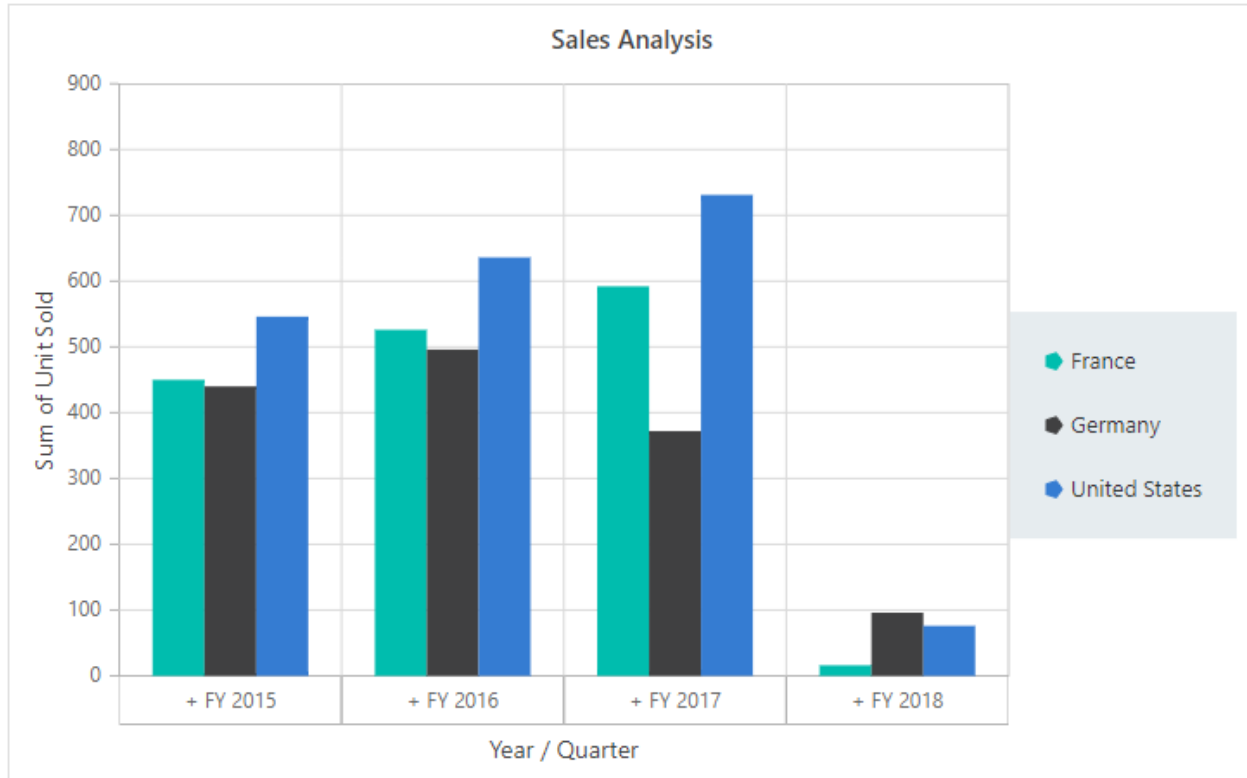
By default, the legend is not visible for the accumulation chart types like pie, doughnut, pyramid, and funnel.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
  <PivotViewDataSourceSettings DataSource="@data">
    <PivotViewColumns>
      <PivotViewColumn Name="Country"></PivotViewColumn>
      <PivotViewColumn Name="Products"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Year"></PivotViewRow>
      <PivotViewRow Name="Quarter"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
    </PivotViewValues>
  </PivotViewDataSourceSettings>
  <PivotViewDisplayOption View=View.Chart></PivotViewDisplayOption>
  <PivotChartSettings Title="Sales Analysis">
    <PivotChartLegendSettings Position=PivotChartLegendPosition.Right
      Background="rgb(230, 236, 239)" Padding=20
      Alignment="PivotChartAlignment.Center">
    </PivotChartLegendSettings>
    <PivotChartSeries Type=ChartSeriesType.Column
      LegendShape=PivotChartLegendShape.Pentagon></PivotChartSeries>
  </PivotChartSettings>
</SfPivotView>

@code{
  public List<ProductDetails> data { get; set; }
  protected override void OnInitialized()
  {
    this.data = ProductDetails.GetProductData().ToList();
  }
}
```

```
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
}
```



User Interaction

Marker and CrossHair

User can enable and customize the marker and crosshair using [PivotChartMarkerSettings](#) and [PivotChartCrosshairSettings](#) properties in [PivotChartSettings](#) class respectively.

Also user can enable and customize the crosshair tooltip for axes using [PivotChartPrimaryXAxisCrosshairTooltip](#) and [PivotChartPrimaryYAxisCrosshairTooltip](#) classes.

Marker and crosshair is not applicable for the accumulation chart types like pie, doughnut, pyramid, and funnel.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
  <PivotViewDataSourceSettings DataSource="@data">
    <PivotViewColumns>
      <PivotViewColumn Name="Country"></PivotViewColumn>
      <PivotViewColumn Name="Products"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Year"></PivotViewRow>
      <PivotViewRow Name="Quarter"></PivotViewRow>
    </PivotViewRows>
```

```

<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
</PivotViewValues>
</PivotViewDataSourceSettings>
<PivotViewDisplayOption View=View.Chart></PivotViewDisplayOption>
<PivotChartSettings>
<PivotChartCrosshairSettings Enable="true"></PivotChartCrosshairSettings>
<PivotChartSeries Type=ChartSeriesType.Line>
<PivotChartMarkerSettings Fill="#EEE" Height=10 Width=10 Visible="true"
Shape=PivotChartShape.Pentagon></PivotChartMarkerSettings>
</PivotChartSeries>
<PivotChartPrimaryXAxis>
<PivotChartPrimaryXAxisCrosshairTooltip Enable="true"
Fill="#ff0000"></PivotChartPrimaryXAxisCrosshairTooltip>
</PivotChartPrimaryXAxis>
<PivotChartPrimaryYAxis>
<PivotChartPrimaryYAxisCrosshairTooltip Enable="true"
Fill="#0000FF"></PivotChartPrimaryYAxisCrosshairTooltip>
</PivotChartPrimaryYAxis>
</PivotChartSettings>
</SfPivotView>
@code{
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
}

```



Zooming and Panning

User can customize zooming and panning option using the property [PivotChartZoomSettings](#) in [PivotChartSettings](#) class.

The pivot chart support four types of zooming which can be set as follows,

- [EnablePinchZooming](#)
- [EnableSelectionZooming](#)
- [EnableDeferredZooming](#)
- [EnableMouseWheelZooming](#)

and three modes of zooming direction that specifies whether to zoom vertically or horizontally or in both ways which are,

- x: Pivot chart can be zoomed horizontally.
- y: Pivot chart can be zoomed vertically.
- x,y: Pivot chart can be zoomed both vertically and horizontally.

This can be set using the property [Mode](#) in [PivotChartZoomSettings](#) class. By default, if the pivot chart is zoomed, a toolbar would display with the options - Zoom, ZoomIn, ZoomOut, Pan, Reset. User can also customize its option using the property [ToolBarItems](#) in [PivotChartZoomSettings](#) class.

Zooming and panning is not applicable for the accumulation chart types like pie, doughnut, pyramid, and funnel.

ASPX-CS

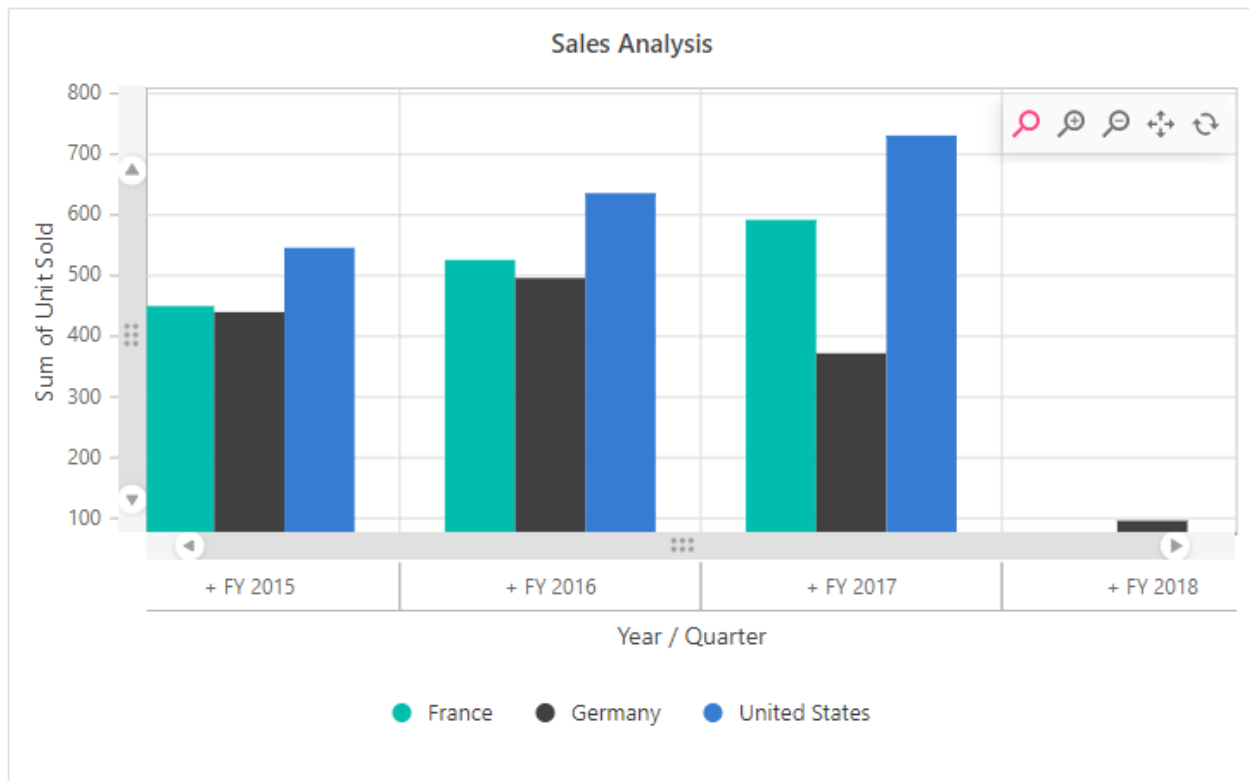
```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
  <PivotViewDataSourceSettings DataSource="@data">
    <PivotViewColumns>
      <PivotViewColumn Name="Country"></PivotViewColumn>
      <PivotViewColumn Name="Products"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Year"></PivotViewRow>
      <PivotViewRow Name="Quarter"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
    </PivotViewValues>
  </PivotViewDataSourceSettings>
  <PivotViewDisplayOption View=View.Chart></PivotViewDisplayOption>
  <PivotChartSettings Title="Sales Analysis">
    <PivotChartSeries Type=ChartSeriesType.Column></PivotChartSeries>
    <PivotChartZoomSettings ToolbarItems=@toolbar EnableDeferredZooming="true"
      EnableMouseWheelZooming="true" EnablePinchZooming="true"
      EnableSelectionZooming="true">
    </PivotChartZoomSettings>
  </PivotChartSettings>
</SfPivotView>

@code{
public List<PivotChartToolBarItems> toolbar = new
List<PivotChartToolBarItems> {
  PivotChartToolBarItems.Zoom,
  PivotChartToolBarItems.ZoomIn,
  PivotChartToolBarItems.ZoomOut,
  PivotChartToolBarItems.Pan,
  PivotChartToolBarItems.Reset
};
```

```

public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
    this.data = ProductDetails.GetProductData().ToList();
    //Bind the data source collection here. Refer "Assigning sample data to the
    pivot table" section in getting started for more details.
}
}

```



Tooltip

By default, tooltip for the pivot chart is enabled. User can customize it by using the property [PivotChartTooltipSettings](#) in [PivotChartSettings](#) class.

The tooltip can be disabled by setting the property [Enable](#) in [PivotChartTooltipSettings](#) class as **false**.

ASPX-CS

```

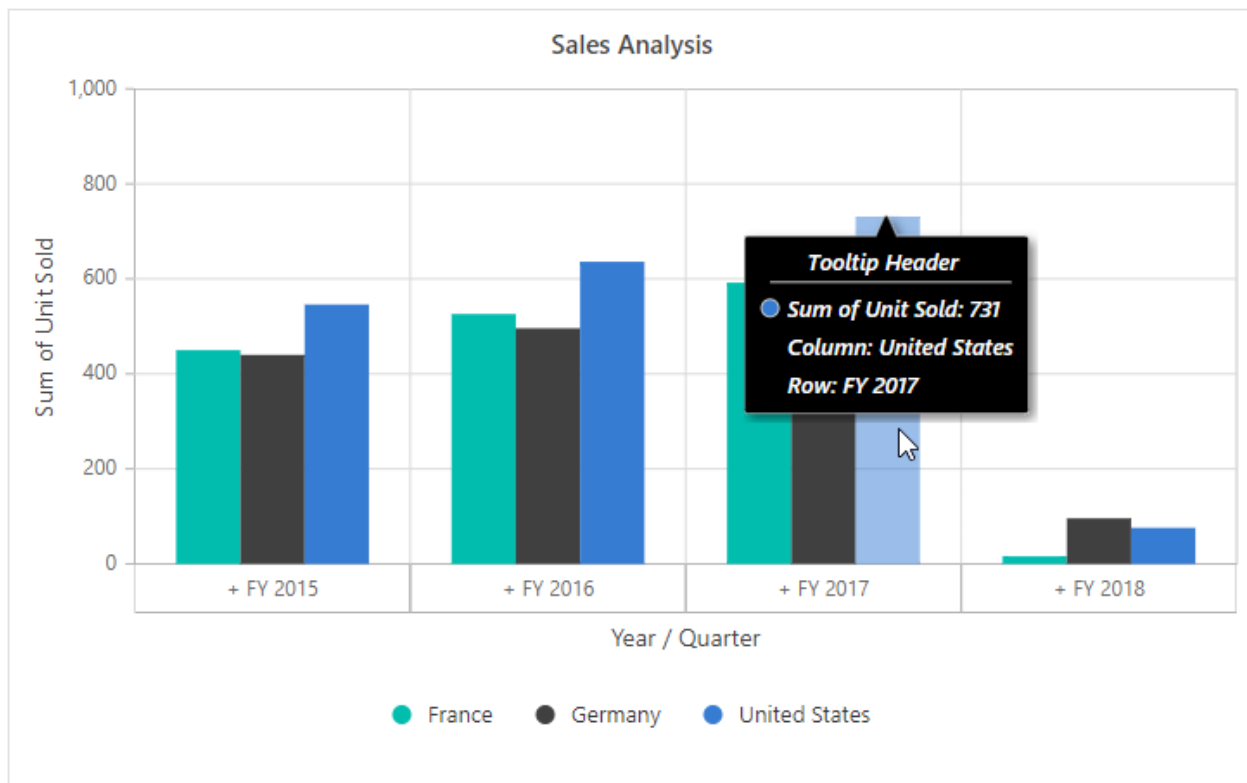
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
  <PivotViewDataSourceSettings DataSource="@data">
    <PivotViewColumns>
      <PivotViewColumn Name="Country"></PivotViewColumn>
      <PivotViewColumn Name="Products"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Year"></PivotViewRow>
      <PivotViewRow Name="Quarter"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
    </PivotViewValues>
  </PivotViewDataSourceSettings>
</SfPivotView>

```

```

</PivotViewValues>
</PivotViewDataSourceSettings>
<PivotViewDisplayOption View=View.Chart></PivotViewDisplayOption>
<PivotChartSettings Title="Sales Analysis">
<PivotChartSeries Type=ChartSeriesType.Column></PivotChartSeries>
<PivotChartTooltipSettings header="Tooltip Header" EnableAnimation="true"
EnableMarker="true" Fill="black" Opacity=1>
<PivotChartTooltipTextStyle Color="#FFF" FontStyle="Italic"
FontWeight=600></PivotChartTooltipTextStyle>
</PivotChartTooltipSettings>
</PivotChartSettings>
</SfPivotView>
@code{
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
}

```



Export

The pivot chart can be exported using the [ExportToChartAsync](#) method which holds parameters like export type, file name, PDF orientation, width, and height in the same order. The mandatory parameters for this method are export type and file name whereas other parameters are optional.

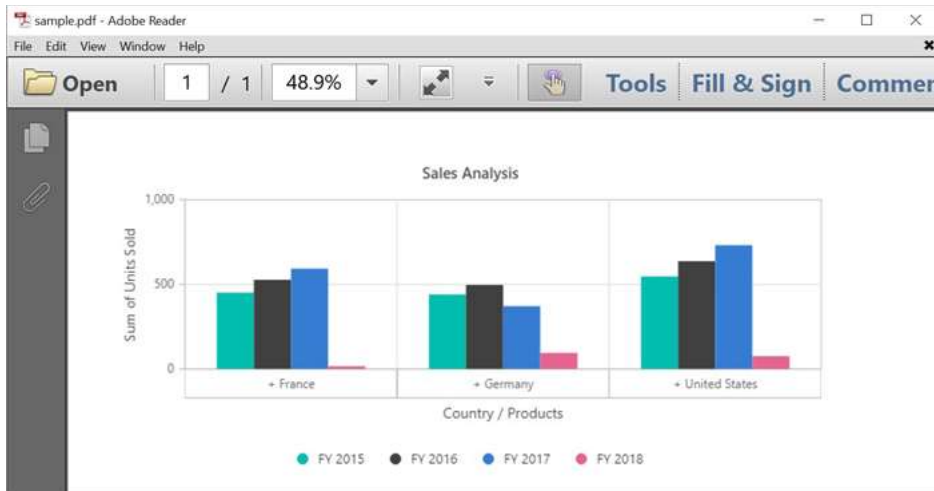
The following are the four export types:

- PNG
- JPEG
- SVG
- PDF

In the following code sample, exporting can be done using an external button named as "Chart Export".

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
@using Syncfusion.Blazor.Buttons
<SfButton OnClick="OnChartExport" Content="Chart Export"></SfButton>
<SfPivotView TValue="ProductDetails" @ref="@pivot" ShowFieldList="true">
  <PivotViewDataSourceSettings DataSource="@data">
    <PivotViewColumns>
      <PivotViewColumn Name="Country"></PivotViewColumn>
      <PivotViewColumn Name="Products"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Year"></PivotViewRow>
      <PivotViewRow Name="Quarter"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
      <PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
    </PivotViewValues>
    <PivotViewFormatSettings>
      <PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
    </PivotViewFormatSettings>
  </PivotViewDataSourceSettings>
  <PivotViewDisplayOption View=View.Chart></PivotViewDisplayOption>
  <PivotChartSettings Title="Sales Analysis">
    <PivotChartSeries Type=ChartSeriesType.Column></PivotChartSeries>
  </PivotChartSettings>
</SfPivotView>
@code{
  SfPivotView<ProductDetails> pivot;
  public List<ProductDetails> data { get; set; }
  protected override void OnInitialized()
  {
    this.data = ProductDetails.GetProductData().ToList();
    //Bind the data source collection here. Refer "Assigning sample data to the
    pivot table" section in getting started for more details.
  }
  public void OnChartExport()
  {
    this.pivot.ExportToChartAsync("PDF", "sample", "Landscape");
    //To export pivot chart to JPEG, SVG or PNG image formats, change the export
    type from "PDF" to JPEG, SVG or PNG.
    //this.pivot.ExportToChartAsync("JPEG", "sample");
  }
}
```



Print

The rendered pivot chart can be printed directly from the browser by calling [PrintChart](#) method.

In the following code sample, printing can be done using an external button named as "Print Chart".

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
@using Syncfusion.Blazor.Buttons
<SfButton OnClick="OnPrintChart" Content="Print Chart"></SfButton>
<SfPivotView TValue="ProductDetails" @ref="@pivot" ShowFieldList="true">
  <PivotViewDataSourceSettings DataSource="@data">
    <PivotViewColumns>
      <PivotViewColumn Name="Country"></PivotViewColumn>
      <PivotViewColumn Name="Products"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Year"></PivotViewRow>
      <PivotViewRow Name="Quarter"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
      <PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
    </PivotViewValues>
    <PivotViewFormatSettings>
      <PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
    </PivotViewFormatSettings>
  </PivotViewDataSourceSettings>
  <PivotViewDisplayOption View=View.Chart></PivotViewDisplayOption>
  <PivotChartSettings Title="Sales Analysis">
    <PivotChartSeries Type=ChartSeriesType.Column></PivotChartSeries>
  </PivotChartSettings>
</SfPivotView>
@code{
  SfPivotView<ProductDetails> pivot;
  public List<ProductDetails> data { get; set; }
  protected override void OnInitialized()
  {
    this.data = ProductDetails.GetProductData().ToList();
  }
}
```

```
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
public void OnPrintChart()
{
    this.pivot.PrintChart();
}
}
```



You can also explore our [Blazor Pivot Table example](#) to know how to render and configure the pivot table.

<!-- markdownlint-disable MD034 -->

Drill Down in Blazor Pivot Table Component

Drill down and drill up

The drill down and drill up action helps to view the bound data in detailed and abstract view respectively. By default, if member(s) has children, then expand and collapse icon will be displayed in the respective row/column header. On clicking the icon, expand or collapse action will be performed automatically through built-in source code. Meanwhile, leaf member(s) does not contain expand and collapse icon.

	FY 2015		FY 2016		FY 2017
	Unit Sold	Sold Amount	Unit Sold	Sold Amount	Unit Sold
▼ France	450	\$714,955	526	\$1,542,104	
Mountain Bikes	197	\$335,688	238	\$405,552	
Road Bikes	253	\$379,267	288	\$1,136,552	
► Germany	440	\$563,515	496	\$1,772,104	
► United States	546	\$754,515	636	\$2,263,104	
Grand Total	1436	\$2,032,985	1658	\$5,577,312	1

Drill position

It allows to drill only the current position of the selected member and exclude the drilled data of the selected member in other positions. For example, if "FY 2015" and "FY 2016" have "Q1" member as child in next level, and when end user attempts to drill "Q1" under "FY 2016", only it will be expanded and not "Q1" under "FY 2015".

This feature is built-in and occurs every time when expand or collapse action is done for better performance.

	FY 2015		FY 2016		
	Q1	FY 2015 Total	Q1	Road Bikes	Q1 Total
France	114	114	27	67	
Germany	74	74	97	57	
United States	144	144	57	97	
Grand Total	332	332	181	221	

Expand all

This property is applicable only for the relational data source.

It allows to either expand or collapse all headers that are displayed in row and column axes. To display all headers in expanded state, set the property [ExpandAll](#) to **true** and to collapse all

headers, set the property [ExpandAll](#) to **false**. By default, [ExpandAll](#) property is set to **false**.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
  <PivotViewDataSourceSettings DataSource="@data" ExpandAll="true">
    <PivotViewColumns>
      <PivotViewColumn Name="Year"></PivotViewColumn>
      <PivotViewColumn Name="Quarter"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Country"></PivotViewRow>
      <PivotViewRow Name="Products"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
      <PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
    </PivotViewValues>
    <PivotViewFormatSettings>
      <PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
    </PivotViewFormatSettings>
  </PivotViewDataSourceSettings>
</SfPivotView>

@code{
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
  this.data = ProductDetails.GetProductData().ToList();
  //Bind the data source collection here. Refer "Assigning sample data to the
  pivot table" section in getting started for more details.
}
}
```

	▼ FY 2015				
	Q1		Q2		Q3
	Unit Sold	Sold Amount	Unit Sold	Sold Amount	Unit Sol
▼ France	114	\$177,246	108	\$172,352	
Mountain Bikes	31	\$52,824	51	\$86,904	
Road Bikes	83	\$124,422	57	\$85,448	
▼ Germany	74	\$117,246	128	\$152,352	
Mountain Bikes	51	\$92,824	61	\$76,904	
Road Bikes	23	\$24,422	67	\$75,448	
▼ United States	144	\$162,246	171	\$145,352	
Mountain Bikes	91	\$67,824	81	\$99,904	

Expand all except specific member(s)

This option is applicable only for the relational data source.

In addition to the previous topic, there is an enhancement to expand and collapse all headers except specific header(s). To achieve this, [PivotViewDrilledMember](#) class is used. The required properties of the [PivotViewDrilledMember](#) class are explained as follows:

- [Name](#): It allows to set the field name whose member(s) needs to be specifically drilled.
- [Items](#): It allows to set the exact member(s) which needs to be drilled.

The [PivotViewDrilledMember](#) option always works in vice-versa with respect to the property [ExpandAll](#) in pivot table. For example, if [ExpandAll](#) is set to **true**, then the member(s) added in [Items](#) collection alone will be in collapsed state.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
  <PivotViewDataSourceSettings DataSource="@data" ExpandAll="true">
    <PivotViewColumns>
      <PivotViewColumn Name="Year"></PivotViewColumn>
      <PivotViewColumn Name="Quarter"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Country"></PivotViewRow>
      <PivotViewRow Name="Products"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
      <PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
    </PivotViewValues>
    <PivotViewFormatSettings>
      <PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
    </PivotViewFormatSettings>
    <PivotViewDrilledMembers>
```

```

<PivotViewDrilledMember Name="Country" Items="@ (new string[] { "France",
"Germany" }) "></PivotViewDrilledMember>
</PivotViewDrilledMembers>
</PivotViewDataSourceSettings>
</SfPivotView>
@code{
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
}

```

	▼ FY 2015				
	Q1		Q2		Q3
	Unit Sold	Sold Amount	Unit Sold	Sold Amount	Unit Sold
► France	114	\$177,246	108	\$172,352	
► Germany	74	\$117,246	128	\$152,352	
▼ United States	144	\$162,246	171	\$145,352	
Mountain Bikes	91	\$67,824	81	\$99,904	
Road Bikes	53	\$94,422	90	\$45,448	
Grand Total	332	\$456,738	407	\$470,056	

Expand specific member(s)

End user can also manually expand or collapse specific member(s) in each fields under row and column axes using the [PivotViewDrilledMember](#) class from the code behind. The required properties of the [PivotViewDrilledMember](#) class are explained as follows:

- [Name](#): It allows to set the field name whose member(s) needs to be specifically drilled.
- [Items](#): It allows to set the exact member(s) which needs to be drilled.
- [Delimiter](#): It allows to separate next level of member from its parent member.

ASPX-CS

```

@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
<PivotViewDataSourceSettings DataSource="@data">
<PivotViewRows>
<PivotViewRow Name="Year"></PivotViewRow>
<PivotViewRow Name="Quarter"></PivotViewRow>
<PivotViewRow Name="Products"></PivotViewRow>
</PivotViewRows>
<PivotViewColumns>
<PivotViewColumn Name="Country"></PivotViewColumn>
</PivotViewColumns>

```

```

<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
<PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
</PivotViewFormatSettings>
<PivotViewDrilledMembers>
<PivotViewDrilledMember Name="Quarter" Delimiter="~~" Items="@ (new string[]
{ "FY 2015~~Q1" })"></PivotViewDrilledMember>
<PivotViewDrilledMember Name="Year" Items="@ (new string[] { "FY 2015","FY
2016" })"></PivotViewDrilledMember>
</PivotViewDrilledMembers>
</PivotViewDataSourceSettings>
</SfPivotView>
@code{
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
}

```

	France		Germany		United S
	Unit Sold	Sold Amount	Unit Sold	Sold Amount	Unit Sol
▼ FY 2015	450	\$714,955	440	\$563,515	
▼ Q1	114	\$177,246	74	\$117,246	
Mountain Bikes	31	\$52,824	51	\$92,824	
Road Bikes	83	\$124,422	23	\$24,422	
► Q2	108	\$172,352	128	\$152,352	
► Q3	110	\$183,345	140	\$95,705	
► Q4	118	\$182,012	98	\$198,212	
▼ FY 2016	526	\$1,542,104	496	\$1,772,104	
► Q1	94	\$116,016	154	\$176,016	
► Q2	138	\$143,992	98	\$183,992	
► Q3	170	\$963,760	90	\$603,760	

You can refer to the [Blazor Pivot Table](#) feature tour page for its groundbreaking feature representations. You can also explore the [Blazor Pivot Table example](#) to know how to render and configure the pivot table.

Aggregation in Blazor Pivot Table Component

This feature is applicable only for relational data source.

End user can perform calculations over a group of values (exclusively for value fields bound in value axis) using the aggregation option. By default, values are added (summed) together. The other aggregation types are explained below.

The fields with data type such as number supports all aggregation types mentioned below except for **“CalculatedField”**. The fields with data type such as string, date, datetime, boolean, etc., supports **“Count”** and **“DistinctCount”** aggregation types alone.

Operator	Description
-----	-----
Sum	Displays the pivot table values with sum.
Product	Displays the pivot table values with product.
Count	Displays the pivot table values with count.
DistinctCount	Displays the pivot table values with distinct count.
Min	Displays the pivot table with minimum value.
Max	Displays the pivot table with maximum value.
Avg	Displays the pivot table values with average.
Index	Displays the pivot table values with index.
PopulationStDev	Displays the pivot table values with standard deviation of population.
SampleStDev	Displays the pivot table values with sample standard deviation.
PopulationVar	Displays the pivot table values with variance of population.
SampleVar	Displays the pivot table values with sample variance.
RunningTotals	Displays the pivot table values with running totals.
DifferenceFrom	Displays the pivot table values with difference from the value of the base item in the base field.
PercentageOfDifferenceFrom	Displays the pivot table values with percentage difference from the value of the base item in the base field.
PercentageOfGrandTotal	Displays the pivot table values with percentage of grand total of all values.
PercentageOfColumnTotal	Displays the pivot table values in each column with percentage of total values for the column.
PercentageOfRowTotal	Displays the pivot table values in each row with percentage of total values for the row.
PercentageOfParentTotal	Displays the pivot table values with percentage of total of all values based on selected field.
PercentageOfParentColumnTotal	Displays the pivot table values with percentage of its parent total in each column.
PercentageOfParentRowTotal	Displays the pivot table values with percentage of its parent total in each row.

| CalculatedField | Displays the pivot table with calculated field values. It allows user to create a new calculated field alone. |

Assigning aggregation type for value fields through API

For each value field, the aggregation type can be set using the property [Type](#) in [PivotViewValue](#) class. Meanwhile, aggregation types like [SummaryTypes.DifferenceFrom](#) and [SummaryTypes.PercentageOfDifferenceFrom](#) can check for specific field of specific item using [BaseField](#) and [BaseItem](#) properties. Likewise, [SummaryTypes.PercentageOfParentTotal](#) type can check for specific field using [BaseField](#) property. For instance, the aggregation type [SummaryTypes.DifferenceFrom](#) would intake the specified field and its corresponding member as input and its value is compared across other members in the same field and also across different fields to formulate an appropriate output value.

- [Type](#): It allows to set the aggregate type of the field.
- [BaseField](#): It allows to set the specific field to aggregate the values.
- [BaseItem](#): It allows to set the specific member to aggregate the values.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
  <PivotViewDataSourceSettings DataSource="@data">
    <PivotViewColumns>
      <PivotViewColumn Name="Year"></PivotViewColumn>
      <PivotViewColumn Name="Quarter"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Country"></PivotViewRow>
      <PivotViewRow Name="Products"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Sold" Caption="Unit Sold"
        Type="SummaryTypes.DifferenceFrom" BaseField="Country"
        BaseItem="France"></PivotViewValue>
      <PivotViewValue Name="Amount" Caption="Sold Amount"
        Type="SummaryTypes.Min"></PivotViewValue>
    </PivotViewValues>
    <PivotViewFormatSettings>
      <PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
    </PivotViewFormatSettings>
  </PivotViewDataSourceSettings>
</SfPivotView>

@code{
  public List<ProductDetails> data { get; set; }
  protected override void OnInitialized()
  {
    this.data = ProductDetails.GetProductData().ToList();
    //Bind the data source collection here. Refer "Assigning sample data to the
    pivot table" section in getting started for more details.
  }
}
```

	FY 2015		FY 2016		FY 2017
	Unit Sold	Sold Amount	Unit Sold	Sold Amount	Unit Sold
France		\$29,985		\$46,008	
Germany	-10	\$24,422	-30	\$86,008	-
United States	96	\$45,448	110	\$90,008	
Grand Total		\$24,422		\$46,008	

By default, the aggregation will be considered as [SummaryTypes.Sum](#) to the value fields which had number type and for the value fields which had non-number type such as string, date, datetime, boolean, etc., the aggregation type will be considered as [SummaryTypes.Count](#).

Show desired aggregation types in its dropdown menu

By default, all the aggregation types are displayed in the dropdown menu available in buttons. However, based on the request for an application, there may be a need to show selective aggregation types on our own. This can be achieved using the [AggregateTypes](#) property.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails" ShowGroupingBar="true"
AggregateTypes="@aggregateType">
<PivotViewDataSourceSettings DataSource="@dataSource">
<PivotViewColumns>
<PivotViewColumn Name="Year"></PivotViewColumn>
<PivotViewColumn Name="Quarter"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Country"></PivotViewRow>
<PivotViewRow Name="Products"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Units Sold"></PivotViewValue>
<PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
</PivotViewFormatSettings>
</PivotViewDataSourceSettings>
</SfPivotView>
@code{
public List<AggregateTypes> aggregateType = new List<AggregateTypes> {
AggregateTypes.DistinctCount,
AggregateTypes.Avg,
AggregateTypes.Product
};
public List<ProductDetails> dataSource { get; set; }
protected override void OnInitialized()
{
this.dataSource = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
```

```
}
}
```

The screenshot shows a pivot table interface. At the top, there are filter buttons for 'Sum of Units Sold', 'Sum of Sold Amount', 'Year', and 'Quarter'. A dropdown menu is open for 'Sum of Units Sold', showing options: 'Distinct Count', 'Avg', and 'Product'. The table below has columns for 'FY 2015' and 'FY 2016', each with 'Units Sold' and 'Sold Amount'. The rows are grouped by 'Country' (France, Germany, United States) and 'Products'. A 'Grand Total' row is at the bottom.

	FY 2015		FY 2016		
	Units Sold	Sold Amount	Units Sold	Sold Amount	Un
France	450	\$714,955	526	\$1,542,104	
Germany	440	\$563,515	496	\$1,772,104	
United States	546	\$754,515	636	\$2,263,104	
Grand Total	1436	\$2,032,985	1658	\$5,577,312	

Modifying aggregation type for value fields at runtime

Aggregation types can be changed easily through UI at runtime. The value fields bound to grouping bar and field list appears with a dropdown icon which helps to select an appropriate aggregation type for the respective value field. On selection, the values in the pivot table will be changed dynamically.

<!-- markdownlint-disable MD012 -->

The screenshot shows a pivot table interface with a 'Rows' section containing 'Country' and 'Products', and a 'Values' section containing 'Sum of Units Sold' and 'Sum of Sold Amount'. A dropdown menu is open for 'Sum of Units Sold', showing options: 'Sum', 'Count', 'Distinct Count', 'Product', 'Min', 'Max', 'Avg', and 'More...'. A 'Close' button is visible at the bottom right.

Sum of Units Sold	Drop filter here
Sum of Sold Amount	Year ↑ ▾ × Quarter ↑ ▾ ×
Country ↑	▸ FY 2015 ▸ FY 2016
Products	Units Sold Sold Amount Units Sold
▸ France	450 \$714,955 526
▸ Germany	440 \$563,515 496
▸ United States	546 \$754,515 636
Grand Total	1436 \$2,032,985 1658

Hiding aggregation type from button text

By default, in value axis each field would be displayed by its name and aggregation type together. To hide aggregation type and display field name alone, set the property [ShowAggregationOnValueField](#) in [PivotViewDataSourceSettings](#) class to **false**.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
  <PivotViewDataSourceSettings DataSource="@data"
    ShowAggregationOnValueField="false">
    <PivotViewColumns>
      <PivotViewColumn Name="Year"></PivotViewColumn>
      <PivotViewColumn Name="Quarter"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Country"></PivotViewRow>
      <PivotViewRow Name="Products"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
      <PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
    </PivotViewValues>
    <PivotViewFormatSettings>
      <PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
    </PivotViewFormatSettings>
  </PivotViewDataSourceSettings>
</SfPivotView>

@code{
  public List<ProductDetails> data { get; set; }
  protected override void OnInitialized()
  {
    this.data = ProductDetails.GetProductData().ToList();
    //Bind the data source collection here. Refer "Assigning sample data to the
    pivot table" section in getting started for more details.
  }
}
```

	Σ Values
↑ ▼	Unit Sold ▼
↑ ▼	Sold Amount ▼

Unit Sold ▼ ×	Drop filter here
Sold Amount ▼ ×	Year ↑ ▼ ×
Country ↑ ▼ ×	► FY 2015

Hiding aggregation type icon from UI

By default, the icon to set aggregation type is enabled in the grouping bar. To disable this icon, set the property [ShowValueTypeIcon](#) in [PivotViewGroupingBarSettings](#) class to **false**.

Icon to change the aggregation type can be hidden only in Grouping Bar but not in Field List at the moment.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails" ShowGroupingBar="true">
  <PivotViewDataSourceSettings DataSource="@data">
    <PivotViewColumns>
      <PivotViewColumn Name="Year"></PivotViewColumn>
      <PivotViewColumn Name="Quarter"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Country"></PivotViewRow>
      <PivotViewRow Name="Products"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
      <PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
    </PivotViewValues>
    <PivotViewFormatSettings>
      <PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
    </PivotViewFormatSettings>
  </PivotViewDataSourceSettings>
  <PivotViewGroupingBarSettings
    ShowValueTypeIcon=false></PivotViewGroupingBarSettings>
</SfPivotView>
@code{
  public List<ProductDetails> data { get; set; }
  protected override void OnInitialized()
  {
```

```

this.data = ProductDetails.GetDefaultData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
}

```



You can refer to the [Blazor Pivot Table](#) feature tour page for its groundbreaking feature representations. You can also explore the [Blazor Pivot Table example](#) to know how to render and configure the pivot table.

Number Formatting in Blazor Pivot Table Component

Allows you to specify the required display format that should be used in values of the pivot table.

Supported display formats are:

- Number
- Currency
- Percentage
- Custom

You can apply format for the numeric values using the following properties in the [FormatSettings](#).

- [Name](#): It allows to specify the field name.
- [Format](#): It allows to specify the format of the respective field.

Possible formatting values are:

1. N - denotes numeric type.
2. C - denotes currency type.
3. P - denotes percentage type.

If no format is specified it takes number as default format type.

Other properties include:

- [UseGrouping](#): It allows to enable or disable the separator, for example, \$100,000,000 or \$100000000 respectively. By default, it will be set as **true**.
- [Currency](#): It allows to set the currency code which needs to be considered for the currency formatting.

ASPX-CS

```

@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">

```

```

<PivotViewDataSourceSettings DataSource="@data">
  <PivotViewColumns>
    <PivotViewColumn Name="Year"></PivotViewColumn>
    <PivotViewColumn Name="Quarter"></PivotViewColumn>
  </PivotViewColumns>
  <PivotViewRows>
    <PivotViewRow Name="Country"></PivotViewRow>
    <PivotViewRow Name="Products"></PivotViewRow>
  </PivotViewRows>
  <PivotViewValues>
    <PivotViewValue Name="Sold" Caption="Units Sold"></PivotViewValue>
    <PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
  </PivotViewValues>
  <PivotViewFormatSettings>
    <PivotViewFormatSetting Name="Amount" Format="C2" UseGrouping="false"
    Currency='EUR'></PivotViewFormatSetting>
  </PivotViewFormatSettings>
</PivotViewDataSourceSettings>
</SfPivotView>
@code{
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
}

```

	FY 2015			FY 2016		
	Units Sold	Sold Amount	Total Amount	Units Sold	Sold Amount	Total Amount
France	450	€714955.00	450	526	€1542104.00	
Germany	440	€563515.00	440	496	€1772104.00	
United States	546	€754515.00	546	636	€2263104.00	
Grand Total	1436	€2032985.00	1436	1658	€5577312.00	

You can also format the values at runtime using the formatting dialog. This option can be enabled by setting the [AllowNumberFormatting](#) property to **true**. The same has been discussed in some of the upcoming topics.

Custom format

You can add any custom format directly to the [Format](#) property in the [FormatSettings](#). Custom format can be achieved by using one or more format specifiers listed in the below table.

Specifier	Description	Input	Format Output
0	Replaces the zero with the corresponding digit if it is present. Otherwise, zero will appear in the result string.	{ format: '0000' }	'0123'

| # | Replaces the ' # ' symbol with the corresponding digit if it is present. Otherwise, no digits will appear in the result string. | { format: '####' } | '1234' |

| . | Denotes the number of digits permitted after the decimal point. | { format: '###0.##0#' } | '546321.000' |

| % | Percent specifier denotes the percentage type format. | { format: '0000 %' } | '0100 %' |

| \$ | Denotes the currency type format that is based on the global currency code specified. | { format: '\$###.00' } | '\$ 13.00' |

| ; | Denotes separate formats for positive, negative and zero values. | { format: '###.##;(###.00);-0' } | '(120.00)' |

| 'String' (single Quotes) | Denotes the characters that are enclosed in the single quote (') to be replaced in the resulting string. | { format: '####.00 '@' ' } | "123.00 @" |

If custom format is defined, certain properties such as [UseGrouping](#) and [Currency](#) will not be considered.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
<PivotViewDataSourceSettings DataSource="@data">
<PivotViewColumns>
<PivotViewColumn Name="Year"></PivotViewColumn>
<PivotViewColumn Name="Quarter"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Country"></PivotViewRow>
<PivotViewRow Name="Products"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Units Sold"></PivotViewValue>
<PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
<PivotViewFormatSetting Name="Sold" Format="####.00
'Nos'"></PivotViewFormatSetting>
</PivotViewFormatSettings>
</PivotViewDataSourceSettings>
</SfPivotView>
@code{
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
}
```


	FY 2015			FY 2016		
	Units Sold	Sold Amount	Total Amount	Units Sold	Sold Amount	Total
France	450.00 Nos	\$714,955	450	526.00 Nos	\$1,542,104	
Germany	440.00 Nos	\$563,515	440	496.00 Nos	\$1,772,104	
United States	546.00 Nos	\$754,515	546	636.00 Nos	\$2,263,104	
Grand Total	1436.00 Nos	\$2,032,985	1436	1658.00 Nos	\$5,577,312	

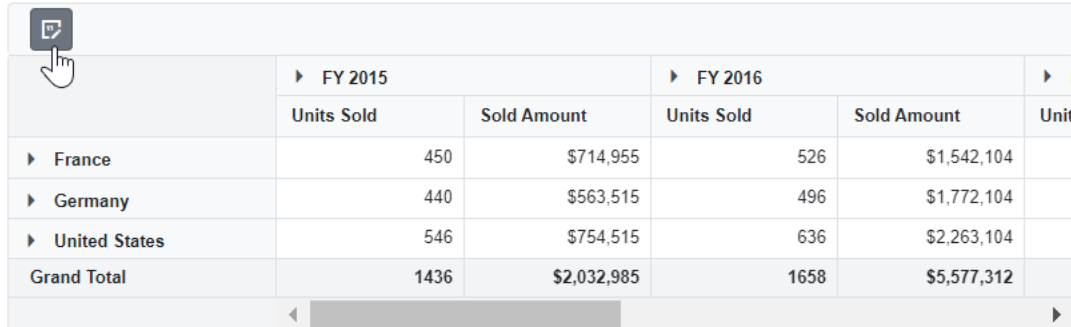
Toolbar

You can enable formatting dialog option in the toolbar by adding `NumberFormatting` in the [Toolbar](#). After that, you can see the option to invoke the formatting dialog in the toolbar.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView @ref="pivot" TValue="ProductDetails" ShowToolbar="true"
Toolbar="@toolbar" AllowNumberFormatting="true" Height="300" Width="800">
<PivotViewDisplayOption Primary=Primary.Table
View=View.Both></PivotViewDisplayOption>
<PivotViewDataSourceSettings DataSource="@data">
<PivotViewColumns>
<PivotViewColumn Name="Year"></PivotViewColumn>
<PivotViewColumn Name="Quarter"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Country"></PivotViewRow>
<PivotViewRow Name="Products"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Units Sold"></PivotViewValue>
<PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
</PivotViewFormatSettings>
</PivotViewDataSourceSettings>
<PivotViewEvents TValue="ProductDetails"></PivotViewEvents>
<PivotViewGridSettings ColumnWidth="140"></PivotViewGridSettings>
</SfPivotView>
<style>
.e-pivotview {
min-height: 300px;
width: 722px;
}
</style>
@code{
SfPivotView<ProductDetails> pivot;
public List<ToolbarItems> toolbar = new List<ToolbarItems> {
ToolbarItems.NumberFormatting
};
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
}
```

```
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
}
```



	FY 2015		FY 2016		
	Units Sold	Sold Amount	Units Sold	Sold Amount	Unit
France	450	\$714,955	526	\$1,542,104	
Germany	440	\$563,515	496	\$1,772,104	
United States	546	\$754,515	636	\$2,263,104	
Grand Total	1436	\$2,032,985	1658	\$5,577,312	

You can refer to our [Blazor Pivot Table](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor Pivot Table example](#) to know how to render and configure the pivot table.

Calculated Field in Blazor Pivot Table Component

Allows end user to create a new calculated field in the pivot table, based on the available fields from the bound data source or using simple formula with basic arithmetic operators. It can be added at runtime through the built-in dialog, invoked from Field List UI. To do so, set the [AllowCalculatedField](#) property in [SfPivotView](#) class to **true** in the pivot table. End user can now see a "CALCULATED FIELD" button enabled in Field List UI automatically, which on clicking will invoke the calculated field dialog and perform necessary operation.

Calculated field can also be included in the pivot table through code behind using the [PivotViewCalculatedFieldsSettings](#) class. The required properties to create a new calculate field are:

- [Name](#): It allows to indicate the calculated field with a unique name.
- [Formula](#): It allows to set the formula.
- [Format](#): It helps to set the number format for the resultant value.

The calculated field is applicable only for value fields. Also, the calculated field created through the code behind will be automatically listed in the UI dialog as well.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails" ShowFieldList="true"
AllowCalculatedField="true">
<PivotViewDataSourceSettings DataSource="@data">
<PivotViewColumns>
<PivotViewColumn Name="Year"></PivotViewColumn>
<PivotViewColumn Name="Quarter"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Country"></PivotViewRow>
<PivotViewRow Name="Products"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
```

```

<PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
<PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
<PivotViewValue Name="Total" Caption="Total Amount"
Type=SummaryTypes.CalculatedField></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
<PivotViewFormatSetting Name="Total" Format="C"></PivotViewFormatSetting>
</PivotViewFormatSettings>
<PivotViewCalculatedFieldSettings>
<PivotViewCalculatedFieldSetting Name="Total"
Formula="@totalPrice"></PivotViewCalculatedFieldSetting>
</PivotViewCalculatedFieldSettings>
</PivotViewDataSourceSettings>
</SfPivotView>
@code{
public string totalPrice = "\" + "Sum(Amount)" + "\" + "+" + "\" + "
"Sum(Sold)" + "\"";
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
}

```

	FY 2015			FY 2016	
	Unit Sold	Sold Amount	Total Amount	Unit Sold	Sold Ar
France	450	\$714,955	\$715,405	526	\$1,4
Germany	440	\$563,515	\$563,955	496	\$1,7
United States	546	\$754,515	\$755,061	636	\$2,2
Grand Total	1436	\$2,032,985	\$2,034,421	1658	\$5,1

Meanwhile, the user can also view calculated field dialog in UI by invoking [CreateCalculatedFieldDialogAsync](#) method on an external button click.

ASPX-CS

```

@using Syncfusion.Blazor.PivotView
@using Syncfusion.Blazor.Buttons
<SfButton OnClick="@calc" IsPrimary="true">Calculated Field</SfButton>
<SfPivotView TValue="ProductDetails" @ref="pivot"
AllowCalculatedField="true">
<PivotViewDataSourceSettings DataSource="@data">
<PivotViewColumns>
<PivotViewColumn Name="Year"></PivotViewColumn>
<PivotViewColumn Name="Quarter"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Country"></PivotViewRow>

```

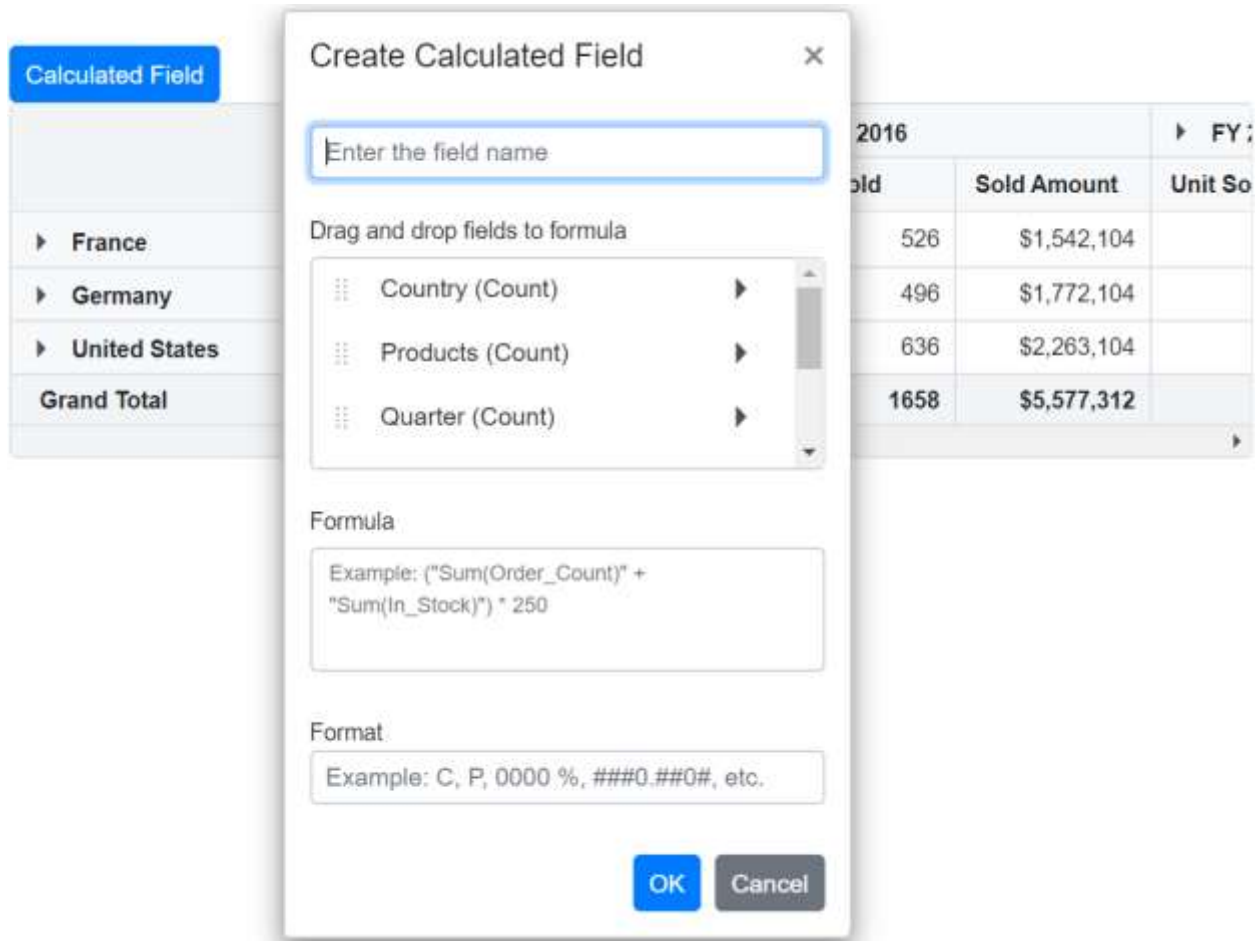
```

<PivotViewRow Name="Products"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
<PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
<PivotViewFormatSetting Name="Total" Format="C"></PivotViewFormatSetting>
</PivotViewFormatSettings>
</PivotViewDataSourceSettings>
</SfPivotView>
@code{
public SfPivotView<ProductDetails> pivot;
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
public void calc(Microsoft.AspNetCore.Components.Web.MouseEventArgs args)
{
this.pivot.CreateCalculatedFieldDialogAsync();
}
}

```

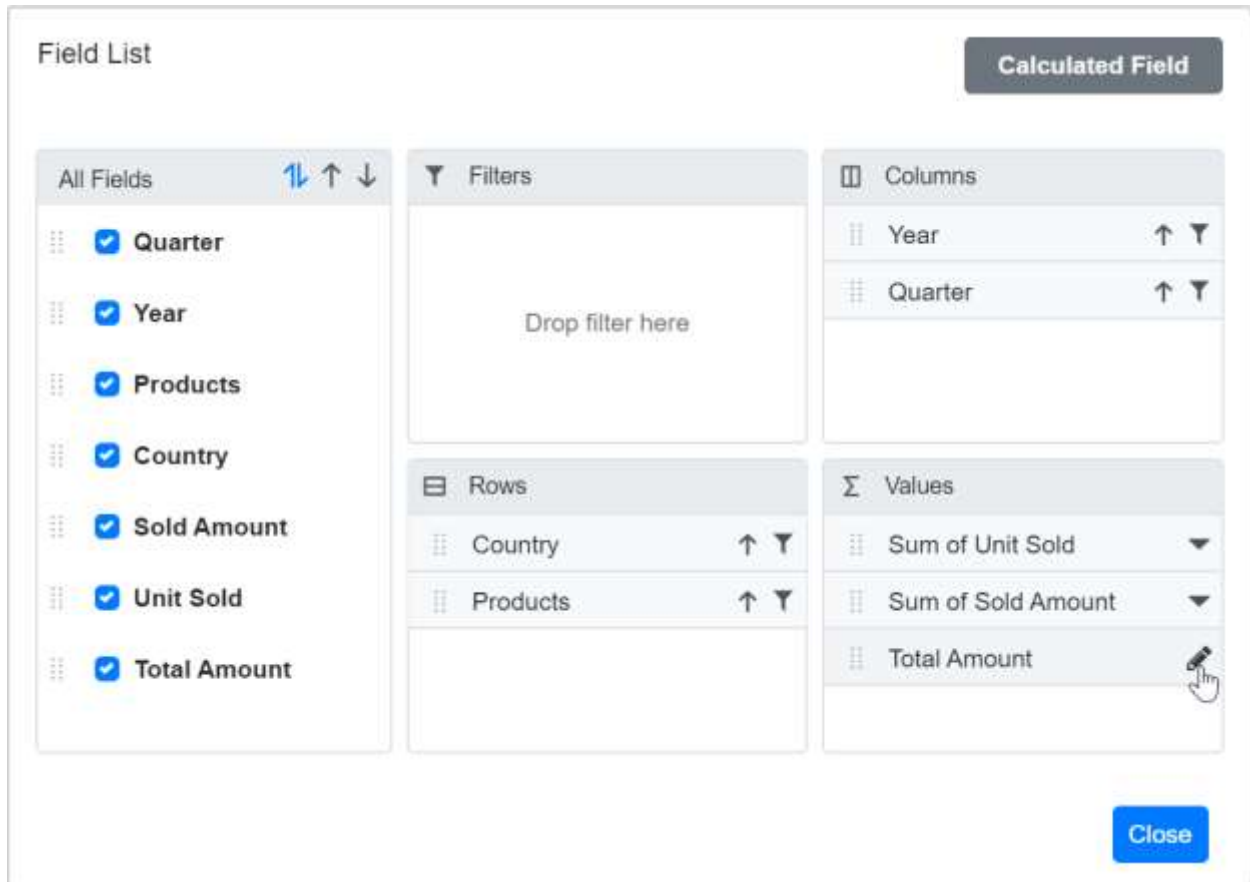
Calculated Field

	FY 2015		FY 2016		FY 2017
	Unit Sold	Sold Amount	Unit Sold	Sold Amount	Unit Sold
France	450	\$714,955	526	\$1,542,104	
Germany	440	\$563,515	496	\$1,772,104	
United States	546	\$754,515	636	\$2,263,104	
Grand Total	1436	\$2,032,985	1658	\$5,577,312	



Editing through the field list and the grouping bar


User can also modify the existing calculated field using the built-in edit option available directly in the field list (or) grouping bar. To do so, click the "Edit" icon available in the calculated field button. Now the calculated field dialog is opened and the current calculated field name, formula and format can be changed at runtime.



Create Calculated Field

Total Amount

Drag and drop fields to formula

Total Amount (Calculated F...

Unit Sold (Sum) ▶

Year (Count) ▶

Formula

"Sum(Sold)"+"Sum(Amount)"

Format

C2

OK

Cancel

Renaming the existing calculated field

Existing calculated field can be renamed only through the UI at runtime. To do so, open the calculated field dialog, select the target field and click "Edit" icon. User can now see the existing name getting displayed in the text box at the top of the dialog. Now, change the name based on user requirement and click "OK".

<!-- markdownlint-disable MD012 -->

Create Calculated Field ×

Enter the field name

Drag and drop fields to formula

Sold Amount (Sum)

Total Amount (Calculated F...

Unit Sold (Sum)

▶

Edit

Formula

Example: ("Sum(Order_Count)" +
"Sum(In_Stock)") * 250

Format

Example: C, P, 0000 %, ###0.##0#, etc.

OK

Cancel

Create Calculated Field

Total Sales Amount

Drag and drop fields to formula

- Sold Amount (Sum)
- Total Amount (Calculated F...
- Unit Sold (Sum)

Clear edited field info

Formula

"Sum(Sold)"+"Sum(Amount)"

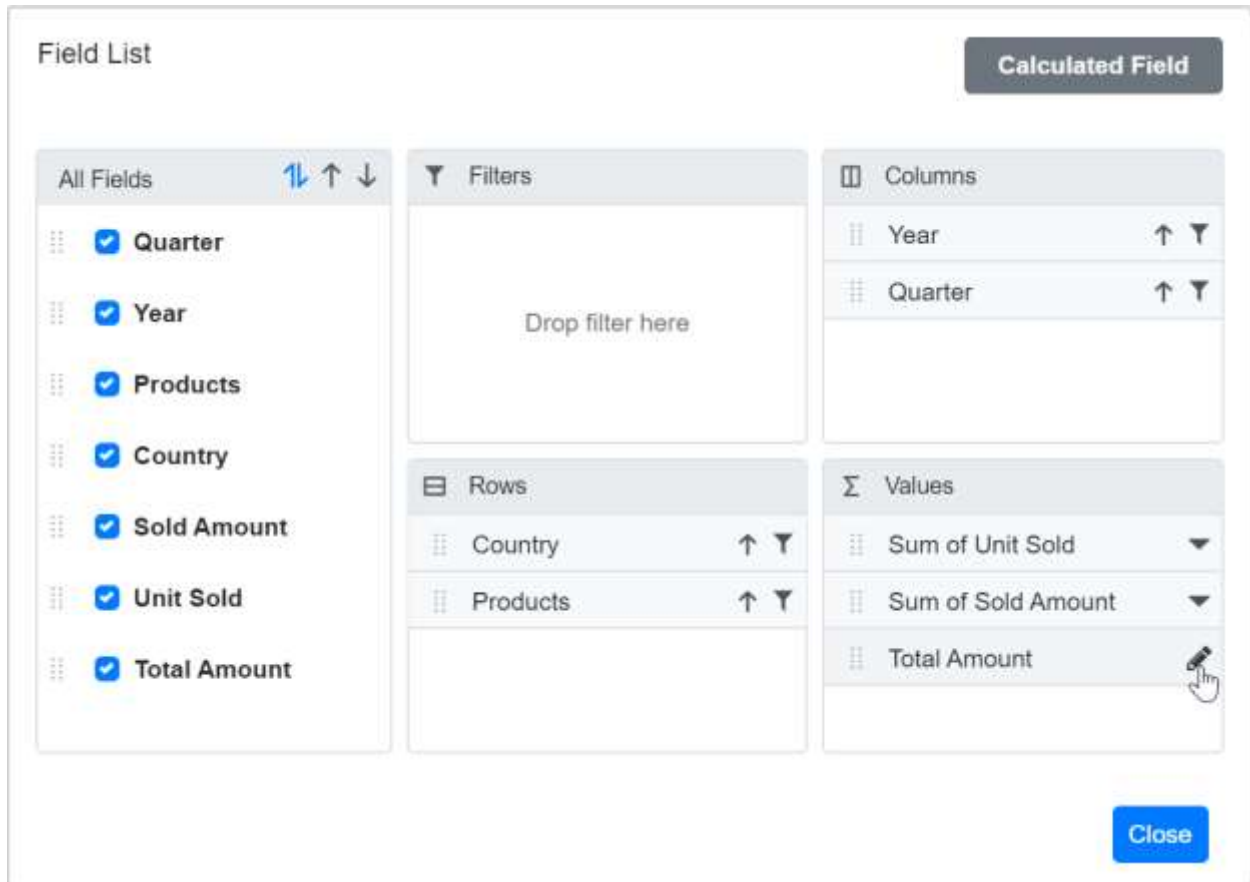
Format

Example: C, P, 0000 %, ###0.##0#, etc.

OK Cancel

Editing the existing calculated field formula

Existing calculated field formula can be edited only through the UI at runtime. To do so, open the calculated field dialog, select the target field and click "Edit" icon. User can now see the existing formula getting displayed in a multiline text box at the bottom of the dialog. Now, change the formula based on user requirement and click "OK".



Create Calculated Field

Total Units

Drag and drop fields to formula

Sold Amount (Count)

Total Amount (Calculated F...)

Unit Sold (Count)

Formula

"Count(Amount)"+"Count(Sold)"

Format

Example: C, P, 0000 %, ###0.##0#, etc.

OKCancel

Reusing the existing formula in a new calculate field

While creating a new calculated field, if the user wants to add the formula of an existing calculated field, it can be done easily. To do so, simply drag-and-drop the existing calculated field to the "Formula" section.

Create Calculated Field

Sales Amount

Drag and drop fields to formula

Sold Amount (Sum)

Total Amount (Calculated F...)

Drag field to formula

Formula

Example: ("Sum(Order_Count)" +
"Sum(In_Stock)") * 250

Format

Example: C, P, 0000 %, ###0.##0#, etc.

OK

Cancel

Create Calculated Field

Sales Amount

Drag and drop fields to formula

Sold Amount (Sum)

Total Amount (Calculated F...

Unit Sold (Sum)

Formula

Example: ("Sum(Order_Count)" +
"Sum(In_Stock)" * 250

Total Amount (Calculated Field)

Format

Example: C, P, 0000 %, ###0.##0#, etc.

OK

Cancel

Create Calculated Field

Sales Amount

Drag and drop fields to formula

Sold Amount (Sum)

Total Amount (Calculated F...

Unit Sold (Sum)

Formula

"Sum(Sold)"+"Sum(Amount)"

Format

Example: C, P, 0000 %, ###0.##0#, etc.

OK

Cancel

Apply the format to the calculated field values

The values in the new or existing calculated field can be formatted through its UI and also through code behind. To format the calculated field values at runtime, the built-in textbox is available under the "Format" label where the user can set the desired format. Likewise, in code-behind, you can set the desired format using the [PivotViewFormatSettings](#) property as illustrated in the introduction section. For more information about the supported formats [refer here](#).

Create Calculated Field

×

Enter the field name

Drag and drop fields to formula

Quarter (Count)

Sold Amount (Sum)

Total Amount (Calculated F...

Formula

Example: ("Sum(Order_Count)" + "Sum(In_Stock)") * 250

Format

C2

OK

Cancel

Supported operators and functions for the calculated field formula

Below is a list of operators and functions that can be used in the formula to create the calculated fields.

- **+** – addition operator.

TYPESCRIPT

Syntax: X + Y

- **-** – subtraction operator.

TYPESCRIPT

Syntax: X - Y

- ***** – multiplication operator.

TYPESCRIPT

Syntax: X * Y

- / – division operator.

TYPESCRIPT

Syntax: X / Y

- `^` – power operator.

[Field List in Blazor Pivot Table Component](#)[Grouping Bar in Blazor Pivot Table Component](#)

<!-- markdownlint-disable MD012 -->

[Filtering in Blazor Pivot Table Component](#)

Filtering allows to view the pivot table with selective records based on the members that can be either included or excluded through UI and code-behind.

The following are the three different types of filtering:

- Member filtering
- Label filtering
- Value filtering

When all the above filtering options are disabled via code-behind, then the filter icon would be disabled in the field list or grouping bar UI.

[Member filtering](#)

<!-- markdownlint-disable MD012 -->

[Sorting in Blazor Pivot Table Component](#)[Member Sorting](#)

Allows to order field members in rows and columns either in ascending or descending order. By default, field members in rows and columns are in ascending order.

[Grouping in Blazor Pivot Table Component](#)

Grouping is the most-useful feature in pivot table and the component automatically groups date, time, number and string. For example, the date type can be formatted and displayed based on year, quarter, month, and more. Likewise, the number type can be grouped range-wise, such as 1-5, 6-10, etc. These group fields will act as individual fields and allows users to drag them between different axes such as columns, rows, values, and filters and create pivot table at runtime.

This feature is applicable only for relational data source.

<!-- markdownlint-disable MD036 -->

[Virtual Scrolling in Blazor Pivot Table Component](#)

[Defer Layout Update in Blazor Pivot Table Component](#)

<!-- markdownlint-disable MD012 -->

[Row and Column in Blazor Pivot Table Component](#)

[Width and Height](#)

[State Persistence in Blazor Pivot Table Component](#)

[Hide Totals in Blazor Pivot Table Component](#)

[Show or hide grand totals](#)

[Conditional Formatting in Blazor Pivot Table Component](#)

It allows the end user to change the appearance of the pivot table value cells with its background color, font color, font family, and font size based on the specific conditions.

[Drill Through in Blazor Pivot Table Component](#)

[Editing in Blazor Pivot Table Component](#)

This feature is applicable only for relational data source.

[Hyperlink in Blazor Pivot Table Component](#)

[Toolbar in Blazor Pivot Table Component](#)

[Tooltip in Blazor Pivot Table Component](#)

[CSS Customization in Blazor Pivot Table Component](#)

[Hiding Axis](#)

The visibility of row, column, value and filter axis in Field List and Grouping Bar can be changed using custom CSS setting.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails" ID="PivotView" ShowGroupingBar="true"
ShowFieldList="true">
  <PivotViewDataSourceSettings DataSource="@data">
    <PivotViewColumns>
      <PivotViewColumn Name="Year"></PivotViewColumn>
      <PivotViewColumn Name="Quarter"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Country" ></PivotViewRow>
      <PivotViewRow Name="Country" ></PivotViewRow>
      <PivotViewRow Name="Products"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Sold" Caption="Units Sold"></PivotViewValue>
      <PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
    </PivotViewValues>
    <PivotViewFormatSettings>
      <PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
    </PivotViewFormatSettings>
  </PivotViewDataSourceSettings>
</SfPivotView>
<style>
```

```

/* Hiding column axis in grouping bar */
#PivotView .e-group-columns {
display: none;
}
/* Increasing filter axis height to fill column axis portion */
#PivotView .e-group-filters {
min-height: 74.67px !important;
}
/* Hiding column axis in field list */
.e-pivotfieldlist-container .e-field-list-columns {
display: none;
}
/* Increasing value axis height to fill column axis portion */
.e-pivotfieldlist-container .e-field-list-values {
margin-top: 0px !important;
min-height: 338px !important;
}
.e-pivotfieldlist-container .e-values {
height: 310px !important;
}
/* Hiding row axis in grouping bar */
// #PivotView .e-group-rows {
//     display: none;
// }
/* Hiding row axis in field list */
// .e-pivotfieldlist-container .e-field-list-rows {
//     display: none;
// }
/* Hiding value axis in grouping bar */
// #PivotView .e-group-values {
//     display: none;
// }
/* Hiding value axis in field list */
// .e-pivotfieldlist-container .e-field-list-values {
//     display: none;
// }
/* Hiding filter axis in grouping bar */
// #PivotView .e-group-filters {
//     display: none;
// }
/* Hiding filter axis in field list */
//.e-pivotfieldlist-container .e-field-list-filters {
//     display: none;
// }
</style>
@code{
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
}

```

Globalization in Blazor Pivot Table Component

Add **UseRequestLocalization** middle-ware in Configure method in **Startup.cs** file to get browser Culture Info.

Refer the following code to add configuration in Startup.cs file

CSHARP

```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Localization;
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            app.UseRequestLocalization();
            ....
            ....
        }
    }
}
```

Localization

Localization library allows you to localize default text content of the pivot table. The pivot table component has static text of some features (like drop and drop region, pivot field list, etc...) that can be changed to other cultures (Arabic, Deutsch, French, etc...). The static local texts in the pivot table component can be changed to other culture by referring the Resource file. You can refer more details about localization [here](#).

The Resource file is an XML file which contains the strings(key and value pairs) that you want to translate into different language. You can also refer [Localization](#) link to know more about how to configure and use localization in the ASP.NET Core application framework.

- Add **.resx** file to [Resource](#) folder and enter the key value (Locale Keywords) in the **Name** column and the translated string in the **Value** column as follows.

ASPX-CS

<Component Name>	<Feature Name>	<Locale Key>
------------------	----------------	--------------

The following are the list of properties and its values used in the pivot table.

Name | Value (in Deutsch culture)

PivotView_AddCondition | Bedingung hinzufügen

PivotView_AddToColumn | Zur Spalte hinzufügen

PivotView_AddToFilter | Zum Filter hinzufügen

PivotView_AddToRow | Zur Zeile hinzufügen

PivotView_AddToValue | Zum Wert hinzufügen

PivotView_After | Nach

PivotView_AfterOrEqualTo | Nach oder gleich

PivotView_Aggregate | Aggregat

PivotView_Alert | Warnen

PivotView_All | Alle

PivotView_All Fields | Alle Felder

PivotView_AllValues | Alle Werte

PivotView_And | und

PivotView_Apply | ANWENDEN

PivotView_Area | Bereich

PivotView_Ascending | Aufsteigend

PivotView_Avg | durchschnittlich

PivotView_Bar | Bar

PivotView_BaseField | Basisfeld

PivotView_Baseltem | Basisgegenstand

PivotView_Before | Vor

PivotView_BeforeOrEqualTo | Vorher oder gleich

PivotView_BeginWith | Beginnt mit

PivotView_Between | Zwischen

PivotView_Blank | (Leer)

PivotView_By | durch

PivotView_CalculatedField | Berechnetes Feld

PivotViewCalculatedFieldConfirmMessage | In diesem Namen ist bereits ein Berechnungsfeld vorhanden. Möchten Sie es ersetzen?

PivotViewCalculatedFieldDragDropMessage | Ziehen Sie Felder in die Formel und legen Sie sie dort ab

PivotViewCalculatedFieldDragMessage | Ziehen Sie das Feld in die Formel

PivotViewCalculatedFieldDropMessage | Das berechnete Feld kann nur in der Werteachse in einem anderen Bereich platziert werden.

PivotViewCalculatedFieldExampleWatermark | Beispiel: ('Sum(OrderCount)' + 'Sum(InStock)') * 250

PivotViewCalculatedFieldExistMessage | In diesem Namen ist bereits ein Feld vorhanden. Bitte geben Sie einen anderen Namen ein.

PivotViewCalculatedFieldMobileWatermark | Fügen Sie hier Felder hinzu und bearbeiten Sie die Formel.

PivotViewCalculatedFieldNameWatermark | Geben Sie den Feldnamen ein

PivotViewCalculatedFieldOLAPExampleWatermark | Beispiel: [Measures].[Order Quantity] + ([Measures].[Order Quantity] * 0.10)

PivotViewCalculatedFieldTooltip | Ziehen Sie Felder per Drag & Drop, um einen Ausdruck zu erstellen. Und wenn Sie die vorhandenen die berechneten Felder bearbeiten möchten! Dann können Sie dies erreichen, indem Sie einfach das Feld unter 'Berechnete Mitglieder' auswählen.

PivotView_Cancel | Stornieren

PivotView_Chart | Diagramm

PivotView_Clear | Klar

PivotView_ClearFilter | Klar

PivotView_Close | Schließen

PivotView_Collapse | Zusammenbruch

PivotView_Column | Säule

PivotView_ColumnAxisWatermark | Spalte hier ablegen

PivotView_Columns | Säulen

PivotView_ConditionalFormating | Bedingte Formatierung

PivotView_ConditionalFormatting | Bedingte formatierung

PivotView_Contains | Enthält

PivotView_Copy | Kopieren

PivotView_Count | Anzahl

PivotView_CreateCalculatedField | Berechnetes Feld erstellen

PivotView_CSV | CSV

PivotView_Currency | Währung

PivotView_CurrencySymbol | Währungszeichen

PivotView_Custom | Benutzerdefiniert

PivotView_CustomFormat | Benutzerdefiniertes Format

PivotView_CustomFormatMessage | Geben Sie eine benutzerdefinierte Formatzeichenfolge ein

PivotView_Date | Datum

PivotView_DateTextMessage | Zeigen Sie die Elemente an, für die das Datum gilt

PivotView_Days | Tage

PivotView_DecimalPlaces | Nachkommastellen

PivotView_Delete | Löschen

PivotView_DeleteReport | Löschen Sie einen aktuellen Bericht

PivotView_Descending | Absteigend

PivotView_Details | Einzelheiten

PivotView_DifferenceFrom | Unterschied von

PivotView_Dimension | Abmessungen

PivotView_DistinctCount | Deutliche Anzahl

PivotView_DoesNotBeginWith | Beginnt nicht mit

PivotView_DoesNotContains | Beinhaltet nicht

PivotView_DoesNotEndsWith | Endet nicht mit

PivotView_DoesNotEquals | Ist nicht gleich

PivotView_DoNotShowGrandTotals | Keine Gesamtsummen anzeigen

PivotView_DoNotShowSubtotals | Zwischensummen nicht anzeigen

PivotView_Drag | Ziehen

PivotView_DrillThrough | Durchbohren

PivotView_DrillThroughErrorMessage | Die Roh Elemente berechneter Felder können nicht angezeigt werden.

PivotView_Edit | Bearbeiten

PivotView_EmptyRecordsMessage | Keine Datensätze zum Anzeigen

PivotView_EmptyReportMessage | Keine Berichte gefunden !!

PivotView_EndAt | Endet bei

PivotView_EndsWith | Endet mit

PivotView_EnterDate | Datum eingeben

PivotView_EnterReportNameMessage | Geben Sie einen Berichtsnamen ein

PivotView_EnterValue | Wert eingeben

PivotView_Equals | Gleich

PivotView_Error | Error

PivotView_Example | z.B:

PivotView_Excel | Excel

PivotView_Expand | Erweitern

PivotView_Export | Export

PivotView_Expression | Ausdruck

PivotView_False | Falsch

PivotView_FieldCaption | Feldbeschriftung

PivotView_FieldCaptionMessage | Feldbeschriftung

PivotView_FieldDropErrorMessage | Das Feld, das Sie verschieben, kann nicht in diesem Bereich des Berichts platziert werden

PivotView_FieldName | Feldname

PivotView_FieldNameMessage | Feldname :
PivotView_FieldType | Feldtyp
PivotView_Filter | Filter
PivotView_FilterAxisWatermark | Filter hier ablegen
PivotView_Filtered | Gefiltert
PivotView_Filters | Filter
PivotView_Format | Format
PivotView_FormatString | Zeichenfolge formatieren
PivotView_FormatType | Formattyp
PivotView_Formula | Formel
PivotView_GrandTotal | Gesamtsumme
PivotView_GrandTotals | Gesamtsummen
PivotView_GreaterThan | Größer als
PivotView_GreaterThanOrEqualTo | Größer als oder gleich wie
PivotView_Group | Gruppe
PivotView_GroupCaptionMessage | Geben Sie die Beschriftung ein, die in der Kopfzeile angezeigt werden soll
PivotView_GroupFieldCaptionMessage | Geben Sie die Beschriftung für das Gruppenfeld ein
PivotView_Grouping | Gruppierung
PivotView_GroupName | Gruppenname
PivotView_Hours | Std
PivotView_Index | Index
PivotView_IntervalBy | Intervall von
PivotView_InvalidFormat | Ungültiges Format.
PivotView_InvalidFormula | Ungültige Formel.
PivotView_InvalidGroupSelectionMessage | Diese Auswahl kann nicht gruppiert werden.
PivotView_JPEG | JPEG
PivotView_Label | Etiketle
PivotView_LabelTextMessage | Zeigen Sie die Elemente an, für die das Etikett
PivotView_Left | Links
PivotView_LessThan | Weniger als
PivotView_LessThanOrEqualTo | Gleich oder kleiner als
PivotView_Line | Linie

PivotView_LoadReport | Belastung
PivotView_ManageRecords | Datensätze verwalten
PivotView_Max | Max
PivotView_MdxQuery | MDX-Abfrage
PivotView_Measure | Messen
PivotView_Member | Mitglied
PivotView_MemberLimitMessage | weitere Artikel. Suche, um weiter zu verfeinern.
PivotView_Min | Min
PivotView_Minutes | Protokoll
PivotView_Months | Monate
PivotView_MoreOption | Mehr...
PivotView_MultipleItems | Mehrere Elemente
PivotView_NewReport | Erstellen Sie einen neuen Bericht
PivotView_NewReportConfirmMessage | Möchten Sie Änderungen speichern, um sie zu melden?
PivotView_NoFormatMessage | Kein Format gefunden !!!
PivotView_NoInputMessage | Geben Sie einen Wert ein
PivotView_NoMatchesMessage | Keine Treffer
PivotView_NotBetween | Nicht zwischen
PivotView_NotEquals | Nicht gleich
PivotView_NoValue | Kein Wert
PivotView_Null | Null
PivotView_Number | Nummer
PivotView_NumberFormatting | Zahlenformatierung
PivotView_Of | von
PivotView_OK | OK
PivotView_OutOfRange | Außer Reichweite
PivotView_ParentHierarchy | Übergeordnete Hierarchie
PivotView_PDF | PDF
PivotView_Percent | Prozent
PivotView_Percentage | Prozentsatz
PivotView_PercentageOfColumnTotal | % der Spalte insgesamt
PivotView_PercentageOfDifferenceFrom | % des Unterschieds von
PivotView_PercentageOfGrandTotal | % der Gesamtsumme

PivotView_PercentageOfParentColumnTotal | % der übergeordneten Spalte insgesamt
PivotView_PercentageOfParentRowTotal | % der Gesamtzahl der übergeordneten Zeilen
PivotView_PercentageOfParentTotal | % der Elternsumme
PivotView_PercentageOfRowTotal | % der Zeilensumme
PivotView_PNG | PNG
PivotView_Polar | Polar
PivotView_PopulationStDev | Bevölkerungsstandardabweichung
PivotView_PopulationVar | Populationsvarianz
PivotView_Product | Produkt
PivotView_Quarter | Qtr
PivotView_Quarters | Viertel
PivotView_QuarterYear | Vierteljahr
PivotView_Remove | Entfernen
PivotView_RemoveReportConfirmMessage | Möchten Sie diesen Bericht wirklich löschen?
PivotView_RenameReport | Benennen Sie einen aktuellen Bericht um
PivotView_ReportList | Berichtsliste
PivotView_ReportNameMessage | Berichtsname:
PivotView_Right | Recht
PivotView_Row | Reihe
PivotView_RowAxisWatermark | Hier eine Zeile ablegen
PivotView_Rows | Reihen
PivotView_RunningTotals | Laufende Summen
PivotView_SampleReport | Beispielbericht
PivotView_SampleStDev | Standardabweichung der Probe
PivotView_SampleVar | Stichprobenvarianz
PivotView_SaveAsReport | Als aktuellen Bericht speichern
PivotView_SaveReport | Speichern Sie einen Bericht
PivotView_Scatter | Streuen
PivotView_Search | Suche
PivotView_Seconds | Sekunden
PivotView_SelectedItems | Ausgewählte Artikel
PivotView_SelectGroups | Wählen Sie Gruppen aus
PivotView_ShowColumnGrandTotalsOnly | Nur Gesamtsummenspalten anzeigen

PivotView_ShowColumnSubtotalsOnly | Nur Zwischensummenspalten anzeigen

PivotView_ShowFieldList | Feldliste anzeigen

PivotView_ShowGrandTotals | Gesamtsummen anzeigen

PivotView_ShowRowGrandTotalsOnly | Nur Gesamtsummenzeilen anzeigen

PivotView_ShowRowSubtotalsOnly | Nur Zwischensummenzeilen anzeigen

PivotView_ShowSubtotals | Zwischensummen anzeigen

PivotView_ShowTable | Tabelle anzeigen

PivotView_Sort | Sortieren

PivotView_Standard | Standard

PivotView_StartAt | Beginnt um

PivotView_Subtotals | Zwischensummen

PivotView_Sum | Summe

PivotView_Summaries | Werte zusammenfassen mit

PivotView_SummarizeValuesBy | Werte zusammenfassen mit

PivotView_SVG | SVG

PivotView_SymbolPosition | Symbolposition

PivotView_Total | Gesamt

PivotView_True | Wahr

PivotView_Undefined | nicht definiert

PivotView_Ungroup | Gruppierung aufheben

PivotView_ValidReportNameMessage | Bitte geben Sie einen gültigen Datensatznamen ein !!!

PivotView_Value | Wert

PivotView_ValueAxisWatermark | Wert hier ablegen

PivotView_ValueFieldSettings | Wertefeldeinstellungen

PivotView_Values | Werte

PivotView_ValueTextMessage | Zeigen Sie die Elemente an, für die

PivotView_Warning | Warnung

PivotView_Years | Jahre

PivotView_MultipleAxes | Mehrere Achsen

PivotView_ChartTypeSettings | Configuración de tipo de gráfico

PivotView_ChartType | Tipo de carta

PivotView_Yes | si

PivotView_No | No

PivotView_NumberFormatMenu | Formato de número ...

PivotView_ConditionalFormatingMenu | formato condicional ...

PivotViewCalculatedFieldRemoveMessage | Está seguro de que desea eliminar este campo calculado?

PivotView_StackedArea | Área apilada

PivotView_StackedColumn | Columna apilada

PivotView_StackedBar | Barra apilada

PivotView_StepLine | Línea de paso

PivotView_StepArea | Área de paso

PivotView_SplineArea | Área de spline

PivotView_Spline | Ranura

PivotView_StackedColumn100 | Columna 100% apilada

PivotView_StackedBar100 | Barra 100% apilada

PivotView_StackedArea100 | Área 100% apilada

PivotView_Bubble | Burbuja

PivotView_Pareto | Pareto

PivotView_Radar | Radar

PivotViewCalculatedFieldClearTooltipMessage | Bearbeiten Sie die bearbeiteten Feldinformationen

PivotViewCalculatedFieldEditTooltipMessage | Berechnetes Feld bearbeiten

PivotViewNumberFormatExampleWatermark | Beispiel: C, P, 0000%, ### 0. ## 0 # usw.

PivotView_ShowLegend | Legende anzeigen

PivotView_FieldCaptionWatermark | Geben Sie die Feldbeschriftung ein

PivotViewSortNoneTooltipMessage | Datenreihenfolge sortieren

PivotViewSortAscendingTooltipMessage | Aufsteigende Reihenfolge sortieren

PivotViewSortDescendingTooltipMessage | Absteigende Reihenfolge sortieren

PivotViewReplaceReportBeforeMessage | Ein Bericht mit dem Namen

PivotViewReplaceReportAfterMessage | ist bereits vorhanden. Möchten Sie es ersetzen?

PivotView_StaticFieldList | Pivot-Feldlis

PivotView_FieldList | Feldliste

PivotView_AddFieldMessage | Feld hier hinzufügen

PivotView_ChooseFieldMessage | Feld auswählen

PivotView_DragFieldsMessage | Ziehen Sie Felder zwischen den folgenden Achsen:

PivotView_Add | Hinzufügen

PivotView_DeferLayoutUpdate | Layoutaktualisierung verschieben

All locale files for different cultures are available in this [GitHub](#) location. You can get any resource file from there and utilize it in your application.

Blazor Server Side

In the following example, we have demonstrate how to enable **Localization** for pivot table in server side Blazor sample.

- Open the **Startup.cs** file and add the below configuration in the **ConfigureServices** function as follows.

CSHARP

```
using Syncfusion.Blazor;
using System.Globalization;
using Microsoft.AspNetCore.Localization;
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void ConfigureServices(IServiceCollection services)
        {
            ....
            ....
            services.AddSyncfusionBlazor();
            services.AddLocalization(options => options.ResourcesPath = "Resources");
            services.Configure<RequestLocalizationOptions>(options =>
            {
                // define the list of cultures your app will support
                var supportedCultures = new List<CultureInfo>()
                {
                    new CultureInfo("de")
                };
                // set the default culture
                options.DefaultRequestCulture = new RequestCulture("de");
                options.SupportedCultures = supportedCultures;
                options.SupportedUICultures = supportedCultures;
                options.RequestCultureProviders = new List<IRequestCultureProvider>() {
                    new QueryStringRequestCultureProvider() // Here, You can also use other
                    localization provider
                };
            });
            services.AddSingleton(typeof(ISyncfusionStringLocalizer),
                typeof(SampleLocalizer));
        }
    }
}
```

Add [UseRequestLocalization\(\)](#) middle-ware in Configure method in **Startup.cs** file to get browser Culture Information.

- Then, write a **class** by inheriting **ISyncfusionStringLocalizer** interface and override the **Manager** property to get the resource file details from the application end.

CSHARP

```

using Syncfusion.Blazor;
namespace BlazorApplication
{
    public class SampleLocalizer : ISyncfusionStringLocalizer
    {
        public string Get(string key)
        {
            return this.Manager.GetString(key);
        }
        public System.Resources.ResourceManager Manager
        {
            get
            {
                return BlazorApplication.Resources.SyncfusionBlazorLocale.ResourceManager;
            }
        }
    }
}

```

ASPX-CS

```

@using Syncfusion.Blazor.PivotView;
@using Syncfusion.Blazor
<SfPivotView TValue="ProductDetails">
    <PivotViewDataSourceSettings DataSource="@data">
        <PivotViewColumns>
            <PivotViewColumn Name="Year"></PivotViewColumn>
            <PivotViewColumn Name="Quarter"></PivotViewColumn>
        </PivotViewColumns>
        <PivotViewRows>
            <PivotViewRow Name="Country"></PivotViewRow>
            <PivotViewRow Name="Products"></PivotViewRow>
        </PivotViewRows>
        <PivotViewValues>
            <PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
            <PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
        </PivotViewValues>
        <PivotViewFormatSettings>
            <PivotViewFormatSetting Name="Amount" Format="C0"
            UseGrouping=true></PivotViewFormatSetting>
        </PivotViewFormatSettings>
    </PivotViewDataSourceSettings>
</SfPivotView>
@code{
    public List<ProductDetails> data { get; set; }
    protected override void OnInitialized()
    {
        this.data = ProductDetails.GetProductData().ToList();
        //Bind the data source collection here. Refer "Assigning sample data to the
        pivot table" section in getting started for more details.
    }
}

```

	FY 2015		FY 2016		FY 2017
	Unit Sold	Sold Amount	Unit Sold	Sold Amount	Unit Sold
France	450	\$714,955	526	\$1,542,104	
Germany	440	\$563,515	496	\$1,772,104	
United States	546	\$754,515	636	\$2,263,104	
Gesamtsumme	1436	\$2,032,985	1658	\$5,577,312	1

Blazor WebAssembly

In the following examples, demonstrate how to enable **Localization** for pivot table in client side Blazor samples.

- Open the **Program.cs** file and add the below configuration in the **Main** function as follows.

CSHARP

```
using Syncfusion.Blazor;
using System.Globalization;
namespace ClientApplication
{
    public class Program
    {
        public static async Task Main(string[] args)
        {
            var builder = WebAssemblyHostBuilder.CreateDefault(args);
            builder.RootComponents.Add<App>("app");
            builder.Services.AddTransient(sp => new HttpClient { BaseAddress = new
            Uri(builder.HostEnvironment.BaseAddress) });
            builder.Services.AddSyncfusionBlazor();
            // Register the Syncfusion locale service to customize the SyncfusionBlazor
            // component locale culture
            builder.Services.AddSingleton(typeof(ISyncfusionStringLocalizer),
            typeof(SyncfusionLocalizer));
            // Set the default culture of the application
            CultureInfo.DefaultThreadCurrentCulture = new CultureInfo("de");
            CultureInfo.DefaultThreadCurrentUICulture = new CultureInfo("de");
            await builder.Build().RunAsync();
        }
    }
}
```

- Then, create a **~/Shared/SyncfusionLocalizer.cs** file and implement **ISyncfusionStringLocalizer** interface to the class and override the **ResourceManager** property to get the resource file details from the application end.

CSHARP

```
using Syncfusion.Blazor;
public class SyncfusionLocalizer : ISyncfusionStringLocalizer
{
}
```

```
// To get the locale key from mapped resources file
public string GetText(string key)
{
    return this.ResourceManager.GetString(key);
}
// To access the resource file and get the exact value for locale key
public System.Resources.ResourceManager ResourceManager
{
    get
    {
        // Replace the ApplicationNamespace with your application name.
        return ClientApplication.Resources.SfResources.ResourceManager;
    }
}
}
```

ClientApplication denotes the ApplicationNameSpace of your project.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView;
<SfPivotView TValue="ProductDetails">
<PivotViewDataSourceSettings DataSource="@data">
<PivotViewColumns>
<PivotViewColumn Name="Year"></PivotViewColumn>
<PivotViewColumn Name="Quarter"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Country"></PivotViewRow>
<PivotViewRow Name="Products"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
<PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="Amount" Format="C0"
UseGrouping=true></PivotViewFormatSetting>
</PivotViewFormatSettings>
</PivotViewDataSourceSettings>
</SfPivotView>
@code{
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
    this.data = ProductDetails.GetProductData().ToList();
    //Bind the data source collection here. Refer "Assigning sample data to the
    pivot table" section in getting started for more details.
}
}
```

	FY 2015		FY 2016		FY 2017
	Unit Sold	Sold Amount	Unit Sold	Sold Amount	Unit Sold
France	450	\$714,955	526	\$1,542,104	
Germany	440	\$563,515	496	\$1,772,104	
United States	546	\$754,515	636	\$2,263,104	
Gesamtsumme	1436	\$2,032,985	1658	\$5,577,312	1

Internationalization

Internationalization library is used to globalize number, date, and time values in pivot table component using format strings in the [Format](#). In the below sample we set the culture and currency using the [SetCulture](#) and [SetCurrencyCode](#) methods.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView;
@using Syncfusion.Blazor
@using Microsoft.JSInterop;
<SfPivotView TValue="ProductDetails">
<PivotViewDataSourceSettings DataSource="@data">
<PivotViewColumns>
<PivotViewColumn Name="Year"></PivotViewColumn>
<PivotViewColumn Name="Quarter"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Country"></PivotViewRow>
<PivotViewRow Name="Products"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
<PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="Amount" Format="C0"
UseGrouping=true></PivotViewFormatSetting>
</PivotViewFormatSettings>
</PivotViewDataSourceSettings>
</SfPivotView>
@code{
public List<PivotViewData.ProductDetails> data { get; set; }
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
[Inject]
protected IJSRuntime JsRuntime { get; set; }
protected override void OnAfterRender(bool firstRender)
{
if (firstRender)
```



```
{
    this.JsRuntime.Sf().SetCulture("de-DE").SetCurrencyCode("EUR");
}
}
```

* In the above sample, **Amount** column is formatted by [Format](#).

* Default culture is **en-US**. If you want to change the **en-US** culture to a different culture, you have to set accordingly in `SetCulture` method.

* The decimal separators of pivot table values varies based on the culture applied to the component.

	FY 2015		FY 2016		FY 2017
	Unit Sold	Sold Amount	Unit Sold	Sold Amount	Unit Sold
France	450	714.955 €	526	1.542.104 €	
Germany	440	563.515 €	496	1.772.104 €	
United States	546	754.515 €	636	2.263.104 €	
Grand Total	1436	2.032.985 €	1658	5.577.312 €	

Right-to-left (RTL)

Right-to-left (RTL) provides an option to switch the text direction and layout of the pivot table component from right to left. It improves user experiences and accessibility for users who use right-to-left languages (Arabic, Farsi, Urdu, etc...). In the below code sample `EnableRtl` property is used to enable RTL in the pivot table.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails" ShowFieldList="true" EnableRtl="true">
  <PivotViewDataSourceSettings DataSource="@data">
    <PivotViewColumns>
      <PivotViewColumn Name="Year"></PivotViewColumn>
      <PivotViewColumn Name="Quarter"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Country"></PivotViewRow>
      <PivotViewRow Name="Products"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
      <PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
    </PivotViewValues>
    <PivotViewFormatSettings>
      <PivotViewFormatSetting Name="Amount" Format="C0"
        UseGrouping=true></PivotViewFormatSetting>
    </PivotViewFormatSettings>
  </PivotViewDataSourceSettings>
</SfPivotView>
@code{
```

```

public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
    this.data = ProductDetails.GetProductData().ToList();
    //Bind the data source collection here. Refer "Assigning sample data to the
    pivot table" section in getting started for more details.
}
}

```

2017 <	FY 2016 <		FY 2015 <		
Unit Sold	Sold Amount	Unit Sold	Sold Amount	Unit Sold	
	\$1,542,104	526	\$714,955	450	France <
	\$1,772,104	496	\$563,515	440	Germany <
	\$2,263,104	636	\$754,515	546	United States <
	\$5,577,312	1658	\$2,032,985	1436	Grand Total

You can refer to our [Blazor Pivot Table](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor Pivot Table example](#) to know how to render and configure the pivot table.

PDF Export in Blazor Pivot Table Component

The PDF export allows Pivot Table data to be exported as PDF document. To enable PDF export in the pivot table, set the [AllowPdfExport](#) in [SfPivotView](#) as **true**. Once the API is set, user needs to call the [ExportToPdfAsync](#) method for exporting, on external button click.

The pivot table component can be exported to PDF format using options available in the toolbar. For more details [refer](#) here.

ASPX-CS

```

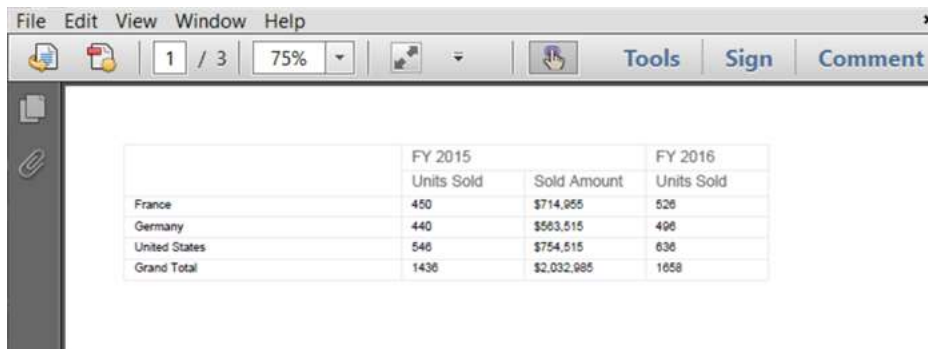
@using Syncfusion.Blazor.PivotView
@using Syncfusion.Blazor.Buttons
<SfButton OnClick="OnPdfExport" Content="Pdf Export"></SfButton>
<SfPivotView TValue="ProductDetails" @ref="@pivot" AllowPdfExport="true" >
<PivotViewDataSourceSettings DataSource="@data" EnableSorting=true>
<PivotViewColumns>
<PivotViewColumn Name="Year"></PivotViewColumn>
<PivotViewColumn Name="Quarter"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Country"></PivotViewRow>
<PivotViewRow Name="Products"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Units Sold"></PivotViewValue>
<PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="Amount" Format="C0"
UseGrouping=true></PivotViewFormatSetting>

```

```

</PivotViewFormatSettings>
</PivotViewDataSourceSettings>
</SfPivotView>
@code{
SfPivotView<ProductDetails> pivot;
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
public void OnPdfExport(Microsoft.AspNetCore.Components.Web.MouseEventArgs
args)
{
this.pivot.ExportToPdfAsync();
}
}

```



	FY 2015		FY 2016
	Units Sold	Sold Amount	Units Sold
France	450	\$714,955	526
Germany	440	\$563,515	498
United States	546	\$754,515	636
Grand Total	1436	\$2,032,985	1658

Changing the pivot table style while exporting

The PDF export provides an option to change colors for headers, caption and records in pivot table before exporting. In-order to apply colors, define **theme** settings in **pdfExportProperties** object and pass it as a parameter to the [ExportToPdfAsync](#) method.

By default, material theme will be applied to the pivot table during PDF exporting.

ASPX-CS

```

@using Syncfusion.Blazor.PivotView
@using Syncfusion.Blazor.Buttons
<SfButton OnClick="OnPdfExport" Content="Pdf Export"></SfButton>
<SfPivotView TValue="ProductDetails" @ref="@pivot" AllowPdfExport="true" >
<PivotViewDataSourceSettings DataSource="@data" EnableSorting=true>
<PivotViewColumns>
<PivotViewColumn Name="Year"></PivotViewColumn>
<PivotViewColumn Name="Quarter"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Country"></PivotViewRow>
<PivotViewRow Name="Products"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Units Sold"></PivotViewValue>
<PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>

```

```

</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="Amount" Format="C0"
UseGrouping=true></PivotViewFormatSetting>
</PivotViewFormatSettings>
</PivotViewDataSourceSettings>
</SfPivotView>
@code{
SfPivotView<ProductDetails> pivot;
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
public void OnPdfExport(Microsoft.AspNetCore.Components.Web.MouseEventArgs
args)
{
Syncfusion.Blazor.Grids.PdfExportProperties pdfExportProperties = new
Syncfusion.Blazor.Grids.PdfExportProperties()
{
Theme = new Syncfusion.Blazor.Grids.PdfTheme()
{
Header = new Syncfusion.Blazor.Grids.PdfThemeStyle()
{
FontColor = "#0fb5fc",
FontName = "Calibri",
FontSize = 15,
Bold = true,
Border = new Syncfusion.Blazor.Grids.PdfBorder() { Color = "#000000" }
},
Record = new Syncfusion.Blazor.Grids.PdfThemeStyle()
{
FontColor = "#000000",
FontName = "Segoe UI",
FontSize = 12
},
Caption = new Syncfusion.Blazor.Grids.PdfThemeStyle()
{
FontColor = "#000000",
FontName = "Segoe UI",
FontSize = 12,
}
}
};
this.pivot.ExportToPdfAsync(pdfExportProperties);
}
}

```

	FY 2015		FY 2016
	Units Sold	Sold Amount	Units Sold
France	450	\$714,955	526
Germany	440	\$563,515	496
United States	546	\$754,515	636
Grand Total	1436	\$2,032,985	1658

Changing the file name while exporting

The PDF export provides an option to change file name of the document before exporting. In-order to change the file name, define **fileName** property in **pdfExportProperties** object and pass it as a parameter to the [ExportToPdfAsync](#) method.

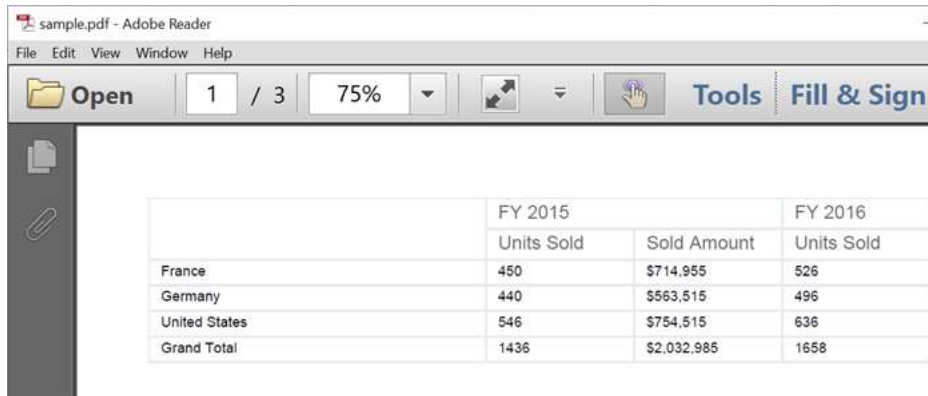
ASPX-CS

```
@using Syncfusion.Blazor.PivotView
@using Syncfusion.Blazor.Buttons
<SfButton OnClick="OnPdfExport" Content="Pdf Export"></SfButton>
<SfPivotView TValue="ProductDetails" @ref="@pivot" AllowPdfExport="true" >
  <PivotViewDataSourceSettings DataSource="@data" EnableSorting=true>
    <PivotViewColumns>
      <PivotViewColumn Name="Year"></PivotViewColumn>
      <PivotViewColumn Name="Quarter"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Country"></PivotViewRow>
      <PivotViewRow Name="Products"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Sold" Caption="Units Sold"></PivotViewValue>
      <PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
    </PivotViewValues>
    <PivotViewFormatSettings>
      <PivotViewFormatSetting Name="Amount" Format="C0"
      UseGrouping=true></PivotViewFormatSetting>
    </PivotViewFormatSettings>
  </PivotViewDataSourceSettings>
</SfPivotView>
@code{
  SfPivotView<ProductDetails> pivot;
  public List<PivotViewData.ProductDetails> data { get; set; }
  protected override void OnInitialized()
  {
    this.data = PivotViewData.GetProductData().ToList();
    //Bind your dataSource collection here, kindly refer the getting started
    section. for more information.
  }
  public void OnPdfExport (Microsoft.AspNetCore.Components.Web.MouseEventArgs
  args)
  {
```

```

Syncfusion.Blazor.Grids.PdfExportProperties pdfExportProperties = new
Syncfusion.Blazor.Grids.PdfExportProperties() { FileName = "sample.pdf" };
this.pivot.ExportToPdfAsync(pdfExportProperties);
}
}

```



Changing page size while exporting

The PDF export provides an option to change page size of the document before exporting. In-order to change the page size, define **pageSize** property in **pdfExportProperties** object and pass it as a parameter to the [ExportToPdfAsync](#) method.

Supported page sizes are: Letter, Note, Legal, A0, A1, A2, A3, A5, A6, A7, A8, A9, B0, B1, B2, B3, B4, B5, Archa, Archb, Archc, Archd,

Arche, Flsa, HalfLetter, Letter11x17, Ledger.

ASPX-CS

```

@using Syncfusion.Blazor.PivotView
@using Syncfusion.Blazor.Buttons
<SfButton OnClick="OnPdfExport" Content="Pdf Export"></SfButton>
<SfPivotView TValue="ProductDetails" @ref="@pivot" AllowPdfExport="true" >
<PivotViewDataSourceSettings DataSource="@data" EnableSorting=true>
<PivotViewColumns>
<PivotViewColumn Name="Year"></PivotViewColumn>
<PivotViewColumn Name="Quarter"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Country"></PivotViewRow>
<PivotViewRow Name="Products"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Units Sold"></PivotViewValue>
<PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="Amount" Format="C0"
UseGrouping=true></PivotViewFormatSetting>
</PivotViewFormatSettings>
</PivotViewDataSourceSettings>
</SfPivotView>
@code{
SfPivotView<ProductDetails> pivot;

```

```

public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
    this.data = ProductDetails.GetProductData().ToList();
    //Bind the data source collection here. Refer "Assigning sample data to the
    pivot table" section in getting started for more details.
}
public void OnPdfExport(Microsoft.AspNetCore.Components.Web.MouseEventArgs
args)
{
    Syncfusion.Blazor.Grids.PdfExportProperties pdfExportProperties = new
    Syncfusion.Blazor.Grids.PdfExportProperties() { PageSize =
    Syncfusion.Blazor.Grids.PdfPageSize.A3 };
    this.pivot.ExportToPdfAsync(pdfExportProperties);
}
}

```

	FY 2015		FY 2016		FY 2017	
	Units Sold	Sold Amount	Units Sold	Sold Amount	Units Sold	Sold Amount
France	450	\$714,955	526	\$1,542,104	592	\$2,903,308
Germany	440	\$563,515	496	\$1,772,104	372	\$1,634,808
United States	546	\$754,515	636	\$2,263,104	731	\$3,387,308
Grand Total	1436	\$2,032,985	1658	\$5,577,312	1695	\$7,925,424

Changing page orientation while exporting

The PDF export provides an option to change page orientation of the document before exporting. In order to change the page orientation, define **pageOrientation** property in **pdfExportProperties** object and pass it as a parameter to the [ExportToPdfAsync](#) method. By default, the page orientation will be in **Portrait** and it can be changed to **Landscape** based on user requirement.

ASPX-CS

```

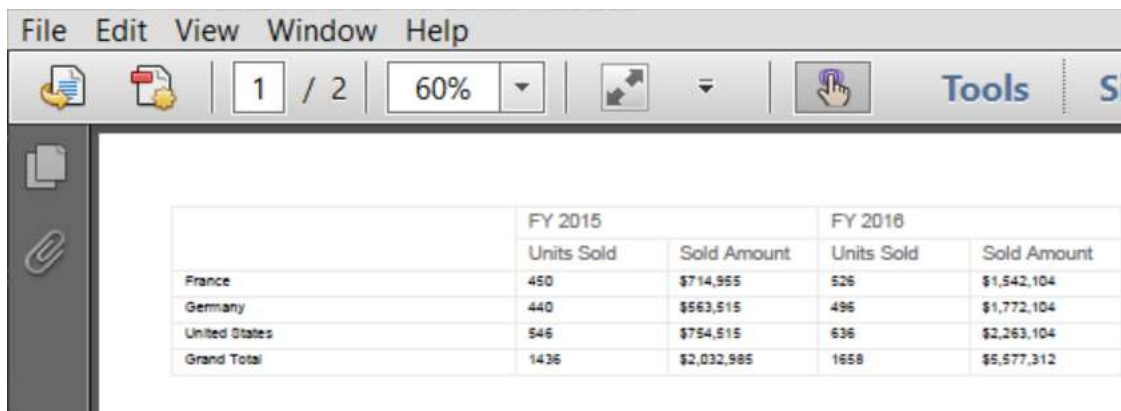
@using Syncfusion.Blazor.PivotView
@using Syncfusion.Blazor.Buttons
<SfButton OnClick="OnPdfExport" Content="Pdf Export"></SfButton>
<SfPivotView TValue="ProductDetails" @ref="@pivot" AllowPdfExport="true" >
<PivotViewDataSourceSettings DataSource="@data" EnableSorting=true>
<PivotViewColumns>
<PivotViewColumn Name="Year"></PivotViewColumn>
<PivotViewColumn Name="Quarter"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Country"></PivotViewRow>
<PivotViewRow Name="Products"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Units Sold"></PivotViewValue>
<PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="Amount" Format="C0"
UseGrouping=true></PivotViewFormatSetting>

```

```

</PivotViewFormatSettings>
</PivotViewDataSourceSettings>
</SfPivotView>
@code{
SfPivotView<ProductDetails> pivot;
public List<ProductDetails> data { get; set; }
protected override void
OnInitialized(Microsoft.AspNetCore.Components.Web.MouseEventArgs args)
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
public void OnPdfExport()
{
Syncfusion.Blazor.Grids.PdfExportProperties pdfExportProperties = new
Syncfusion.Blazor.Grids.PdfExportProperties() { PageOrientation =
Syncfusion.Blazor.Grids.PageOrientation.PageOrientation.Landscape };
this.pivot.ExportToPdfAsync(pdfExportProperties);
}
}

```



	FY 2015		FY 2016	
	Units Sold	Sold Amount	Units Sold	Sold Amount
France	450	\$714,955	526	\$1,542,104
Germany	440	\$563,515	496	\$1,772,104
United States	546	\$754,515	636	\$2,263,104
Grand Total	1436	\$2,032,985	1658	\$5,577,312

You can refer to our [Blazor Pivot Table](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor Pivot Table example](#) to know how to render and configure the pivot table.

Excel Export in Blazor Pivot Table Component

The Excel export allows Pivot Table data to be exported as Excel document. To enable Excel export in the pivot table, set the [AllowExcelExport](#) property in [SfPivotView](#) class to **true**. Once the API is set, user needs to call the [ExportToExcelAsync](#) method for exporting on external button click.

The pivot table component can be exported to Excel format using options available in the toolbar. For more details [refer](#) here.

ASPX-CS

```

@using Syncfusion.Blazor.PivotView
@using Syncfusion.Blazor.Buttons
<SfButton OnClick="OnExcelExport" Content="Excel Export"></SfButton>
<SfPivotView TValue="ProductDetails" @ref="@pivot" AllowExcelExport="true" >
<PivotViewDataSourceSettings DataSource="@data" EnableSorting=true>

```



```

<PivotViewColumns>
<PivotViewColumn Name="Year"></PivotViewColumn>
<PivotViewColumn Name="Quarter"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Country"></PivotViewRow>
<PivotViewRow Name="Products"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Units Sold"></PivotViewValue>
<PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="Amount" Format="C0"
UseGrouping=true></PivotViewFormatSetting>
</PivotViewFormatSettings>
</PivotViewDataSourceSettings>
</SfPivotView>
@code{
SfPivotView<ProductDetails> pivot;
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
public void OnExcelExport(Microsoft.AspNetCore.Components.Web.MouseEventArgs
args){
this.pivot.ExportToExcelAsync();
}
}

```

	A	B	C
1		FY 2015	
2		Units Sold	Sold Amount
3	France	450	\$714955
4	Germany	440	\$563515
5	United States	546	\$754515
6	Grand Total	1436	\$2032985

Changing the pivot table style while exporting

The Excel export provides an option to change colors for headers, caption and records in pivot table before exporting. In-order to apply colors, define **theme** settings in **excelExportProperties** object and pass it as a parameter to the [ExportToExcelAsync](#) method.

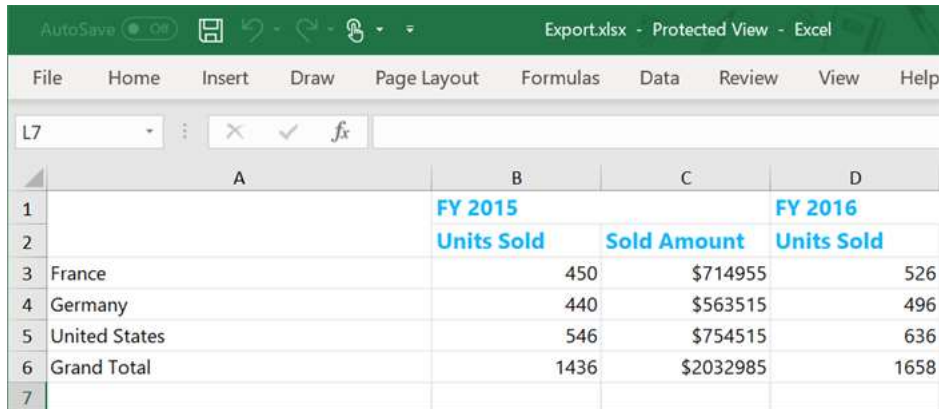
By default, material theme will be applied to the pivot table during Excel exporting.

ASPX-CS

```

@using Syncfusion.Blazor.PivotView
@using Syncfusion.Blazor.Buttons
<SfButton OnClick="OnExcelExport" Content="Excel Export"></SfButton>
<SfPivotView TValue="ProductDetails" @ref="@pivot" AllowExcelExport="true" >
<PivotViewDataSourceSettings DataSource="@data" EnableSorting=true>
<PivotViewColumns>
<PivotViewColumn Name="Year"></PivotViewColumn>
<PivotViewColumn Name="Quarter"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Country"></PivotViewRow>
<PivotViewRow Name="Products"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Units Sold"></PivotViewValue>
<PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="Amount" Format="C0"
UseGrouping=true></PivotViewFormatSetting>
</PivotViewFormatSettings>
</PivotViewDataSourceSettings>
</SfPivotView>
@code{
SfPivotView<ProductDetails> pivot;
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
public void OnExcelExport(Microsoft.AspNetCore.Components.Web.MouseEventArgs
args)
{
Syncfusion.Blazor.Grids.ExcelExportProperties excelExportProperties = new
Syncfusion.Blazor.Grids.ExcelExportProperties()
{
Theme = new Syncfusion.Blazor.Grids.ExcelTheme()
{
Header = new Syncfusion.Blazor.Grids.ExcelStyle() { FontName = "Segoe UI",
FontColor = "#0fb5fc", FontSize = 15, Bold = true },
Record = new Syncfusion.Blazor.Grids.ExcelStyle() { FontName = "Segoe UI",
FontColor = "#000000" },
Caption = new Syncfusion.Blazor.Grids.ExcelStyle() { FontName = "Segoe UI",
FontColor = "#000000" }
}
};
this.pivot.ExportToExcelAsync(excelExportProperties);
}
}

```



The screenshot shows an Excel spreadsheet titled 'Export.xlsx - Protected View - Excel'. The data is organized as follows:

	A	B	C	D
1		FY 2015		FY 2016
2		Units Sold	Sold Amount	Units Sold
3	France	450	\$714955	526
4	Germany	440	\$563515	496
5	United States	546	\$754515	636
6	Grand Total	1436	\$2032985	1658
7				

Changing the file name while exporting

The Excel export provides an option to change file name of the document before exporting. In-order to change the file name, define **fileName** property in **excelExportProperties** object and pass it as a parameter to the [ExportToExcelAsync](#) method.

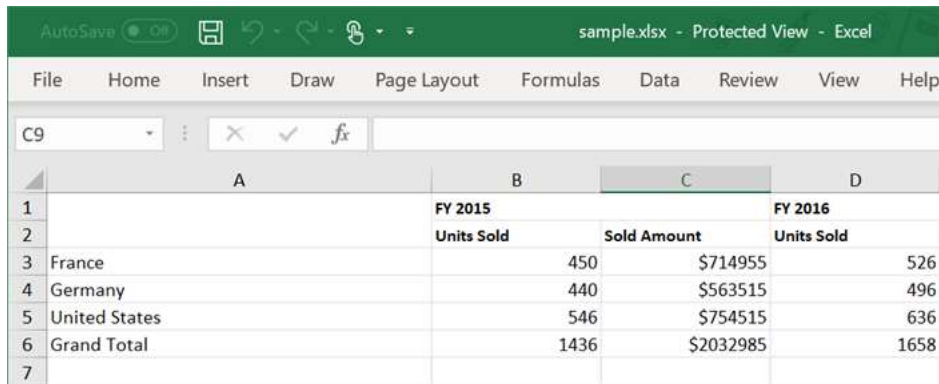
ASPX-CS

```
@using Syncfusion.Blazor.PivotView
@using Syncfusion.Blazor.Buttons
<SfButton OnClick="OnExcelExport" Content="Excel Export"></SfButton>
<SfPivotView TValue="ProductDetails" @ref="@pivot" AllowExcelExport="true" >
<PivotViewDataSourceSettings DataSource="@data" EnableSorting=true>
<PivotViewColumns>
<PivotViewColumn Name="Year"></PivotViewColumn>
<PivotViewColumn Name="Quarter"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Country"></PivotViewRow>
<PivotViewRow Name="Products"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Units Sold"></PivotViewValue>
<PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="Amount" Format="C0"
UseGrouping=true></PivotViewFormatSetting>
</PivotViewFormatSettings>
</PivotViewDataSourceSettings>
</SfPivotView>
@code{
SfPivotView<ProductDetails> pivot;
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
public void OnExcelExport(Microsoft.AspNetCore.Components.Web.MouseEventArgs
args) {
```

```

Syncfusion.Blazor.Grids.ExcelExportProperties excelExportProperties = new
Syncfusion.Blazor.Grids.ExcelExportProperties ()
{
    FileName = "sample.xlsx"
};
this.pivot.ExportToExcelAsync (excelExportProperties);
}
}

```



		FY 2015		FY 2016
		Units Sold	Sold Amount	Units Sold
3	France	450	\$714955	526
4	Germany	440	\$563515	496
5	United States	546	\$754515	636
6	Grand Total	1436	\$2032985	1658

CSV Export

The Excel export allows pivot table data to be exported in **CSV** file format as well. To enable CSV export in the pivot table, set the [AllowExcelExport](#) property in [SfPivotView](#) class as **true**. Once the API is set, user needs to call the [ExportToCsvAsync](#) method for exporting on external button click.

The pivot table component can be exported to CSV format using options available in the toolbar. For more details [refer](#) here.

ASPX-CS

```

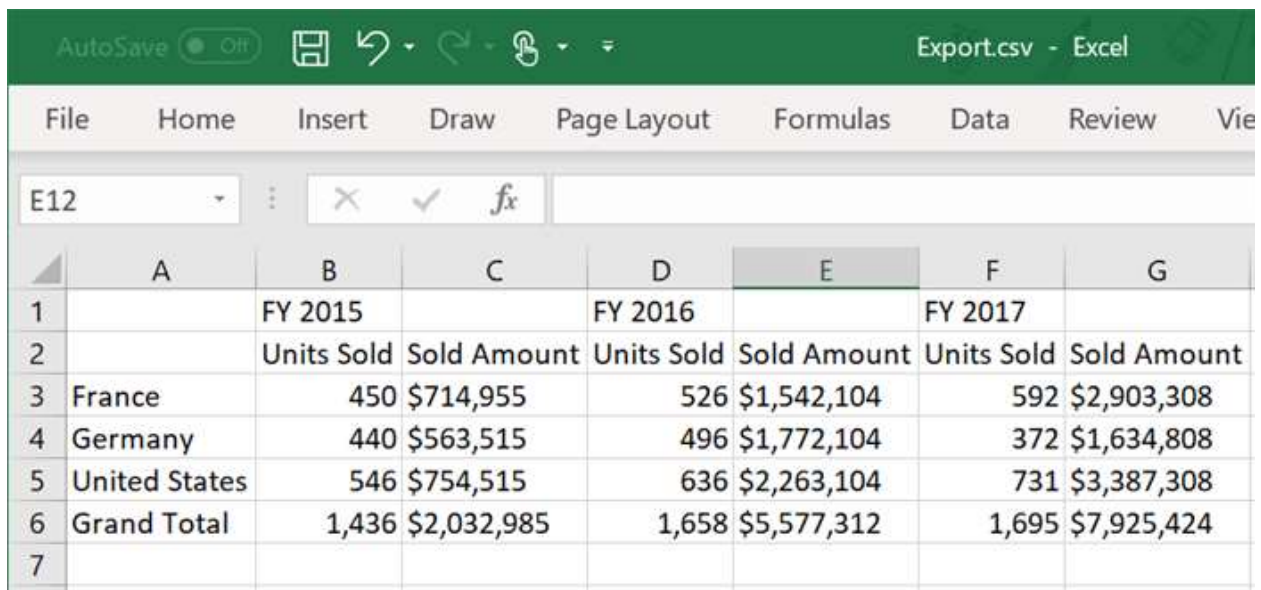
@using Syncfusion.Blazor.PivotView
@using Syncfusion.Blazor.Buttons
<SfButton OnClick="OnCsvExport" Content="Csv Export"></SfButton>
<SfPivotView TValue="ProductDetails" @ref="@pivot" AllowExcelExport="true" >
<PivotViewDataSourceSettings DataSource="@data" EnableSorting=true>
<PivotViewColumns>
<PivotViewColumn Name="Year"></PivotViewColumn>
<PivotViewColumn Name="Quarter"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Country"></PivotViewRow>
<PivotViewRow Name="Products"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Units Sold"></PivotViewValue>
<PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="Amount" Format="C0"
UseGrouping=true></PivotViewFormatSetting>
</PivotViewFormatSettings>
</PivotViewDataSourceSettings>
</SfPivotView>
@code{

```

```

SfPivotView<ProductDetails> pivot;
public List<ProductDetails> data { get; set; }
protected override void
OnInitialized(Microsoft.AspNetCore.Components.Web.MouseEventArgs args)
{
    this.data = ProductDetails.GetProductData().ToList();
    //Bind the data source collection here. Refer "Assigning sample data to the
    pivot table" section in getting started for more details.
}
public void OnCsvExport() {
    this.pivot.ExportToCsvAsync();
}
}

```



	A	B	C	D	E	F	G
1		FY 2015		FY 2016		FY 2017	
2		Units Sold	Sold Amount	Units Sold	Sold Amount	Units Sold	Sold Amount
3	France	450	\$714,955	526	\$1,542,104	592	\$2,903,308
4	Germany	440	\$563,515	496	\$1,772,104	372	\$1,634,808
5	United States	546	\$754,515	636	\$2,263,104	731	\$3,387,308
6	Grand Total	1,436	\$2,032,985	1,658	\$5,577,312	1,695	\$7,925,424
7							

You can refer to the [Blazor Pivot Table](#) feature tour page for its groundbreaking feature representations. You can also explore the [Blazor Pivot Table example](#) to know how to render and configure the pivot table.

Accessibility in Blazor Pivot Table Component

Accessibility is achieved in the pivot table component through WAI-ARIA standard and keyboard navigation. The pivot table 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.

The following ARIA attributes are used in the pivot table:

- grid (role)
- row (role)
- gridcell (role)

- aria-selected (attribute)
- aria-expanded (attribute)
- aria-sort (attribute)
- aria-busy (attribute)
- aria-invalid (attribute)
- aria-grabbed (attribute)
- aria-owns (attribute)
- aria-label (attribute)

Keyboard Navigation

All the pivot table actions can be controlled via keyboard keys. The applicable key combinations and their relative functionalities are listed below for the appropriate UI features available in the component.

Pivot Table

Interaction Keys | Description

Shift + Ctrl + F | If the popup field list is enabled in either the pivot table or the pivot chart, the field list dialog will be opened.

Tab | Moves to the next active element in the field list. If no active elements are present, it moves to the next active element in the browser page.

Shift + Tab | Moves to the previous active element in the field list. If no active elements are present, it moves to the previous active element in the browser page.

Shift + F | If the current active element is a field's button and if it has a filter icon, the filter dialog will open to perform filtering.

Shift + S | If the current active element is a field's button and if it has a sort icon, the sorting will be performed to the selected field.

Shift + E | If the current active element is a calculated field's button and if it has an edit icon, the calculated field dialog will be opened to edit the selected calculated field.

Enter | Performs the selection operation of the current active element. If the current active element is a field's button and it has a dropdown icon, the aggregation menu will open to perform calculations using aggregation options to the selected value field.

Delete | If the current active element is a field's button, the selected field will be removed from the current report.

DownArrow | If the current active element is a tree node, it moves to the next node.

UpArrow | If the current active element is a tree node, it moves to the previous node.

LeftArrow | If the current active element is a tree node, it collapses the current node.

RightArrow | If the current active element is a tree node, it expands the current node.

Home | If the current active element is a tree node, it goes to the first node.

End | If the current active element is a tree node, it goes to the last node.

Space | If the current active element is a tree node or a checkbox element, it will be either checked or unchecked.

Esc or Escape | Closes the popup field list dialog.

Grouping Bar

Interaction Keys | Description

Shift + F | If the current active element is a field's button and if it has a filter icon in either the field list or grouping bar UI, the filter dialog will be opened to perform filtering.

Tab | Moves to the next active element in the filter dialog. If no active elements present, it moves to the next active element in the browser page.

Shift + Tab | Moves to the previous active element in the filter dialog. If no active elements present, it moves to the previous active element in the browser page.

DownArrow | If the current active element is a tree node, it moves to the next node.

UpArrow | If the current active element is a tree node, it moves to the previous node.

LeftArrow | If the current active element is a tree node, it collapses the current node. If the current active element is a tab, it moves focus to the previous tab element.

RightArrow | If the current active element is a tree node, it expands the current node. If the current active element is a tab, it moves focus to the next tab element.

Home | If the current active element is a tree node, it goes to the first node.

End | If the current active element is a tree node, it goes to the last node.

Space | If the current active element is a tree node or a checkbox element, it will be either checked or unchecked.

Alt + Down | If the current active element is a DropDownList or DatePicker or DateTimePicker, the popup will be opened.

Alt + Up | If the current active element is a DropDownList or DatePicker or DateTimePicker, the popup will be closed.

Enter | Performs the selection operation of the current active element. If the current active element is a tab, the current tab element will be selected. If the current active element is a tree node, the current node will be either checked or unchecked. If the current active element is DropDownList, the focus item will be selected, and the popup list will close when it is open. Otherwise, toggles the popup list.

Esc or Escape | Closes the filter dialog.

Calculated Field Dialog

Interaction Keys | Description

Tab | Moves to the next active element in the formatting dialog. If no active elements present, it moves to the next active element in the browser page.

Shift + Tab | Moves to the previous active element in the formatting dialog. If no active elements present, it moves to the previous active element in the browser page.

DownArrow | If the current active element is a DropDownList, the next item will be selected.

UpArrow | If the current active element is a DropDownList, the previous item will be selected.

Home | If the current active element is a DropDownList, the first item will be selected.

End | If the current active element is a DropDownList, the last item will be selected.

Alt + Down | If the current active element is a DropDownList or ColorPicker, the popup will be opened.

Alt + Down | If the current active element is a DropDownList or ColorPicker, the popup will be closed.

Enter | Performs the selection operation of the current active element.

Esc or Escape | Closes the formatting dialog.

Toolbar

Interaction Keys | Description

Tab | Moves to the next active element in the drill-through dialog. If the current active element is a Grid cell, it moves the cell focus to right side. If no active elements present, then it moves to the next active element in the browser page.

Shift + Tab | Moves to the previous active element in the drill-through dialog. If the current active element is a Grid cell, it moves the cell focus to left side. If no active elements present, then it moves to the previous active element in the browser page.

DownArrow | Moves the row/cell focus downwards.

UpArrow | Moves the row/cell focus upwards.

LeftArrow | Moves the cell focus left side.

RightArrow | Moves the cell focus right side.

Home | Goes to the first cell in the current row.

End | Goes to the last cell in the current row.

Ctrl + Home | Goes to the first cell in the table.

Ctrl + End | Goes to the last cell in the table.

Enter | Performs the selection operation of the current active element.

Esc or Escape | If the cell is in selected state, then it deselects all rows/cells. If the row/cell is in edit state, it cancels the current entries in the row/cell. If the current active element is not a row/cell, it closes the drill-through dialog.

F2 | Initiate editing a row/cell in the data grid.

Insert | Adds a new row/cell in the data grid.

Delete | Removes the selected row in the data grid.

Some commonly used applicable key combinations and their relative functionalities in all dialogs are listed below.

Interaction Keys | Description

Tab | Moves to the next active element in the dialog. If either no active elements present in the dialog or an overlay is not present in the dialog, it moves to the next active element in the browser page.

Shift + Tab | Moves to the previous active element in the dialog. If either no active elements present in the dialog or an overlay is not present in the dialog, it moves to the previous active element in the browser page.

Space | If the current active element is a tree node or a checkbox element, it will be either checked or unchecked.

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 be triggered. The Enter key will not be worked, when the dialog content contains any text area with initial focus.

Esc or Escape | Closes the dialog.

You can refer to the [Blazor Pivot Table](#) feature tour page for its groundbreaking feature representations. You can also explore the [Blazor Pivot Table example](#) to know how to render and configure the pivot table.

Events in Blazor Pivot Table Component

AggregateMenuOpen

To know more about this event, refer [here](#).

BeforeExport

To know more about this event, refer [here](#).

BeginDrillThrough

The event [BeginDrillThrough](#) is triggered when the value cell is clicked in the Pivot Table while editing. To use this event, user need to enable editing option with the help of [PivotViewCellEditSettings](#) class. It has following parameters - [CellInfo](#) and [Type](#). This event allows user to view the [CellInfo](#) which contains [ColumnHeaders](#), [CurrentCell](#), [CurrentTarget](#), [RawData](#), [RowHeaders](#) and [Value](#).

In this event, the parameter [GridObj](#) is returned as **null** due to its size.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
  <PivotViewDataSourceSettings DataSource="@data">
    <PivotViewColumns>
      <PivotViewColumn Name="Year"></PivotViewColumn>
      <PivotViewColumn Name="Quarter"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Country"></PivotViewRow>
```

```

<PivotViewRow Name="Products"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
<PivotViewValue Name="Amount" Caption="Amount"></PivotViewValue>
</PivotViewValues>
</PivotViewDataSourceSettings>
<PivotViewEvents TValue="ProductDetails"
BeginDrillThrough="beginDrill"></PivotViewEvents>
<PivotViewCellEditSettings AllowEditing=true AllowAdding=true
AllowDeleting=true Mode=EditMode.Normal></PivotViewCellEditSettings>
</SfPivotView>
@code{
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
private void beginDrill(BeginDrillThroughEventArgs args)
{
//args.CellInfo -> consists of cell info for drillthrough
}
}

```

CalculatedFieldCreate

To know more about this event, refer [here](#).

CellClick

The event [CellClick](#) is triggered whenever a cell is clicked in the Pivot Table component. It has following parameters - [CurrentCell](#) and [Data](#). For instance, using this event end user can either add or remove CSS class name from the respective cell and also perform many other DOM manipulations.

ASPX-CS

```

@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
<PivotViewDataSourceSettings DataSource="@data">
<PivotViewColumns>
<PivotViewColumn Name="Year"></PivotViewColumn>
<PivotViewColumn Name="Quarter"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Country"></PivotViewRow>
<PivotViewRow Name="Products"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
<PivotViewValue Name="Amount" Caption="Amount"></PivotViewValue>
</PivotViewValues>
</PivotViewDataSourceSettings>
<PivotViewEvents TValue="ProductDetails"
CellClick="cellClick"></PivotViewEvents>
</SfPivotView>
@code{

```

```

public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
    this.data = ProductDetails.GetProductData().ToList();
    //Bind the data source collection here. Refer "Assigning sample data to the
    pivot table" section in getting started for more details.
}
private void cellClick(CellClickEventArgs args)
{
    args.CurrentCell.AddClass(new string[] { "e-test" });
}
}

```

CellSelected

To know more about this event, refer [here](#).

CellSelecting

The event [CellSelecting](#) is triggered when cell selection is about to initiate in the Pivot Table. To use this event, [AllowSelection](#) property in [PivotViewGridSettings](#) must be set to **true**. It has the following parameters - [Cancel](#), [CurrentCell](#), [IsCellClick](#) and [Data](#). For instance, using this event, user can pass those selected cell information to any external component for data binding.

ASPX-CS

```

@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
<PivotViewDataSourceSettings DataSource="@data">
<PivotViewColumns>
<PivotViewColumn Name="Year"></PivotViewColumn>
<PivotViewColumn Name="Quarter"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Country"></PivotViewRow>
<PivotViewRow Name="Products"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
<PivotViewValue Name="Amount" Caption="Amount"></PivotViewValue>
</PivotViewValues>
</PivotViewDataSourceSettings>
<PivotViewEvents TValue="ProductDetails"
CellSelecting="cellSelecting"></PivotViewEvents>
</SfPivotView>
@code{
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
    this.data = ProductDetails.GetProductData().ToList();
    //Bind the data source collection here. Refer "Assigning sample data to the
    pivot table" section in getting started for more details.
}
private void cellSelecting(PivotCellSelectedEventArgs args)
{
    args.CurrentCell.AddClass(new string[] { "e-test" });
}
}

```

ChartSeriesCreated

The event [ChartSeriesCreated](#) is triggered once chart series are completely rendered. This event is triggered only when [View](#) property in [PivotViewDisplayOption](#) class is set to **Chart**. It has following parameter - [Cancel](#), and [Series](#). When the parameter [Cancel](#) is set to **true**, chart series rendered will be revoked.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
  <PivotViewDataSourceSettings DataSource="@data">
    <PivotViewColumns>
      <PivotViewColumn Name="Year"></PivotViewColumn>
      <PivotViewColumn Name="Quarter"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Country"></PivotViewRow>
      <PivotViewRow Name="Products"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
      <PivotViewValue Name="Amount" Caption="Amount"></PivotViewValue>
    </PivotViewValues>
  </PivotViewDataSourceSettings>
  <PivotViewDisplayOption View=View.Chart></PivotViewDisplayOption>
  <PivotViewEvents TValue="ProductDetails"
    ChartSeriesCreated="chartSeries"></PivotViewEvents>
</SfPivotView>

@code{
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
  this.data = ProductDetails.GetProductData().ToList();
  //Bind the data source collection here. Refer "Assigning sample data to the
  pivot table" section in getting started for more details.
}
private void chartSeries(ChartSeriesCreatedEventArgs args)
{
  args.Cancel = true;//chart series will not be rendered
}
}
```

ConditionalFormatting

The event [ConditionalFormatting](#) is triggered initially while clicking the "ADD CONDITION" button inside the conditional formatting dialog in-order to fill user specific condition instead of default condition at runtime. To use this event, [AllowConditionalFormatting](#) property in [SfPivotView](#) class must be set to **true**. It has following parameters - [ApplyGrandTotals](#), [Conditions](#), [Label](#), [Measure](#), [Style](#), [Value1](#) and [Value2](#).

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
@using Syncfusion.Blazor.Buttons
```

```

<SfButton CssClass="apply-button" IsPrimary="true" OnClick="@OnClick">Apply
Format</SfButton>
<SfPivotView TValue="ProductDetails" @ref="@Pivot"
AllowConditionalFormatting="true">
<PivotViewDataSourceSettings DataSource="@data">
<PivotViewColumns>
<PivotViewColumn Name="Year"></PivotViewColumn>
<PivotViewColumn Name="Quarter"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Country"></PivotViewRow>
<PivotViewRow Name="Products"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
<PivotViewValue Name="Amount" Caption="Amount"></PivotViewValue>
</PivotViewValues>
</PivotViewDataSourceSettings>
<PivotViewEvents TValue="ProductDetails"
ConditionalFormatting="conditionalFormat"></PivotViewEvents>
</SfPivotView>
@code{
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
SfPivotView<ProductDetails> Pivot;
public async Task OnClick(Microsoft.AspNetCore.Components.Web.MouseEventArgs
args)
{
await this.Pivot.ShowConditionalFormattingDialogAsync();
}
private void conditionalFormat(ConditionalFormatSettings args)
{
//to change the conditional formatting settings in conditional format dialog
args.Style.BackgroundColor = "Blue";
args.Value1 = 23459;
}
}

```

Drill

The event [Drill](#) is triggered whenever a member is expanded or collapsed in the Pivot Table. It has following parameters - [DrillInfo](#) and [PivotView](#). For instance using this event, user can alter delimiter and drill action for the respective item.

In this event, the parameter [PivotView](#) is returned as **null** due to its size.

ASPX-CS

```

@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails" AllowConditionalFormatting="true">
<PivotViewDataSourceSettings DataSource="@data">
<PivotViewColumns>

```

```

<PivotViewColumn Name="Year"></PivotViewColumn>
<PivotViewColumn Name="Quarter"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Country"></PivotViewRow>
<PivotViewRow Name="Products"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
<PivotViewValue Name="Amount" Caption="Amount"></PivotViewValue>
</PivotViewValues>
</PivotViewDataSourceSettings>
<PivotViewEvents TValue="ProductDetails" Drill="drill"></PivotViewEvents>
</SfPivotView>
@code{
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
private void drill(DrillArgs<ProductDetails> args)
{
//args.DrillInfo --> Here you can get drilled cell information.
}
}

```

DrillThrough

The event [DrillThrough](#) is triggered when a value cell is clicked in the Pivot Table during drill through operation. It has following parameter - [ColumnHeaders](#), [CurrentCell](#), [CurrentTarget](#), [RawData](#), [RowHeaders](#) and [Value](#). This event allows user to view cell information like [ColumnHeaders](#), [CurrentCell](#), [CurrentTarget](#), [RawData](#), [RowHeaders](#) and [Value](#) for the appropriate cell in which drill through is performed. Exclusively the event helps to view and process the raw data information behind a aggregated value inside value cell.

ASPX-CS

```

@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails" AllowConditionalFormatting="true"
AllowDrillThrough="true">
<PivotViewDataSourceSettings DataSource="@data">
<PivotViewColumns>
<PivotViewColumn Name="Year"></PivotViewColumn>
<PivotViewColumn Name="Quarter"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Country"></PivotViewRow>
<PivotViewRow Name="Products"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
<PivotViewValue Name="Amount" Caption="Amount"></PivotViewValue>
</PivotViewValues>
</PivotViewDataSourceSettings>

```

```

<PivotViewEvents TValue="ProductDetails"
DrillThrough="drillThrough"></PivotViewEvents>
</SfPivotView>
@code{
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
private void drillThrough(DrillThroughEventArgs args)
{
//args --> Here you can get the information of the clicked cell.
}
}

```

EnginePopulating

The [EnginePopulating](#) event is available in both Pivot Table and Field List.

- The event [EnginePopulating](#) is triggered from Field List object whenever the report gets modified in its UI.
- Likewise, [EnginePopulating](#) event is triggered from Pivot Table object whenever the report gets modified via grouping bar, toolbar, etc.

This event will be triggered before engine framing works gets initiated and allows user to customize the pivot datasource settings. It has the following parameter - [DataSourceSettings](#).

ASPX-CS

```

@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails" AllowConditionalFormatting="true">
<PivotViewDataSourceSettings DataSource="@data">
<PivotViewColumns>
<PivotViewColumn Name="Year"></PivotViewColumn>
<PivotViewColumn Name="Quarter"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Country"></PivotViewRow>
<PivotViewRow Name="Products"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Unit Sold"></PivotViewValue>
<PivotViewValue Name="Amount" Caption="Amount"></PivotViewValue>
</PivotViewValues>
</PivotViewDataSourceSettings>
<PivotViewEvents TValue="ProductDetails"
EnginePopulating="enginePopulating"></PivotViewEvents>
</SfPivotView>
@code{
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
}
}

```

```
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
private void enginePopulating(EnginePopulatingEventArgs args)
{
//args.DataSourceSettings --> User can modify the report before engine
populates.
}
}
```

EnginePopulated

The [EnginePopulated](#) event is available in both Pivot Table and Field List. To know more about this event, refer [here](#).

FetchReport

To know more about this event, refer [here](#).

FieldListRefreshed

To know more about this event, refer [here](#).

FieldDragStart

To know more about this event, refer [here](#).

FieldDrop

To know more about this event, refer [here](#).

FieldDropped

The [FieldDropped](#) event is available in both Grouping Bar and Field List.

To know more about this event with respect to field list operation, refer [here](#).

To know more about this event with respect to grouping bar operation, refer [here](#).

FieldRemove

To know more about this event, refer [here](#).

HyperlinkCellClicked

To know more about this event, refer [here](#).

LoadReport

To know more about this event, refer [here](#).

MemberEditorOpen

To know more about this event, refer [here](#).

NewReport

To know more about this event, refer [here](#).

OnLoad

The [OnLoad](#) event is available in both Pivot Table and Field List.

To know more about this event, refer [here](#).

RenameReport

To know more about this event, refer [here](#).

RemoveReport

To know more about this event, refer [here](#).

SaveReport

To know more about this event, refer [here](#).

ToolbarRender

The event [ToolbarRender](#) is triggered before rendering of toolbar. This event is available only when toolbar is enabled in the Pivot Table. It has following parameter - [CustomToolbar](#). Using this event user can add custom toolbar items as well as remove existing items from the toolbar.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails" ShowFieldList="true" ShowToolbar="true"
Toolbar="@toolbar" AllowConditionalFormatting="true" AllowPdfExport="true"
AllowExcelExport="true">
  <PivotViewDisplayOption Primary=Primary.Table
View=View.Both></PivotViewDisplayOption>
  <PivotViewDataSourceSettings DataSource="@data" ShowGrandTotals="true"
ShowSubTotals="true">
    <PivotViewColumns>
      <PivotViewColumn Name="Year"></PivotViewColumn>
      <PivotViewColumn Name="Quarter"></PivotViewColumn>
    </PivotViewColumns>
    <PivotViewRows>
      <PivotViewRow Name="Country"></PivotViewRow>
      <PivotViewRow Name="Products"></PivotViewRow>
    </PivotViewRows>
    <PivotViewValues>
      <PivotViewValue Name="Sold" Caption="Units Sold"></PivotViewValue>
      <PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
    </PivotViewValues>
    <PivotViewFormatSettings>
      <PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
    </PivotViewFormatSettings>
  </PivotViewDataSourceSettings>
  <PivotViewEvents TValue="ProductDetails" ToolbarRendered="toolbarRender"
RenameReport="renameReport" RemoveReport="removeReport"
SaveReport="saveReport" LoadReport="loadReport" FetchReport="fetchReport"
></PivotViewEvents>
  <PivotViewGridSettings ColumnWidth="140"></PivotViewGridSettings>
</SfPivotView>
@code{
public List<ToolbarItems> toolbar = new List<ToolbarItems> {
  ToolbarItems.New,
  ToolbarItems.Load,
  ToolbarItems.Remove,
  ToolbarItems.Rename,
  ToolbarItems.SaveAs,
  ToolbarItems.Save,
  ToolbarItems.Grid,
  ToolbarItems.Chart,
  ToolbarItems.Export,
  ToolbarItems.SubTotal,
  ToolbarItems.GrandTotal,
  ToolbarItems.ConditionalFormatting,
```

```

ToolbarItems.FieldList
};
SfPivotView<ProductDetails> pivot;
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
    this.data = ProductDetails.GetProductData().ToList();
    //Bind the data source collection here. Refer "Assigning sample data to the
    pivot table" section in getting started for more details.
}
public void toolbarRender(ToolbarArgs args)
{
    //args.CustomToolbar -> Can add or remove toolbar items.
}
public List<string> report = new List<string>();
public List<string> reportName = new List<string>();
//to save report
public void saveReport(SaveReportArgs args)
{
    var i = 0;
    bool isSaved = false;
    for (i = 0; i < this.reportName.Count; i++)
    {
        if (this.reportName[i] == args.ReportName)
        {
            this.report[i] = args.Report;
            isSaved = true;
        }
    }
    if (args.Report != null && !(isSaved))
    {
        this.report.Add(args.Report);
        this.reportName.Add(args.ReportName);
    }
}
//fetch reports
public void fetchReport(FetchReportArgs args)
{
    args.ReportName = this.reportName.ToArray();
}
//to load the selected report
public void loadReport(LoadReportArgs args)
{
    var i = 0;
    var j = 0;
    for (i = 0; i < this.reportName.Count; i++)
    {
        if (this.reportName[i] == args.ReportName)
        {
            j = i;
        }
    }
    this.pivot.LoadPersistDataAsync(this.report[j]);
}
//to delete a report
public void removeReport(RemoveReportArgs args)
{

```

```

var i = 0;
for( i=0;i<this.reportName.Count; i++)
{
    if(this.reportName[i] == args.ReportName)
    {
        this.reportName.RemoveAt(i);
        this.report.RemoveAt(i);
    }
}
// to rename a report
public void renameReport(RenameReportArgs args)
{
    var i = 0;
    for( i=0;i<=(this.reportName.Count - 1); i++)
    {
        if(this.reportName[i] == args.ReportName)
        {
            this.reportName.RemoveAt(i);
            this.reportName.Add(args.Rename);
        }
    }
}
}

```

You can refer to the [Blazor Pivot Table](#) feature tour page for its groundbreaking feature representations. You can also explore the [Blazor Pivot Table example](#) to know how to render and configure the pivot table.

How To

<!-- markdownlint-disable MD012 -->

<!-- markdownlint-disable MD009 -->

Switching to older themes style in Blazor Pivot Table Component

From Volume 1, 2020 onwards Syncfusion has revised the theming and layout of the Pivot Table. So, to inherit the older theme style and layout, please do the necessary changes in CSS and pivot table height.

CSS Selectors

In current theme, the cells can be differentiated by their background colors. To avoid it, you need to override its background colors via simple CSS coding within your application. The below CSS selectors allow to achieve the same for material, fabric, bootstrap and bootstrap v4 themes.

HTML

```

<!-- Codes here... -->
<style>
.e-pivotview .e-rowsheader,
.e-pivotview .e-columnsheader,
.e-pivotview .e-gtot,
.e-pivotview .e-content,
.e-pivotview .e-gridheader,
.e-pivotview .e-headercell {
background-color:#fff !important;
}
</style>

```

Meanwhile for high contrast theme, set the following CSS.

HTML

```
<!-- Codes here... -->
<style>
.e-pivotview .e-rowsheader,
.e-pivotview .e-columnsheader,
.e-pivotview .e-gtot,
.e-pivotview .e-content,
.e-pivotview .e-gridheader,
.e-pivotview .e-headercell {
background-color:#000 !important;
}
</style>
```

Adjusting Row Height

In current theme, to make the component compact the height of each pivot table rows has been reduced. But user can reset the height of the pivot table using the [RowHeight](#) property in [PivotViewGridSettings](#). In older theme, the property was set to 36 pixels for desktop layout and 48 pixels for mobile layout. So reset the [RowHeight](#) accordingly to visualize the older theme style.

ASPX-CS

```
@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
<PivotViewDataSourceSettings DataSource="@data">
<PivotViewColumns>
<PivotViewColumn Name="Year"></PivotViewColumn>
<PivotViewColumn Name="Quarter"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Country"></PivotViewRow>
<PivotViewRow Name="Products"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Units Sold"></PivotViewValue>
<PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
</PivotViewFormatSettings>
</PivotViewDataSourceSettings>
<PivotViewGridSettings RowHeight=36></PivotViewGridSettings>
</SfPivotView>
@code{
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
}
```

```

<style>
.e-pivotview .e-rowsheader,
.e-pivotview .e-columnsheader,
.e-pivotview .e-gtot,
.e-pivotview .e-content,
.e-pivotview .e-gridheader,
.e-pivotview .e-headercell {
background-color:#fff !important;
}
</style>

```

	FY 2015		FY 2016	
	Units Sold	Sold Amount	Units Sold	Sold Amount
France	450	\$714,955	526	\$1,5
Germany	440	\$563,515	496	\$1,7
United States	546	\$754,515	636	\$2,2
Grand Total	1436	\$2,032,985	1658	\$5,5

You can refer to [Blazor Pivot Table](#) feature tour page for its groundbreaking feature representations. You can also explore [Blazor Pivot Table example](#) to know how to render and configure the pivot table.

Customize number and date format in Blazor Pivot Table Component

User can format the number, date, and time values for each field using [PivotViewFormatSettings](#) option under [PivotViewDataSource](#). It can be configured through code behind, during initial rendering.

Number formatting

For numbers, the formatting settings required to apply through code behind are:

- [Name](#): It allows to set the field name.
- [Format](#): It allows to set the format of the respective field.

ASPX-CS

```

@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
<PivotViewDataSourceSettings DataSource="@data">
<PivotViewColumns>
<PivotViewColumn Name="Year"></PivotViewColumn>
<PivotViewColumn Name="Quarter"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Country"></PivotViewRow>
<PivotViewRow Name="Products"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Units Sold"></PivotViewValue>
<PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>

```

```

</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="Sold" Format="N"></PivotViewFormatSetting>
<PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
</PivotViewFormatSettings>
</PivotViewDataSourceSettings>
</SfPivotView>
@code{
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
}

```

	FY 2015		FY 2016		FY 2017
	Units Sold	Sold Amount	Units Sold	Sold Amount	Units Sold
France	450	\$714,955.00	526	\$1,542,104.00	
Germany	440	\$563,515.00	496	\$1,772,104.00	
United States	546	\$754,515.00	636	\$2,263,104.00	
Grand Total	1,436	\$2,032,985.00	1,658	\$5,577,312.00	1,770

Date and time formatting

This property is applicable only for relational data source.

For date and time, the formatting settings required to apply through code behind are:

- **Name:** It allows to set the field name.
- **Format:** It allows to set the format of the respective field.
- **Type:** It allows to set the type of format to be used for the respective field.

ASPX-CS

```

@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails" Width="800" Height="300">
<PivotViewDataSourceSettings DataSource="@data">
<PivotViewColumns>
<PivotViewColumn Name="Id" Caption="ID"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Date" Caption="Date"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="ProCost" Caption="Amount"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="Date" Type="FormatType.DateTime"
Format="dd/MM/yyyy-hh:mm a"></PivotViewFormatSetting>

```

```

</PivotViewFormatSettings>
</PivotViewDataSourceSettings>
</SfPivotView>
@code{
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
}
}

```

	1001	1002	1003	1004	1005
02/01/2019-12:00 AM	1	3	1	1	
03/01/2019-12:00 AM	2	2	2	4	
04/01/2019-12:00 AM		6	9	6	
05/01/2019-12:00 AM	12		4	16	
06/01/2019-12:00 AM	5				
07/01/2019-12:00 AM		12	18		
Grand Total	20	23	34	27	

Limitations of date formatting

As per Firefox and Edge browsers standards, most of the date and time formats used in data source aren't supported. For example: Apr-2000, Apr-01-2000, 01-03-2000, 2000-Apr-01 etc. are not supported. Meanwhile [ISO formats](#) will be supported across all browsers.

You can refer to our [Blazor Pivot Table](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor Pivot Table example](#) to know how to render and configure the pivot table.

Customizing loading indicator in Blazor Pivot Table Component

You can customize the appearance of the loading indicator in the pivot table by using the [SpinnerTemplate](#) property. This property accepts an HTML string which can be used for appearance customization.

You can also disable the loading indicator by setting [SpinnerTemplate](#) to empty string.

ASPX-CS

```

@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails" SpinnerTemplate="<i class='fa fa-cog fa-spin fa-3x fa-fw'></i>">
<PivotViewDataSourceSettings DataSource="@data">
<PivotViewColumns>
<PivotViewColumn Name="Year"></PivotViewColumn>
<PivotViewColumn Name="Quarter"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Country"></PivotViewRow>
<PivotViewRow Name="Products"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Units Sold"></PivotViewValue>

```

```

<PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="Sold" Format="N"></PivotViewFormatSetting>
<PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
</PivotViewFormatSettings>
</PivotViewDataSourceSettings>
</SfPivotView>
@code{
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
this.data = ProductDetails.GetProductData().ToList();
//Bind the data source collection here. Refer "Assigning sample data to the
pivot table" section in getting started for more details.
}
}

```

You can refer to [Blazor Pivot Table](#) feature tour page for its groundbreaking feature representations. You can also explore [Blazor Pivot Table example](#) to know how to render and configure the pivot table.

Hide empty headers in Blazor Pivot Table Component

If the raw data for a particular field is not defined, it will be shown as 'Undefined' in the pivot table headers. You can hide those headers by setting the [ShowHeaderWhenEmpty](#) property to **false** in the pivot table.

For example, if the raw data for the field 'Country' is defined as “United Kingdom” and “State” is not defined means, it will be shown as “United Kingdom >> Undefined” in the header section. Here, you can hide those 'Undefined' header using the [ShowHeaderWhenEmpty](#) property.

By default, this property is set as **true**.

ASPX-CS

```

@using Syncfusion.Blazor.PivotView
<SfPivotView TValue="ProductDetails">
<PivotViewDataSourceSettings DataSource="@data" ShowHeaderWhenEmpty=false>
<PivotViewColumns>
<PivotViewColumn Name="Year"></PivotViewColumn>
<PivotViewColumn Name="Quarter"></PivotViewColumn>
</PivotViewColumns>
<PivotViewRows>
<PivotViewRow Name="Country"></PivotViewRow>
<PivotViewRow Name="Products"></PivotViewRow>
</PivotViewRows>
<PivotViewValues>
<PivotViewValue Name="Sold" Caption="Units Sold"></PivotViewValue>
<PivotViewValue Name="Amount" Caption="Sold Amount"></PivotViewValue>
</PivotViewValues>
<PivotViewFormatSettings>
<PivotViewFormatSetting Name="Sold" Format="N"></PivotViewFormatSetting>
<PivotViewFormatSetting Name="Amount" Format="C"></PivotViewFormatSetting>
</PivotViewFormatSettings>
</PivotViewDataSourceSettings>
</SfPivotView>
@code{

```



```
public List<ProductDetails> data { get; set; }
protected override void OnInitialized()
{
    this.data = ProductDetails.GetProductData().ToList();
    //Bind the data source collection here. Refer "Assigning sample data to the
    pivot table" section in getting started for more details.
}
}
```

You can refer to [Blazor Pivot Table](#) feature tour page for its groundbreaking feature representations. You can also explore [Blazor Pivot Table example](#) to know how to render and configure the pivot table.

ProgressBar

Blazor ProgressBar Component in Server Side App using Visual Studio

This section briefly explains how to include a Progress Bar component in the Blazor server-side application. Refer to [Getting Started with Syncfusion Blazor for server-side in Visual Studio](#) page for introduction and configuring common specifications.

Importing Syncfusion Blazor Progress Bar component in the application

1. Install **Syncfusion.Blazor** NuGet package to the application by using the **NuGet Package Manager**.
2. Add the client-side resources using through [CDN](#) or from [NuGet](#) package in the **HEAD** element of the `~/Pages/_Host.cshtml` page.

ASPX-CS

```
<head>
<link href="_content/Syncfusion.Blazor.Themes/bootstrap4.css"
rel="stylesheet" />
<!--CDN-->
@*<link href="https://cdn.syncfusion.com/blazor/{ site.blazorversion
}}/styles/bootstrap4.css" rel="stylesheet" />*@
</head>
```

For Internet Explorer 11, kindly refer to the polyfills. Refer to the [documentation](#) for more information.

ASPX-CS

```
<head>
<link href="https://cdn.syncfusion.com/blazor/{ site.blazorversion
}}/styles/bootstrap4.css" rel="stylesheet" />
<script
src="https://github.com/Daddoon/Blazor.Polyfill/releases/download/3.0.1/blaz
or.polyfill.min.js"></script>
</head>
```

Adding component package to the application

Open the `~/_Imports.razor` file and include the **Syncfusion.Blazor.ProgressBar** namespace.

ASPX-CS

```
@using Syncfusion.Blazor.ProgressBar
```

Adding SyncfusionBlazor service in the Startup.cs

Open the **Startup.cs** file and add services required by Syncfusion components using the **services.AddSyncfusionBlazor()** method. Add this method in the **ConfigureServices** function.

CSHARP

```
using Syncfusion.Blazor;
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void ConfigureServices(IServiceCollection services)
        {
            ....
            ....
            services.AddSyncfusionBlazor();
        }
    }
}
```

To enable the custom client-side source loading from CRG or CDN, please refer to the section about [custom resources in Blazor application](#).

Adding Progress Bar component

To initialize the Progress Bar component, add the following code to the **Index.razor** view page under the **~/Pages** folder. In a new application, if the **Index.razor** page has any default content template, then those content can be completely removed and following code can be added.

ASPX-CS

```
@page "/"
<SfProgressBar Value="50" Minimum="0" Maximum="100" TrackThickness="12"
ProgressThickness="12">
</SfProgressBar>
```

On successful compilation of the application, the Syncfusion Blazor Progress Bar component will render in the web browser.



Progress Type

Change the type of the Progress Bar by using the **Type** property. By default, the **Linear** type of Progress Bar will be rendered. In the following example, view the **Circular** type.

ASPX-CS

```
<SfProgressBar Type="ProgressType.Circular" Value="70" Minimum="0"
Maximum="100" TrackThickness="8" ProgressThickness="8">
```

```
</SfProgressBar>
```



Types in Blazor ProgressBar Component

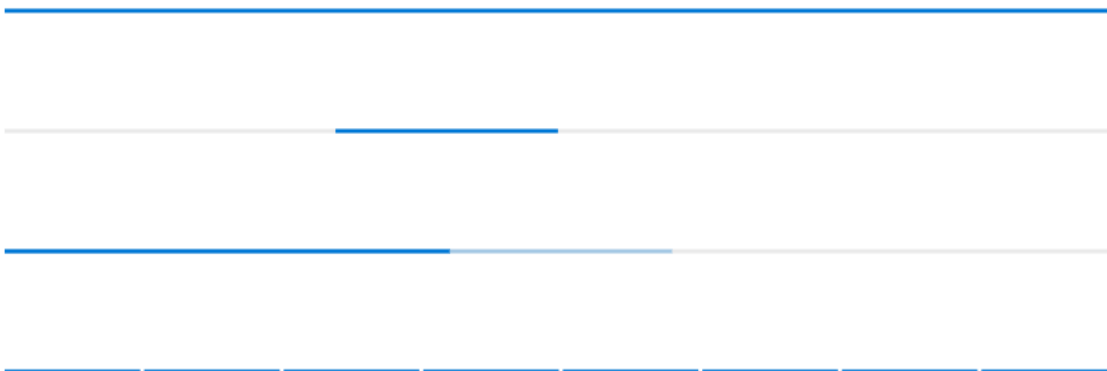
In this section, the progress can be visualized in different shapes, such as a rectangle, circle, or semi-circle to give it a unique look.

Linear

To get a linear progress bar, set the [Type](#) property to [Linear](#). As shown in the following, it also supports secondary progress, indeterminate, segments, and different modes of progress.

ASPX-CS

```
@using Syncfusion.Blazor.ProgressBar
<SfProgressBar Type="ProgressType.Linear" Value="100" Height="60"
Minimum="0" Maximum="100">
</SfProgressBar>
<SfProgressBar Type="ProgressType.Linear" Value="20" Height="60"
IsIndeterminate="true" Minimum="0" Maximum="100">
<ProgressBarAnimation Enable="true"></ProgressBarAnimation>
</SfProgressBar>
<SfProgressBar Type="ProgressType.Linear" Value="40" Height="60"
SecondaryProgress="60" Minimum="0" Maximum="100">
</SfProgressBar>
<SfProgressBar Type="ProgressType.Linear" Value="100" Height="60"
SegmentCount="8" Minimum="0" Maximum="100">
</SfProgressBar>
```



Circular

To get the circular progress bar, set the [Type](#) property to [Circular](#). As shown in the following, it also supports secondary progress, indeterminate, segments, pie progress, and different modes of progress.

ASPX-CS

```
@using Syncfusion.Blazor.ProgressBar
<SfProgressBar Type="ProgressType.Circular" Value="100" Height="60"
Minimum="0" Maximum="100">
</SfProgressBar>
<SfProgressBar Type="ProgressType.Circular" Value="20" Height="60"
IsIndeterminate="true" Minimum="0" Maximum="100">
<ProgressBarAnimation Enable="true"></ProgressBarAnimation>
</SfProgressBar>
<SfProgressBar Type="ProgressType.Circular" Value="40" Height="60"
SecondaryProgress="60" Minimum="0" Maximum="100">
</SfProgressBar>
<SfProgressBar Type="ProgressType.Circular" Value="100" Height="60"
SegmentCount="8" Minimum="0" Maximum="100">
</SfProgressBar>
<SfProgressBar Type="ProgressType.Circular" Value="80" Height="60"
EnablePieProgress="true" Minimum="0" Maximum="100">
</SfProgressBar>
```

**States in Blazor ProgressBar Component**

In this section, the progress can be visualized in different states.

Determinate

This is the default progress state, which can be used when the estimated progress is known.

ASPX-CS

```
@using Syncfusion.Blazor.ProgressBar
<SfProgressBar Type="ProgressType.Linear" Value="100" Height="60"
Minimum="0" Maximum="100">
</SfProgressBar>
```

Indeterminate

When the progress cannot be estimated or calculated, the indeterminate state of the Progress Bar can be used to set the [IsIndeterminate](#) property to true. It can be combined with the determinate mode to know the estimating progress before the actual progress starts.

ASPX-CS

```
@using Syncfusion.Blazor.ProgressBar
<SfProgressBar Type="ProgressType.Linear" Value="20" Height="60"
IsIndeterminate="true" Minimum="0" Maximum="100">
</SfProgressBar>
```



Buffer

When the [SecondaryProgress](#) property value is set to **true**, the secondary progress indicator becomes visible, and the primary progress is dependent on it. Users will be able to view both the primary and the secondary progress simultaneously.

ASPX-CS

```
@using Syncfusion.Blazor.ProgressBar
<SfProgressBar Type="ProgressType.Linear" Value="40" Height="60"
SecondaryProgress="60" Minimum="0" Maximum="100">
</SfProgressBar>
```

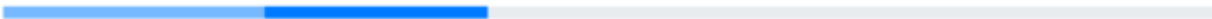


Active

The active animate indicator for the estimated progress can be enabled by setting the [IsActive](#) property to **true** in the `SfProgressBar` and the [Enable](#) property to **true** in the [ProgressBarAnimation](#).

ASPX-CS

```
@using Syncfusion.Blazor.ProgressBar
<SfProgressBar Type="ProgressType.Linear" IsActive="true" Value="40"
Height="60" Minimum="0" Maximum="100">
<ProgressBarAnimation Enable="true"></ProgressBarAnimation>
</SfProgressBar>
```



Striped

The striped visual indicator for the estimated progress can be enabled by setting the [IsStriped](#) property to **true**.

[IsStriped](#) property is only applicable for the [Linear Type](#) of the `Progress Bar`.

ASPX-CS

```
@using Syncfusion.Blazor.ProgressBar
<SfProgressBar Type="ProgressType.Linear" IsStriped="true" Value="40"
Height="60" Minimum="0" Maximum="100">
</SfProgressBar>
```



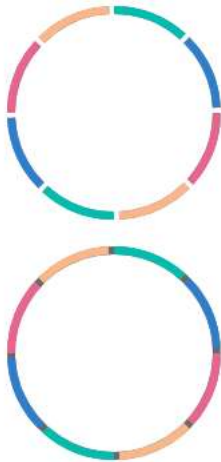
Customization in Blazor ProgressBar Component

Segments

Divide a progress bar into multiple segments by using the [SegmentCount](#) property to visualize the progress for multiple sequential tasks, and the [EnableProgressSegments](#) property to divide the progress into multiple segments without the track. Meanwhile, the [SegmentColor](#) property represents the color of each segment.

ASPX-CS

```
@using Syncfusion.Blazor.ProgressBar
<SfProgressBar Type="ProgressType.Circular" Value="100" Height="180"
SegmentCount="8" SegmentColor='new string[] { "#00bdaf", "#2f7ecc",
"#e9648e", "#fbb78a" }' Minimum="0" Maximum="100" TrackColor="#696969">
</SfProgressBar>
<SfProgressBar Type="ProgressType.Circular" EnableProgressSegments="true"
Value="100" Height="180" SegmentColor='new string[] { "#00bdaf", "#2f7ecc",
"#e9648e", "#fbb78a" }' SegmentCount="8" Minimum="0" Maximum="100"
TrackColor="#696969">
</SfProgressBar>
```



Thickness

The thickness of the track can be customized using the [TrackThickness](#) property, and the progress using the [ProgressThickness](#) property to render the progress bar with different sizes.

ASPX-CS

```
@using Syncfusion.Blazor.ProgressBar
<SfProgressBar Type="ProgressType.Linear" Value="100" Height="60"
Width="90%" TrackThickness="24" ProgressThickness="24"
ShowProgressValue="true" Minimum="0" Maximum="100">
</SfProgressBar>
```



Radius

The radius of the progress bar can be customized using the [Radius](#) property, and the progress edges can be customized using the [CornerRadius](#) property.

ASPX-CS

```
@using Syncfusion.Blazor.ProgressBar
<SfProgressBar Type="ProgressType.Circular" Value="60" Height="160px"
Width="160px" EnableRtl="false" TrackColor="#FFD939" Radius="80%"
InnerRadius="190%" ProgressColor="white" TrackThickness="80"
ProgressThickness="10" CornerRadius="CornerType.Round" Minimum="0"
Maximum="100">
</SfProgressBar>
```



Inner Radius

The inner radius of the progress bar can be customized using the [InnerRadius](#) property.

ASPX-CS

```
@using Syncfusion.Blazor.ProgressBar
<SfProgressBar Type="ProgressType.Circular" Value="60" Height="160px"
Width="160px" EnableRtl="false"
TrackColor="#FFD939" Radius="80%" InnerRadius="80%" ProgressColor="white"
TrackThickness="80" ProgressThickness="10" CornerRadius="CornerType.Round"
Minimum="0" Maximum="100">
</SfProgressBar>
```



Progress Color and Track Color

The color of progress and track can be customized using the [ProgressColor](#) and the [TrackColor](#) properties respectively.

ASPX-CS

```
@using Syncfusion.Blazor.ProgressBar
<SfProgressBar Type="ProgressType.Linear" Value="50" Height="60" Width="90%"
TrackColor="#F8C7D8"
ShowProgressValue="true" InnerRadius="190%" ProgressColor="#E3165B"
TrackThickness="24" CornerRadius="CornerType.Round"
ProgressThickness="24" Minimum="0" Maximum="100">
</SfProgressBar>
```

**Range Colors**

Enhance the readability of progress by visualizing a multiple ranges with different colors, that are mapped to each range. Colors can be mapped to specific ranges using the [ProgressBarRangeColors](#), which holds a collection of the [ProgressBarRangeColor](#) type.

The [Color](#) property represents color to the specified range. The [Start](#) and the [End](#) properties represents start and end range of the color. The [IsGradient](#) property represents whether the gradient effect is applied to the color.

ASPX-CS

```
@using Syncfusion.Blazor.ProgressBar
<SfProgressBar Value="100" IsGradient="true">
<ProgressBarRangeColors>
<ProgressBarRangeColor Start="0" End="25"
Color="#00bdaf"></ProgressBarRangeColor>
<ProgressBarRangeColor Start="25" End="50"
Color="#2f7ecc"></ProgressBarRangeColor>
<ProgressBarRangeColor Start="50" End="75"
Color="#e9648e"></ProgressBarRangeColor>
<ProgressBarRangeColor Start="75" End="100"
Color="#fbb78a"></ProgressBarRangeColor>
</ProgressBarRangeColors>
</SfProgressBar>
```

**RTL**

The progress bar supports right-to-left (RTL) rendering, which can be enabled by setting the [EnableRtl](#) property to **true**.

ASPX-CS

```
@using Syncfusion.Blazor.ProgressBar
<SfProgressBar EnableRtl="true" Value="50" Type="ProgressType.Linear">
</SfProgressBar>
<SfProgressBar EnableRtl="true" Value="80" Type="ProgressType.Circular">
```



```
</SfProgressBar>
```



Annotations and Label in Blazor ProgressBar Component

Annotations

The annotations are used to add text, shapes, or images to the track area in the Progress Bar. It can be added using the [ProgressBarAnnotations](#) collection, and elements that need to be displayed in the track area can be specified using the `ContentTemplate` property in the [ProgressBarAnnotation](#).

ASPX-CS

```
@using Syncfusion.Blazor.ProgressBar
<SfProgressBar Type="ProgressType.Circular" Value="60" Height="160px"
Width="160px" EnableRtl="false"
TrackColor="#FFD939" Radius="100%" InnerRadius="190%" ProgressColor="white"
TrackThickness="80"
ProgressThickness="10" CornerRadius="CornerType.Round" Minimum="0"
Maximum="100">
  <ProgressBarAnnotations>
    <ProgressBarAnnotation>
      <ContentTemplate>
        <div style="font-size:20px;font-weight:bold;color:#ffffff;fill:#ffffff">
          <span>
            60%
          </span>
        </div>
      </ContentTemplate>
    </ProgressBarAnnotation>
  </ProgressBarAnnotations>
</SfProgressBar>
```



Label

When the [ShowProgressValue](#) property is set to **true**, the progress text is rendered in percentage format by default, and can be customized to different types of label formats by using the [Text](#) argument in the [TextRender](#) event.

ASPX-CS

```
@using Syncfusion.Blazor.ProgressBar
<SfProgressBar Type="ProgressType.Linear" Value="50" Height="60" Width="90%"
TrackColor="#F8C7D8"
ShowProgressValue="true" ProgressColor="#E3165B" TrackThickness="24"
CornerRadius="CornerRadius.Round"
ProgressThickness="24" Minimum="0" Maximum="100">
<ProgressBarEvents TextRender="TextHandler"></ProgressBarEvents>
</SfProgressBar>
@code{
public void TextHandler(TextRenderEventArgs args)
{
args.Text = "..."; // Here you can customize the text format.
}
}
```



Animation in Blazor ProgressBar Component

The progress bar supports animation, which can be enabled by setting the [Enable](#) property to **true** in the [ProgressBarAnimation](#). The speed and the delay during an animation can be controlled using the [Duration](#) and the [Delay](#) properties.

ASPX-CS

```
@using Syncfusion.Blazor.ProgressBar
<SfProgressBar Type="ProgressType.Linear" Value="90" Height="60" Width="90%"
TrackColor="#FFFFFF"
ShowProgressValue="true" ProgressColor="#2BB20E" TrackThickness="24"
CornerRadius="CornerRadius.Round"
ProgressThickness="24" Minimum="0" Maximum="100">
<ProgressBarAnimation Enable="true" Duration="2000"
Delay="0"></ProgressBarAnimation>
</SfProgressBar>
```



Range in Blazor ProgressBar Component

The range represents the entire span of the Progress Bar and it can be defined using the [Minimum](#) and the [Maximum](#) properties.

ASPX-CS

```
@using Syncfusion.Blazor.ProgressBar
<SfProgressBar Type="ProgressType.Linear" Value="100" Height="60"
Minimum="0" Maximum="100">
</SfProgressBar>
```

Events in Blazor ProgressBar Component

This section describes the Progress Bar component's events that will be triggered when appropriate actions are performed. The events should be provided to the Progress Bar through the [ProgressBarEvents](#).

ValueChanged

The [ValueChanged](#) event triggers when the progress value is changed.

ASPX-CS

```
@using Syncfusion.Blazor.ProgressBar
<SfProgressBar Type="ProgressType.Linear" Value="80" Height="60" Minimum="0"
Maximum="100">
<ProgressBarEvents ValueChanged="@ValueHandler"></ProgressBarEvents>
</SfProgressBar>
@code{
public void ValueHandler(ProgressValueEventArgs args)
{
// Here you can customize the code.
}
}
```

ProgressCompleted

The [ProgressCompleted](#) event triggers when the progress attains the [Maximum](#) value.

ASPX-CS

```
@using Syncfusion.Blazor.ProgressBar
<SfProgressBar Type="ProgressType.Linear" Value="100" Height="60"
Minimum="0" Maximum="100">
<ProgressBarEvents ProgressCompleted="@ProgressHandler"></ProgressBarEvents>
</SfProgressBar>
@code{
public void ProgressHandler(ProgressValueEventArgs args)
{
// Here you can customize the code.
}
}
```

AnimationComplete

The [AnimationComplete](#) event triggers when an animation is enabled and the progress attains to maximum value.

ASPX-CS

```
@using Syncfusion.Blazor.ProgressBar
```

```

<SfProgressBar Type="ProgressType.Linear" Value="100" Height="60"
Minimum="0" Maximum="100">
<ProgressBarAnimation Enable="true"></ProgressBarAnimation>
<ProgressBarEvents
AnimationComplete="@AnimationHandler"></ProgressBarEvents>
</SfProgressBar>
@code{
public void AnimationHandler(ProgressValueEventArgs args)
{
// Here you can customize the code.
}
}

```

AnnotationRender

The [AnnotationRender](#) event triggers before each annotation is rendered.

ASPX-CS

```

@using Syncfusion.Blazor.ProgressBar
<SfProgressBar Type="ProgressType.Circular" Value="80" Height="160"
Minimum="0" Maximum="100">
<ProgressBarEvents
AnnotationRender="@AnnotationHandler"></ProgressBarEvents>
<ProgressBarAnnotations>
<ProgressBarAnnotation>
<ContentTemplate>
<div style="font-size:20px;font-weight:bold;color:#000000;fill:#ffffff">
<span>
80%
</span>
</div>
</ContentTemplate>
</ProgressBarAnnotation>
</ProgressBarAnnotations>
</SfProgressBar>
@code{
public void AnnotationHandler(AnnotationRenderEventArgs args)
{
// Here you can customize the code.
}
}

```

TextRender

The [TextRender](#) event triggers before the text is rendered.

ASPX-CS

```

@using Syncfusion.Blazor.ProgressBar
<SfProgressBar Type="ProgressType.Linear" ShowProgressValue="true"
Value="100" Height="60" Minimum="0" Maximum="100">
<ProgressBarEvents TextRender="@TextRenderHandler"></ProgressBarEvents>
</SfProgressBar>
@code{
public void TextRenderHandler(TextRenderEventArgs args)
{
// Here you can customize the code.
}
}

```

```
}
}
```

Loaded

The **Loaded** event triggers after the component is rendered.

ASPX-CS

```
@using Syncfusion.Blazor.ProgressBar
<SfProgressBar Type="ProgressType.Linear" Value="100" Height="60"
Minimum="0" Maximum="100">
<ProgressBarEvents Loaded="@LoadedHandler"></ProgressBarEvents>
</SfProgressBar>
@code{
public void LoadedHandler(System.EventArgs args)
{
// Here you can customize the code.
}
}
```

ProgressBar

<!-- markdownlint-disable MD024 -->

Getting Started with Blazor ProgressBar Component

This section briefly explains about how to include Progress Button Component in the Blazor server-side application. You can refer [Getting Started with Syncfusion Blazor for Server-side in Visual Studio](#) page for the introduction and configuring the common specifications.

Importing Syncfusion Blazor component in the application

1. Install the **Syncfusion.Blazor** NuGet package to the application by using the **NuGet Package Manager**.
2. The client-side style resources can be added through [CDN](#) or from [NuGet](#) package in the element of the `~/Pages/_Host.cshtml` page.

Please ensure to check the **Include prerelease** option.

ASPX-CS

```
<head>
<link href="_content/Syncfusion.Blazor/styles/bootstrap4.css"
rel="stylesheet" />
@*<link href="https://cdn.syncfusion.com/blazor/{{ site.blazorversion
}}/styles/bootstrap4.css" rel="stylesheet" />*@
</head>
```

For Internet Explorer 11 kindly refer the polyfills. Refer the [documentation](#) for more information.

ASPX-CS

```
<head>
```

```
<link href="_content/Syncfusion.Blazor/styles/bootstrap4.css"
rel="stylesheet" />
<script
src="https://github.com/Daddoon/Blazor.Polyfill/releases/download/3.0.1/blaz
or.polyfill.min.js"></script>
</head>
```

Adding component package to the application

Open `/_Imports.razor` file and import the **Syncfusion.Blazor.Buttons** package.

ASPX-CS

```
@using Syncfusion.Blazor.SplitButtons
```

Add SyncfusionBlazor service in Startup.cs

Open the **Startup.cs** file and add services required by Syncfusion components. Add **services.AddSyncfusionBlazor()** method in the **ConfigureServices** function as follows.

ASPX-CS

```
using Syncfusion.Blazor;
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void ConfigureServices(IServiceCollection services)
        {
            ....
            ....
            services.AddSyncfusionBlazor();
        }
    }
}
```

To enable custom client side resource loading from CRG or CDN. You need to disable resource loading by **AddSyncfusionBlazor(true)** and load the scripts in the HEAD element of the `~/Pages/_Host.cshtml` page.

ASPX-CS

```
<head>
<environment include="Development">
<script src="https://cdn.syncfusion.com/blazor/{ site.blazorversion
}/syncfusion-blazor.min.js">
</script>
</environment>
</head>
```

Adding Progress Button component to the application

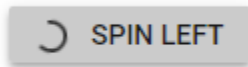
Now, add the Syncfusion Blazor Progress Button component in `razor` page in the `Pages` folder. For example, the Progress Button component is added in the `~/Pages/Index.razor` page.

ASPX-CS

```
<SfProgressBar Content="Spin Left"></SfProgressBar>
```

Run the application

After successful compilation of the application, simply press F5 to run the application. The Blazor Progress Button component will render in the web browser.



See Also

- [Getting Started with Syncfusion Blazor for Client-Side in .NET Core CLI](#)
- [Getting Started with Syncfusion Blazor for Server-Side in Visual Studio](#)
- [Getting Started with Syncfusion Blazor for Server-Side in .NET Core CLI](#)

Native Events in Blazor ProgressBar Component

You can define the native event using an **event** attribute in the component. The value of attribute is treated as an event handler. The event specific data will be available in the event arguments.

The different event argument types for each event are,

- Focus Events - `UIFocusEventArgs`
- Mouse Events - `UIMouseEventArgs`
- Keyboard Events - `UIKeyboardEventArgs`
- Touch Events – `UITouchEventArgs`

List of Native events supported

The following native event support has been provided to the Progress Button component:

```
| List of Native events | | | | |
|---|---|---|---|---|
| onclick | onblur | onfocus | onfocusout |
| onmousemove | onmouseover | onmouseout | onmousedown | onmouseup |
| ondblclick | onkeydown | onkeyup | onkeypress |
| ontouchend | onfocusin | onmouseup | ontouchstart |
```

How to bind click event to Progress Button

The **onclick** attribute is used to bind the click event for Progress Button. Here, the sample code snippets of Progress Button has been explained.

ASPX-CS

```
@using Syncfusion.Blazor.SplitButtons
```

```
<SfProgressBar IsPrimary="true" @onclick="onClick" Content="Spin
Left"></SfProgressBar>
@code {
private void onClick(Microsoft.AspNetCore.Components.Web.MouseEventArgs
args)
{
//onclick Event triggered
}
}
```

Types and Icons in Blazor ProgressBar Component

This section explains the different types and icons of Progress Button.

Types

The types of Blazor Progress Button are as follows:

- Outline Progress Button
- Round Progress Button
- Primary Progress Button

Outline Progress Button

An outline Progress Button has a border with transparent background. To create an outline Progress Button, set the [CssClass](#) property to `e-outline`.

ASPX-CS

```
@using Syncfusion.Blazor.SplitButtons
<SfProgressBar Content="Outline" EnableProgress="true" CssClass="e-
outline e-success">
<ProgressBarSpinSettings
Position="SpinPosition.Center"></ProgressBarSpinSettings>
</SfProgressBar>
```

Round Progress Button

A round Progress Button is shaped like a circle. Usually, it contains an icon representing its action. To create a round Progress Button, set the [CssClass](#) property to `e-round`.

ASPX-CS

```
@using Syncfusion.Blazor.SplitButtons
<SfProgressBar CssClass="e-round e-small e-success" IconCss="e-icons e-
play-icon">
<ProgressBarSpinSettings
Position="SpinPosition.Center"></ProgressBarSpinSettings>
</SfProgressBar>
<style>
.e-play-icon::before {
content: '\e324';
}
</style>
```


Primary Progress Button

The primary progress button is styled with background color and it is used to represent a primary action. To create a Primary Progress Button, set the [IsPrimary](#) property to `true`.

ASPX-CS

```
@using Syncfusion.Blazor.SplitButtons
<SfProgressBar Content="Primary" IsPrimary="true"></SfProgressBar>
```

OUTLINE



PRIMARY

Icons

Progress Button with font icons

The Progress Button can have an icon to provide the visual representation of the action. To place the icon on a Progress 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 Progress Button. You can customize the icon's position by using the [IconPosition](#) property.

ASPX-CS

```
@using Syncfusion.Blazor.SplitButtons
@using Syncfusion.Blazor.Buttons
<SfProgressBar IconCss="e-icons e-play-icon"
IconPosition="IconPosition.Left">PLAY
</SfProgressBar>
<style>
.e-play-icon::before {
content: '\e324';
}
</style>
```

▶ PLAY

<!-- markdownlint-disable MD002 MD022 -->

Spinner and Progress in Blazor ProgressBar Component

This section explains the Spinner and Progress of Progress Button.

Spinner

Change spinner position

Spinner position can be changed by modifying the **Position** property of [SpinSettings](#). By default, the spinner is positioned at the left of the Progress Button. You can position it at the [Left](#), [Right](#), [top](#), [bottom](#), or [Center](#) of the text content.

Change spinner size

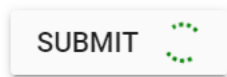
Spinner size can be changed by modifying the **Width** property of [SpinSettings](#). In this demo, the width is set to **20** to change the spinner size.

Spinner template

You can use custom spinner by specifying the [SpinTemplate](#) property of [SpinSettings](#) with custom styles.

ASPX-CS

```
@using Syncfusion.Blazor.SplitButtons
<SfProgressBar Content="Submit">
  <ProgressBarSpinSettings Position="SpinPosition.Right" Width="20">
    <SpinTemplate>
      <div class="template"></div>
    </SpinTemplate>
  </ProgressBarSpinSettings>
</SfProgressBar>
<style>
  @@keyframes custom-rolling {
    0% {
      -webkit-transform: rotate(0deg);
      transform: rotate(0deg);
    }
    100% {
      -webkit-transform: rotate(360deg);
      transform: rotate(360deg);
    }
  }
  .template {
    border: 2px solid green;
    border-style: dotted;
    border-radius: 50%;
    border-top-color: transparent;
    border-bottom-color: transparent;
    height: 16px;
    width: 16px;
  }
  .template {
    -webkit-animation: custom-rolling 1.3s linear infinite;
    animation: custom-rolling 1.3s linear infinite;
  }
</style>
```



Progress

Content animation

The [Content](#) of the Progress Button can be animated during progress using the [Effect](#) property of [AnimationSettings](#). You can also set custom duration and timing function using the [Duration](#) and [Easing](#) properties. The possible [Effect](#) values are [None](#), [SlideLeft](#), [SlideRight](#), [SlideUp](#), [SlideDown](#), [ZoomIn](#), and [ZoomOut](#).

ASPX-CS

```
@using Syncfusion.Blazor.SplitButtons
<SfProgressButton Content="Slide Right">
  <ProgressButtonSpinSettings
    Position="SpinPosition.Center"></ProgressButtonSpinSettings>
  <ProgressButtonAnimationSettings
    Effect="Syncfusion.Blazor.SplitButtons.AnimationEffect.SlideRight" Duration=
    "400" Easing="Linear"></ProgressButtonAnimationSettings>
</SfProgressButton>
```



Change step of the Progress Button

The progress can be visualized at the specified interval by changing the [Step](#) property in the [OnBegin](#) event of the Progress Button. In this demo, the Step property is set to [20](#) to show progress at every 20% increment.

ASPX-CS

```
@using Syncfusion.Blazor.SplitButtons
<SfProgressButton EnableProgress="true" Content="Progress Step" CssClass="e-
hide-spinner">
  <ProgressButtonEvents OnBegin="Begin"></ProgressButtonEvents>
</SfProgressButton>
@code{
  private void Begin(Syncfusion.Blazor.SplitButtons.ProgressEventArgs args)
  {
    args.Step = 20;
  }
}
```

PROGRESS STEP

The class `e-hide-spinner` hides the spinner in the Progress Button, For more information, see [hide spinner](#) section.

Change Progress state dynamically

The progress state can be changed dynamically by modifying the [Percent](#) event argument in the [OnBegin](#) [Progressing](#) [OnEnd](#) events. In this example, on 40% completion of progress, the Percent property is set to 90 to show dynamic change of the progress state. The progress state can be changed dynamically by modifying the Percent property in the [Progress](#) event.

ASPX-CS

```
@using Syncfusion.Blazor.SplitButtons
<SfProgressBar EnableProgress="true" Content="@Content" Duration="15000"
  CssClass="e-hide-spinner">
  <ProgressBarEvents OnBegin="Begin" Progressing="Progressing"
    OnEnd="End"></ProgressBarEvents>
</SfProgressBar>
@code {
public string Content = "Progress";
public void Begin(Syncfusion.Blazor.SplitButtons.ProgressBarEventArgs args)
{
Content = "Progress " + args.Percent + " %";
}
public void Progressing(Syncfusion.Blazor.SplitButtons.ProgressBarEventArgs args)
{
Content = "Progressing " + args.Percent + " %";
if (args.Percent == 40)
{
args.Percent = 90;
}
}
public void End(Syncfusion.Blazor.SplitButtons.ProgressBarEventArgs args)
{
Content = "Progress " + args.Percent + " %";
}
}
```

PROGRESSING 94%

Start and Stop Methods

You can pause and resume the progress using the [Stop](#) and [Start](#) methods, respectively. In this example, clicking the Progress Button will pause and resume the progress.

ASPX-CS

```
@using Syncfusion.Blazor.SplitButtons
<SfProgressButton Content="@Content" EnableProgress="true"
CssClass="@CssClass" IconCss="@IconCss" OnClick="Click" @ref="ProgressBtn">
<ProgressButtonEvents OnEnd="End"></ProgressButtonEvents>
</SfProgressButton>
@code {
SfProgressButton ProgressBtn;
public string Content = "Download";
public string CssClass = "e-hide-spinner";
public string IconCss = "e-icons e-download";
public async Task Click()
{
if(Content == "Download")
{
Content = "Pause";
IconCss = "e-icons e-pause";
}
else if (this.Content == "Pause")
{
Content = "Resume";
IconCss = "e-icons e-play";
await ProgressBtn.Stop();
}
else if (this.Content == "Resume")
{
Content = "Pause";
IconCss = "e-icons e-pause";
await ProgressBtn.Start();
}
}
public void End(Syncfusion.Blazor.SplitButtons.ProgressEventArgs args)
{
Content = "Download";
IconCss = "e-icons e-download";
}
}
<style>
.e-download::before {
content: '\e75d';
}
.e-play::before {
content: '\e72d';
}
.e-pause::before {
content: '\e757';
}
</style>
```



EndProgressAsync Method

You can complete the progress by using the [EndProgressAsync](#) method and it will also hide the spinner. In this example, another button has been added to complete the current progress of the progress button.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.SplitButtons
<SfProgressBar Content="Progress Button" EnableProgress="true"
@ref="ProgressBtnObj">
<ProgressBarSpinSettings
Position="SpinPosition.Left"></ProgressBarSpinSettings>
<ProgressBarAnimationSettings
Effect="Syncfusion.Blazor.SplitButtons.AnimationEffect.SlideRight"
Duration="400" Easing="Linear"></ProgressBarAnimationSettings>
</SfProgressBar>
<SfButton @onclick="OnCompleteClick">Complete</SfButton>
@code
{
SfProgressBar ProgressBtnObj;
private async Task OnCompleteClick()
{
await ProgressBtnObj.EndProgressAsync();
}
}
```

Accessibility in Blazor ProgressBar Component

ARIA attributes

The web accessibility makes web content and web applications more accessible for people with disabilities. Mostly, it helps in dynamic content change and development of advanced user interface controls with AJAX, HTML, JavaScript, and related technologies.

The Progress Button provides a built-in compliance with WAI-ARIA specifications. The WAI-ARIA support is achieved using the `aria-label`, `aria-valuemin`, `aria-valuemax`, and `aria-valuenow` attributes in the Progress Button. It helps by providing information about the widget for assistive technology in the screen readers.

| Properties | Functionality |

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

| `aria-label` | Indicates the text content of the Progress Button. |

| `aria-valuemin` | Indicates the minimum value for the Progress Button. |

| `aria-valuemax` | Indicates the maximum value for the Progress Button. |

| `aria-valuenow` | Indicates the current value for the Progress Button. |

Keyboard interaction

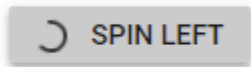
<!-- markdownlint-disable MD033 -->

Keyboard shortcuts	Actions
--------------------	---------

Enter / Space	Starts the progress
---------------	---------------------

ASPX-CS

```
@using Syncfusion.Blazor.SplitButtons
<SfProgressBar Content="Spin Left"></SfProgressBar>
```



Styles and Appearances in Blazor ProgressBar Component

To modify the ProgressBar appearance, you need to override the default CSS of ProgressBar component. Please find the list of CSS classes and its corresponding section in ProgressBar. Also, you have an option to create your own custom theme for the controls using our [Theme Studio](#).

| CSS Class | Purpose of Class |

| ---- | ---- |

| .e-progress-btn | To customize the progress button. |

| .e-progress-btn: hover | To customize the progress button on hover. |

| .e-progress-btn: focus | To customize the progress button on focus. |

| .e-progress-btn .e-spinner-pane .e-spinner-inner svg .e-path-circle | To customize the progress button spinner. |

How To

Change text content and styles of the Blazor ProgressBar Component

You can change the text content and styles of the Progress Button during progress state by changing the text content and the [CssClass](#) property at the [OnBegin](#) and [OnEnd](#) events.

ASPX-CS

```
@using Syncfusion.Blazor.SplitButtons
<SfProgressBar Content="@Content" EnableProgress="true"
CssClass="@CssClass" Duration="4000">
<ProgressBarEvents OnBegin="Begin" OnEnd="End"></ProgressBarEvents>
</SfProgressBar>
@code {
public string Content = "Upload";
public string CssClass = "e-hide-spinner";
public void Begin(Syncfusion.Blazor.SplitButtons.ProgressBarEventArgs args)
{
Content = "Uploading...";
CssClass = "e-hide-spinner e-info";
}
public async Task End(Syncfusion.Blazor.SplitButtons.ProgressBarEventArgs args)
{
Content = "Success";
CssClass = "e-hide-spinner e-success";
}
```

```
await Task.Delay(1000);
Content = "Upload";
CssClass = "e-hide-spinner";
}
```

UPLOADING...

Customize progress using cssClass in Blazor ProgressBar Component

You can customize the background filler UI using the [CssClass](#) property.

- Adding `e-vertical` to `CssClass` shows vertical progress.
- Adding `e-progress-top` to `CssClass` shows progress at the top.

You can also show reverse progress by adding custom class to the [CssClass](#) property.

ASPX-CS

```
@using Syncfusion.Blazor.SplitButtons
<SfProgressBar EnableProgress="true" CssClass="e-hide-spinner e-vertical"
Duration="4000" Content="Vertical Progress"></SfProgressBar>
<SfProgressBar EnableProgress="true" CssClass="e-hide-spinner e-progress-
top" Duration="4000" Content="Progress Top"></SfProgressBar>
<SfProgressBar EnableProgress="true" CssClass="e-hide-spinner e-reverse-
progress" Duration="4000" Content="Reverse Progress"></SfProgressBar>
<style>
.e-reverse-progress .e-progress {
right: 0;
left: auto;
}
</style>
```

VERTICAL PROGRESS

PROGRESS TOP

REVERSE PROGRESS

Hide spinner in Blazor ProgressBar Component

You can hide spinner in the Progress Button by setting the `e-hide-spinner` property to [CssClass](#).

ASPX-CS

```
@using Syncfusion.Blazor.SplitButtons
<SfProgressBar EnableProgress="true" CssClass="e-hide-spinner"
Content="Progress"></SfProgressBar>
```


PROGRESS

Stop Progress Indicator in Blazor ProgressBar Component

You can stop the progress indicator using [EndProgressAsync](#) method. In the following sample, the progress is stopped by clicking the **STOP** button.

ASPX-CS

```
@using Syncfusion.Blazor.SplitButtons
@using Syncfusion.Blazor.Buttons
<SfProgressBar Content="Spin Left" IsPrimary="true"
@ref="ProgressBtnObj"></SfProgressBar>
<SfButton Content="Stop" OnClick="clicked"></SfButton>
@code {
SfProgressBar ProgressBtnObj;
private async Task clicked()
{
await ProgressBtnObj.EndProgressAsync();
}
}
```



Trace Events in Blazor ProgressBar Component

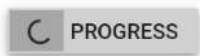
The Progress Button component triggers events based on its actions. The events can be used as extension points to perform custom operations.

The events available in Progress Button are [OnFailure](#), [OnBegin](#), [Progressing](#), and [OnEnd](#).

ASPX-CS

```
@using Syncfusion.Blazor.SplitButtons
<div id="preview">@eventName Event Triggered</div>
<SfProgressBar Content="Progress">
<ProgressBarEvents OnBegin="Begin" OnEnd="End"
Progressing="Progressing"></ProgressBarEvents>
</SfProgressBar>
@code {
private string eventName = "No";
public void Begin(Syncfusion.Blazor.SplitButtons.ProgressEventArgs args)
{
eventName = "Begin";
}
public void End(Syncfusion.Blazor.SplitButtons.ProgressEventArgs args)
{
eventName = "End";
}
}
```

```
public void Progressing(Syncfusion.Blazor.SplitButtons.ProgressEventArgs
args)
{
    eventName = "Progressing";
}
}
<style>
#preview {
float: right;
padding: 0 350px 0 0;
}
</style>
```



Progressing Event Triggered

QueryBuilder

Getting Started with Blazor QueryBuilder Component

This section briefly explains how to include Query Builder Component in the Blazor server-side application. Refer [Getting Started with Syncfusion Blazor for Server-side in Visual Studio](#) page for the introduction and configuring the common specifications.

{% youtube

"youtube:https://www.youtube.com/watch?v=jyWU7XSg3WI"%}

You can also explore our [Blazor Query Builder example](#) to know how to render and configure the query builder.

Importing Syncfusion Blazor component in the application

1. Install the **Syncfusion.Blazor** NuGet package to the application by using the **NuGet Package Manager**.
2. Add the client-side style resources through [CDN](#) or from [NuGet](#) package in the `element` of the `~/Pages/_Host.cshtml` page.

Please ensure to check the **Include prerelease** option.

ASPX-CS

```
<head>
<link href="_content/Syncfusion.Blazor/styles/material.css" rel="stylesheet"
/>
<!--CDN-->
@*<link href="https://cdn.syncfusion.com/blazor/{ site.blazorversion
}}/material.css" rel="stylesheet" />*@
</head>
```

For Internet Explorer 11 kindly refer the polyfills. Refer the [documentation](#) for more information.

ASPX-CS

```
<head>
<environment include="Development">
<link href="_content/Syncfusion.Blazor/styles/material.css" rel="stylesheet"
/>
<script
src="https://github.com/Daddoon/Blazor.Polyfill/releases/download/3.0.1/blaz
or.polyfill.min.js"></script>
</environment>
</head>
```

Adding component package to the application

Open `/_Imports.razor` file and import the **Syncfusion.Blazor.QueryBuilder** package.

CSHARP

```
@using Syncfusion.Blazor.QueryBuilder
```

Add SyncfusionBlazor service in Startup.cs

Open the **Startup.cs** file and add services required by Syncfusion components. Add **services.AddSyncfusionBlazor()** method in the ConfigureServices function.

CSHARP

```
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void ConfigureServices(IServiceCollection services)
        {
            ....
            ....
            services.AddSyncfusionBlazor();
        }
    }
}
```

To enable custom client side resource loading from CRG or CDN. You need to disable resource loading by **AddSyncfusionBlazor(true)** and load the scripts in the HEAD element of the `~/Pages/_Host.cshtml` page.

ASPX-CS

```
<head>
<script src="https://cdn.syncfusion.com/blazor/{ { site.blazorversion
}}/syncfusion-blazor.min.js">
</script>
</head>
```

Adding Query Builder component to the application

Now, add the [Blazor Query Builder](#) component in razor page in the Pages folder. For example, the Query Builder component is added in the ~/Pages/Index.razor page.

ASPX-CS

```
@using Syncfusion.Blazor.QueryBuilder
<SfQueryBuilder TValue="EmployeeDetails">
<QueryBuilderColumns>
<QueryBuilderColumn Field="EmployeeID" Label="Employee ID"
Type="ColumnType.Number"></QueryBuilderColumn>
<QueryBuilderColumn Field="FirstName" Label="First Name"
Type="ColumnType.String"></QueryBuilderColumn>
<QueryBuilderColumn Field="TitleOfCourtesy" Label="Title of Courtesy"
Type="ColumnType.Boolean" Values="Values"></QueryBuilderColumn>
<QueryBuilderColumn Field="Title" Label="Title"
Type="ColumnType.String"></QueryBuilderColumn>
<QueryBuilderColumn Field="HireDate" Label="Hire Date"
Type="ColumnType.Date"></QueryBuilderColumn>
<QueryBuilderColumn Field="Country" Label="Country"
Type="ColumnType.String"></QueryBuilderColumn>
<QueryBuilderColumn Field="City" Label="City"
Type="ColumnType.String"></QueryBuilderColumn>
</QueryBuilderColumns>
</SfQueryBuilder>
@code {
private string[] Values = new string[] { "Mr.", "Mrs." };
public class EmployeeDetails
{
public int EmployeeID { get; set; }
public string FirstName { get; set; }
public bool TitleOfCourtesy { get; set; }
public string Title { get; set; }
public DateTime HireDate { get; set; }
public string Country { get; set; }
public string City { get; set; }
}
}
```

Run the application

After successful compilation of the application, simply press F5 to run the application. The [Blazor Query Builder](#) component will render in the web browser.

The screenshot shows the Blazor QueryBuilder interface. It features two main query groups, each with its own AND/OR selector and a plus sign to add more conditions. The first group contains two conditions: 'Employee ID' equal to 1 and 'Title' equal to 'Sales Manager'. The second group contains two conditions: 'Title Of Courtesy' equal to 'Mr.' (selected with a radio button) and 'Employee ID' equal to 1. The two groups are connected by an 'OR' operator, which is highlighted in red. Each condition has a dropdown for the field, a dropdown for the operator (currently 'Equal'), a text input for the value, and a close button (X).

See Also

- [Getting Started with Syncfusion Blazor for Client-Side in .NET Core CLI](#)
- [Getting Started with Syncfusion Blazor for Server-Side in Visual Studio](#)
- [Getting Started with Syncfusion Blazor for Server-Side in .NET Core CLI](#)

Columns in Blazor QueryBuilder Component

The column definitions are used as the [DataSource](#) schema in the Query Builder. This plays a vital role in rendering column values. The Query Builder operations such as create or delete conditions and create or delete groups are performed based on the column definitions. The [Field](#) property of the [Columns](#) is necessary to map the data source values in the query builder columns.

If the column field is not specified in the data source, the column values will be empty.

Auto generation

The [Columns](#) are automatically generated from the datasource when the [Columns](#) declaration is empty or undefined while initializing the query builder. All the columns in the [DataSource](#) are bound as the query builder columns.

When columns are auto-generated, the column type will be determined from the first record of the data source.

ASPX-CS

```
@using Syncfusion.Blazor.QueryBuilder
<SfQueryBuilder DataSource="@EmployeeData"></SfQueryBuilder>
@code {
//Local datasource
public List<EmployeeDetails> EmployeeData = new List<EmployeeDetails>
{
    new EmployeeDetails{ FirstName = "Martin", EmployeeID = 1001, Country =
    "England", City = "Manchester", HireDate = new DateTime(2014, 4, 23) },
    new EmployeeDetails{ FirstName = "Benjamin", EmployeeID = 1002, Country =
    "England", City = "Birmingham", HireDate = new DateTime(2014, 6, 19) },
}
```

```

new EmployeeDetails{ FirstName = "Stuart", EmployeeID = 1003, Country =
"England", City = "London", HireDate = new DateTime(2014, 7, 4) },
new EmployeeDetails{ FirstName = "Ben", EmployeeID = 1004, Country = "USA",
City = "California", HireDate = new DateTime(2014, 8, 15) },
new EmployeeDetails{ FirstName = "Joseph", EmployeeID = 1005, Country =
"Spain", City = "Madrid", HireDate = new DateTime(2014, 8, 29) }
};
public class EmployeeDetails
{
public string FirstName { get; set; }
public int EmployeeID { get; set; }
public string Country { get; set; }
public string City { get; set; }
public DateTime HireDate { get; set; }
}
}

```

Labels

By default, the column label is displayed from the column [Field](#) value. To override the default label, you have to define the [Label](#) value.

If both the field and label are not defined in the column, the column renders with **empty** text.

Operators

The operator for a column can be defined in the [Columns](#) property. You can directly set the custom operators using [OperatorsModel](#) in columns **Operator** property. The available operators and its supported data types are:

Operators	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, Number ,Date, Boolean
notequal	Checks for values not equal to the specified value.	String, Number, Date, Boolean
greaterthan	Checks whether the value is greater than the specified value.	Date, Number
greaterthanorequal	Checks whether a value is greater than or equal to the specified value.	Date, Number
lessthan	Checks whether the value is lesser than the specified value.	Date, Number
lessthanorequal	Checks whether the value is lesser than or equal to the specified value.	Date, Number
between	Checks whether the value is between the two-specific value.	Date, Number
notbetween	Checks whether the value is not between the two-specific value.	Date, Number
in	Checks whether the value is one of the specific values.	String, Number
notin	Checks whether the value is not in the specific values.	String, Number

- | isempty | Checks whether the value is empty. | String |
- | isnotempty | Checks whether the value is not empty. | String |
- | isnull | Checks whether the value is null. | String, Number |
- | isnotnull | Checks whether the value is not null. | String, Number |

Step

The [Blazor Query Builder](#) allows to set the step values to the number fields. It allows to increase or decrease the numeric value with the predefined Step value using the [Step](#) property.

By default the Step value is 1.

ASPX-CS

```
@using Syncfusion.Blazor.QueryBuilder
<SfQueryBuilder TValue="EmployeeDetails">
  <QueryBuilderColumns>
    <QueryBuilderColumn Field="EmployeeID" Label="Employee ID"
      Type="ColumnType.Number" Step="5"></QueryBuilderColumn>
    <QueryBuilderColumn Field="FirstName" Label="First Name"
      Type="ColumnType.String"></QueryBuilderColumn>
    <QueryBuilderColumn Field="Title" Label="Title"
      Type="ColumnType.String"></QueryBuilderColumn>
  </QueryBuilderColumns>
</SfQueryBuilder>

@code {
  public class EmployeeDetails
  {
    public int EmployeeID { get; set; }
    public string FirstName { get; set; }
    public bool TitleOfCourtesy { get; set; }
    public string Title { get; set; }
    public DateTime HireDate { get; set; }
    public string Country { get; set; }
    public string City { get; set; }
  }
}
```

You can also explore our [Blazor Query Builder example](#) to know how to render and configure the query builder.

Format

The [Blazor Query Builder](#) formats date and number values. Use the [Format](#) property, to format date and number values.

From the standard numeric format, you can use the numeric related format specifiers such as n,p, and c in the format property. By using these format specifiers, you can achieve the percentage and currency textbox behavior.

In the following sample, the date field is formatted as MM/yyyy/dd and the number field is formatted as currency type.

ASPX-CS

```
@using Syncfusion.Blazor.QueryBuilder
```

```

<SfQueryBuilder TValue="EmployeeDetails">
  <QueryBuilderRule Condition="and" Rules="Rules"></QueryBuilderRule>
  <QueryBuilderColumns>
    <QueryBuilderColumn Field="HireDate" Label="Hire Date"
      Type="ColumnType.Date" Format="MM-yyyy-dd"></QueryBuilderColumn>
    <QueryBuilderColumn Field="FirstName" Label="First Name"
      Type="ColumnType.String"></QueryBuilderColumn>
    <QueryBuilderColumn Field="Salary" Label="Salary" Type="ColumnType.Number"
      Format="c2"></QueryBuilderColumn>
  </QueryBuilderColumns>
</SfQueryBuilder>
@code {
  List<RuleModel> Rules = new List<RuleModel>()
  {
    new RuleModel { Field="HireDate", Label="HireDate", Type="Date",
      Operator="equal", Value = new
      DateTime(DateTime.Now.Year,DateTime.Now.Month,4) },
    new RuleModel { Field="Salary", Label="Salary", Type="Number",
      Operator="greaterthan", Value = 23 }
  };
  public class EmployeeDetails
  {
    public int EmployeeID { get; set; }
    public string FirstName { get; set; }
    public bool TitleOfCourtesy { get; set; }
    public string Title { get; set; }
    public DateTime HireDate { get; set; }
    public string Country { get; set; }
    public string City { get; set; }
    public int Salary { get; set; }
  }
}

```

The screenshot shows a Blazor Query Builder interface. At the top, there are three buttons: 'AND' (highlighted in red), 'OR', and a '+' button. Below these, there are two rows of rules. The first row shows 'Hire Date' selected from a dropdown, followed by 'Equal' from another dropdown, and '03-2021-04' as the value. The second row shows 'Salary' selected from a dropdown, followed by 'Greater Than' from another dropdown, and '\$23.00' as the value. Each rule has a small 'X' icon to its right for removal. The interface is clean and modern, with a light gray background.

You can also explore our [Blazor Query Builder example](#) to know how to render and configure the query builder.

Validations

Validation allows to validate the conditions and it display errors for invalid fields. To enable validation in the Query Builder, set [AllowValidation](#) to true. Column fields are validated after setting the [AllowValidation](#) property to true.

You can set **Min** and **Max** values for number values.

ASPX-CS

```
@using Syncfusion.Blazor.QueryBuilder
<SfQueryBuilder TValue="EmployeeDetails" AllowValidation="true">
  <QueryBuilderColumnValidation Max="0"
  Min="100"></QueryBuilderColumnValidation>
  <QueryBuilderColumns>
    <QueryBuilderColumn Field="EmployeeID" Label="Employee ID"
    Type="ColumnType.Number"></QueryBuilderColumn>
    <QueryBuilderColumn Field="FirstName" Label="First Name"
    Type="ColumnType.String"></QueryBuilderColumn>
    <QueryBuilderColumn Field="TitleOfCourtesy" Label="Title of Courtesy"
    Type="ColumnType.Boolean" Values="Values"></QueryBuilderColumn>
    <QueryBuilderColumn Field="Title" Label="Title"
    Type="ColumnType.String"></QueryBuilderColumn>
    <QueryBuilderColumn Field="HireDate" Label="Hire Date"
    Type="ColumnType.Date"></QueryBuilderColumn>
    <QueryBuilderColumn Field="Country" Label="Country"
    Type="ColumnType.String"></QueryBuilderColumn>
    <QueryBuilderColumn Field="City" Label="City"
    Type="ColumnType.String"></QueryBuilderColumn>
  </QueryBuilderColumns>
</SfQueryBuilder>
@code {
private string[] Values = new string[] { "Mr.", "Mrs." };
public class EmployeeDetails
{
public int EmployeeID { get; set; }
public string FirstName { get; set; }
public bool TitleOfCourtesy { get; set; }
public string Title { get; set; }
public DateTime HireDate { get; set; }
public string Country { get; set; }
public string City { get; set; }
}
}
```

The image shows a user interface for the Blazor QueryBuilder component. At the top, there are three buttons: 'AND' (highlighted in red), 'OR', and a '+' icon. Below these is a dropdown menu with the text 'Select a field' and a downward arrow. To the right of the dropdown is a close button (X). A red tooltip with the text 'This field is required' is pointing to the dropdown. At the bottom of the interface is a red button with the text 'VALIDATION'.

Data Binding in Blazor QueryBuilder Component

The [Blazor Query Builder](#) uses **SfDataManager** to bind the **dataSource** which supports both RESTful JSON data services binding and **IEnumerable** binding. The [DataSource](#) property can be assigned either

with the instance of `SfDataManager` or list of objects. It supports the following kinds of data binding method:

- List Binding
- Remote data

When using `DataSource` as `IEnumerable<T>` component type, (`TValue`) will be inferred from its value. While using `SfDataManager` for data binding then the `TValue` must be provided explicitly in the query builder component.

List Binding

To bind list data to the [Blazor Query Builder](#), you can assign the [DataSource](#) property with a list of data.

ASPX-CS

```
@using Syncfusion.Blazor.QueryBuilder
<SfQueryBuilder DataSource="@EmployeeData">
<QueryBuilderRule Condition="or" Rules="@Rules"></QueryBuilderRule>
<QueryBuilderColumns>
<QueryBuilderColumn Field="EmployeeID" Label="Employee ID"
Type="ColumnType.Number"></QueryBuilderColumn>
<QueryBuilderColumn Field="FirstName" Label="First Name"
Type="ColumnType.String"></QueryBuilderColumn>
<QueryBuilderColumn Field="HireDate" Label="Hire Date"
Type="ColumnType.Date"></QueryBuilderColumn>
<QueryBuilderColumn Field="Country" Label="Country"
Type="ColumnType.String"></QueryBuilderColumn>
</QueryBuilderColumns>
</SfQueryBuilder>
@code {
//Rules binding
List<RuleModel> Rules = new List<RuleModel>()
{
new RuleModel { Field="Country", Label="Country", Type="String",
Operator="equal", Value = "England" },
new RuleModel { Field="EmployeeID", Label="EmployeeID", Type="Number",
Operator="notequal", Value = 1001 }
};
//List Datas
public List<EmployeeDetails> EmployeeData = new List<EmployeeDetails>
{
new EmployeeDetails{ FirstName = "Martin", EmployeeID = 1001, Country =
"England", City = "Manchester", HireDate = new DateTime(2014, 4, 23) },
new EmployeeDetails{ FirstName = "Benjamin", EmployeeID = 1002, Country =
"England", City = "Birmingham", HireDate = new DateTime(2014, 6, 19) },
new EmployeeDetails{ FirstName = "Stuart", EmployeeID = 1003, Country =
"England", City = "London", HireDate = new DateTime(2014, 7, 4) },
new EmployeeDetails{ FirstName = "Ben", EmployeeID = 1004, Country = "USA",
City = "California", HireDate = new DateTime(2014, 8, 15) },
new EmployeeDetails{ FirstName = "Joseph", EmployeeID = 1005, Country =
"Spain", City = "Madrid", HireDate = new DateTime(2014, 8, 29) }
};
public class EmployeeDetails
{
public string FirstName { get; set; }
```

```
public int EmployeeID { get; set; }
public string Country { get; set; }
public string City { get; set; }
public DateTime HireDate { get; set; }
}
```

Remote Data

To bind remote data to the [Blazor Query Builder](#) component, assign service data as an instance of [SfDataManager](#) to the [DataSource](#) property or by using [SfDataManager](#) component. To interact with remote data source, provide the endpoint [Url](#).

Refer to the following code example for remote Data binding using [OData](#) service.

Binding with OData services

[OData](#) is a standardized protocol for creating and consuming data. You can retrieve data from OData service using the [SfDataManager](#). Refer to the following code example for remote Data binding using OData service.

ASPX-CS

```
@using Syncfusion.Blazor.QueryBuilder
@using Syncfusion.Blazor.Data
<SfQueryBuilder TValue="OrderDetails" Width="70%">
  <SfDataManager
    Url="https://js.syncfusion.com/ejServices/Wcf/Northwind.svc/Orders"
    Adaptor="Syncfusion.Blazor.Adaptors.ODataAdaptor"></SfDataManager>
  </SfQueryBuilder>
@code {
  public class OrderDetails
  {
    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; }
  }
}
```

Binding with OData v4 services

The ODataV4 is an improved version of OData protocols, and the [SfDataManager](#) can also retrieve and consume OData v4 services. For more details on OData v4 services, refer to the [OData documentation](#). To bind OData v4 service, use the [ODataV4Adaptor](#).

ASPX-CS

```
@using Syncfusion.Blazor.QueryBuilder
@using Syncfusion.Blazor.Data
```

```

<SfQueryBuilder TValue="OrderDetails" Width="70%">
  <SfDataManager
    Url="https://services.odata.org/V4/Northwind/Northwind.svc/Orders"
    Adaptor="Syncfusion.Blazor.Adaptors.ODataV4Adaptor"></SfDataManager>
  </SfQueryBuilder>
@code {
public class OrderDetails
{
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

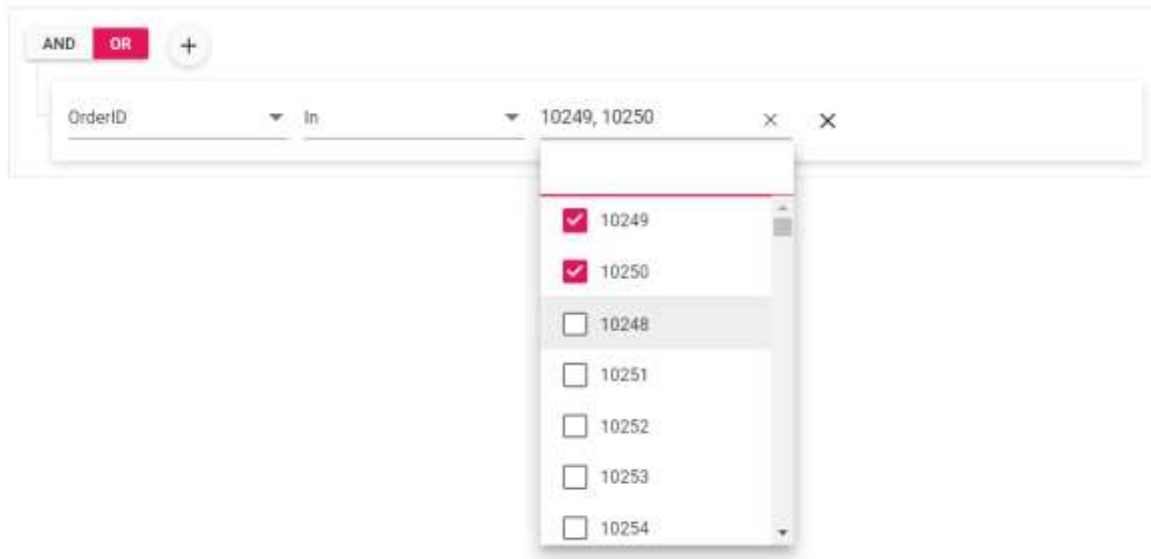
The WebApiAdaptor can be used to bind query builder with Web API created using OData endpoint.

ASPX-CS

```

@using Syncfusion.Blazor.QueryBuilder
@using Syncfusion.Blazor.Data
<SfQueryBuilder TValue="OrderDetails">
  <SfDataManager Url="https://ej2services.syncfusion.com/production/web-
    services/api/Orders"
    Adaptor="Syncfusion.Blazor.Adaptors.WebApiAdaptor"></SfDataManager>
  </SfQueryBuilder>
@code {
public class OrderDetails
{
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; }
}
}

```



You can also explore our [Blazor Query Builder example](#) to know how to render and configure the query builder.

Filtering in Blazor QueryBuilder Component

The [Blazor Query Builder](#) allows to create or delete conditions and groups. You can use [ShowButtons](#) to enable/disable these buttons.

You can **create** or **delete** conditions by interacting through the user interface and methods.

- Use the [AddRule](#), and [DeleteRule](#) methods to create/delete conditions.
- Use [AddGroup](#), and [DeleteGroup](#) methods to create/delete groups.

ASPX-CS

```
@using Syncfusion.Blazor.QueryBuilder
@using Syncfusion.Blazor.Buttons
<SfQueryBuilder TValue="EmployeeDetails" @ref="QueryBuilderObj">
<QueryBuilderShowButtons RuleDelete="true" GroupDelete="true"
GroupInsert="true"></QueryBuilderShowButtons>
<QueryBuilderRule Condition="or" Rules="@Rules"></QueryBuilderRule>
<QueryBuilderColumns>
<QueryBuilderColumn Field="EmployeeID" Label="Employee ID"
Type="ColumnType.Number"></QueryBuilderColumn>
<QueryBuilderColumn Field="FirstName" Label="First Name"
Type="ColumnType.String"></QueryBuilderColumn>
<QueryBuilderColumn Field="TitleOfCourtesy" Label="Title Of Courtesy"
Type="ColumnType.Boolean"></QueryBuilderColumn>
<QueryBuilderColumn Field="HireDate" Label="Hire Date"
Type="ColumnType.Date" Format="MM/dd/yyyy"></QueryBuilderColumn>
<QueryBuilderColumn Field="Country" Label="Country"
Type="ColumnType.String"></QueryBuilderColumn>
</QueryBuilderColumns>
</SfQueryBuilder>
<SfButton @onclick="addRule" IsPrimary="true" Content="Add
Rules"></SfButton>
```

```

<SfButton @onclick="addGroup" IsPrimary="true" Content="Add
Group"></SfButton>
<SfButton @onclick="deleteGroups" IsPrimary="true" Content="Delete
Groups"></SfButton>
@code {
SfQueryBuilder<EmployeeDetails> QuerybuilderObj;
List<RuleModel> Rules = new List<RuleModel>()
{
new RuleModel { Field="Country", Label="Country", Type="String",
Operator="equal", Value = "England" },
new RuleModel { Field="EmployeeID", Label="EmployeeID", Type="Number",
Operator="notequal", Value = 1001 }
};
RuleModel SampRule = new RuleModel()
{
Label = "Employee ID",
Field = "EmployeeID",
Type = "Number",
Operator = "equal",
Value = 1091
};
RuleModel SampGroup = new RuleModel()
{
Condition = "or",
Rules = new List<RuleModel>()
{
new RuleModel()
{
Label = "Employee ID",
Field = "EmployeeID",
Type = "Number",
Operator = "equal",
Value = 1091
}
}
};
public string[] GroupID = new string[] { "group1" };
private void addRule()
{
QueryBuilderObj.AddRule(SampRule, "group0");
}
private void addGroup()
{
QueryBuilderObj.AddGroup(SampGroup, "group0");
}
private void deleteGroups()
{
QueryBuilderObj.DeleteGroup("group1");
}
public class EmployeeDetails
{
public int EmployeeID { get; set; }
public string FirstName { get; set; }
public bool TitleOfCourtesy { get; set; }
public string Title { get; set; }
public DateTime HireDate { get; set; }
public string Country { get; set; }
}

```

```
public string City { get; set; }
}
```

The screenshot shows a Blazor QueryBuilder interface. At the top, there are buttons for 'AND', 'OR', and '+'. Below this, the first rule is 'Employee ID' with the operator 'Not Equal' and the value '1,001'. The second rule is 'Country' with the operator 'Equal' and the value 'England'. These two rules are connected by an 'AND' operator. Below this, there is a second group of rules. The first rule in this group is 'Employee ID' with the operator 'Equal' and the value '0'. The second rule is 'Employee ID' with the operator 'Equal' and the value '1,091'. These two rules are also connected by an 'AND' operator. At the bottom, there are three buttons: 'ADD RULES', 'ADD GROUP', and 'DELETE GROUPS'.

You can also explore our [Blazor Query Builder example](#) to know how to render and configure the query builder.

Templates in Blazor QueryBuilder Component

This section explains the list of templates available in the [Blazor Query Builder](#).

Value Template

The template allows to define our own widgets for column values. Use the [ValueTemplate](#), property to define templates.

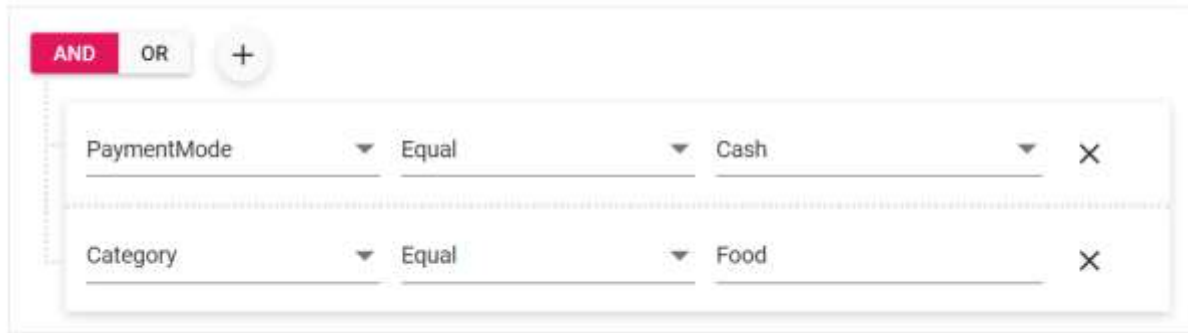
ASPX-CS

```
@using Syncfusion.Blazor.QueryBuilder
@using Syncfusion.Blazor.DropDowns
<div class="col-lg-12 control-section">
<SfQueryBuilder TValue="ExpenseData" Width="100%" DataSource="@DataSource">
<QueryBuilderRule Condition="and" Rules="@Rules"></QueryBuilderRule>
<QueryBuilderColumns>
<QueryBuilderColumn Field="PaymentMode" Label="PaymentMode"
Type=Syncfusion.Blazor.QueryBuilder.ColumnType.String>
<QueryBuilderTemplates>
<ValueTemplate>
<SfDropDownList TValue="string" TItem="ItemFields" DataSource="@Items"
@bind-Value="@DefaultValue">
<DropDownListFieldSettings Text="Id"></DropDownListFieldSettings>
<DropDownListEvents TItem="ItemFields" TValue="string" ValueChange="e =>
OnChange(e, context)"></DropDownListEvents>
</SfDropDownList>
</ValueTemplate>
</QueryBuilderTemplates>
</QueryBuilderColumn>
<QueryBuilderColumn Field="Category" Label="Category"
Type=Syncfusion.Blazor.QueryBuilder.ColumnType.String></QueryBuilderColumn>
```

```

<QueryBuilderColumn Field="Description" Label="Description"
Type=ColumnType.String></QueryBuilderColumn>
<QueryBuilderColumn Field="Amount" Label="Amount"
Type=Syncfusion.Blazor.QueryBuilder.ColumnType.Number></QueryBuilderColumn>
</QueryBuilderColumns>
</SfQueryBuilder>
</div>
@code{
public string DefaultValue = "Cash";
public List<ExpenseData> DataSource;
public class ExpenseData
{
public string Category { get; set; }
public string PaymentMode { get; set; }
public string Description { get; set; }
public int Amount { get; set; }
}
protected override void OnInitialized()
{
DataSource = new List<ExpenseData>()
{
new ExpenseData() {Category= "Food", PaymentMode="Credit Card",
Description="Boiled peanuts", Amount=100 },
new ExpenseData() {Category= "Food", PaymentMode="Debit Card",
Description="Boiled peanuts", Amount=200 },
new ExpenseData() {Category= "Food", PaymentMode="Cash",
Description="Confederate cash", Amount=300 },
new ExpenseData() {Category= "Transportation", PaymentMode="Cash",
Description="Public and other transportation", Amount=150 },
new ExpenseData() {Category= "Transportation", PaymentMode="Debit Card",
Description="Public and other transportation", Amount=250 }
};
}
public class ItemFields
{
public string Id { get; set; }
}
public List<ItemFields> Items = new List<ItemFields>() {
new ItemFields(){ Id= "Cash" },
new ItemFields(){ Id= "Debit Card" },
new ItemFields(){ Id= "Credit Card" },
new ItemFields(){ Id= "Net Banking" }
};
public void OnChange(Syncfusion.Blazor.DropDowns.ChangeEventArgs<string,
ItemFields> args, RuleModel Rule)
{
Rule.Value = args.Value;
}
List<RuleModel> Rules = new List<RuleModel>()
{
new RuleModel { Label="PaymentMode", Field="PaymentMode", Type="String",
Operator="equal", Value="Debit Card" },
new RuleModel { Label="Category", Field="Category", Type="String",
Operator="equal", Value="Food" }
};
}

```

Column Template

Column Template allows to define our own widgets for entire column. Use the [ColumnTemplate](#) property to define templates.

In the following sample, the `SfDropDownList` component is used as the custom components for field and value in the `PaymentMode` column.

ASPX-CS

```
@using Syncfusion.Blazor.QueryBuilder
@using Syncfusion.Blazor.DropDowns
@using Newtonsoft.Json
<div class="col-lg-12 control-section">
<SfQueryBuilder TValue="ExpenseData" @ref="qb" DataSource="@DataSource"
Width="75%">
<QueryBuilderEvents TValue="ExpenseData" Created="UpdateContent"
RuleChanged="UpdateContent"></QueryBuilderEvents>
<QueryBuilderRule Condition="and" Rules="@Rules"></QueryBuilderRule>
<QueryBuilderColumns>
<QueryBuilderColumn Field="PaymentMode" Label="PaymentMode"
Type=Syncfusion.Blazor.QueryBuilder.ColumnType.String>
<QueryBuilderTemplates>
<ColumnTemplate>
<div class="e-field">
<SfDropDownList TItem="string" TValue="string"
DataSource="qb.Columns.Select(e => e.Field)" @bind-Value="context.Label">
<DropDownListFieldSettings Text="PaymentMode"></DropDownListFieldSettings>
<DropDownListEvents TItem="string" TValue="string" Created="e =>
ValueCreated(context)" ValueChange="e => ChangeField(e,
context)"></DropDownListEvents>
</SfDropDownList>
</div>
<div class="e-value-field">
<SfDropDownList TValue="string" TItem="ItemFields" DataSource="@Items"
@bind-Value="@DefaultValue">
<DropDownListFieldSettings Text="Id"></DropDownListFieldSettings>
<DropDownListEvents TItem="ItemFields" TValue="string" ValueChange="e =>
ChangeValue(e, context)"></DropDownListEvents>
</SfDropDownList>
</div>
</ColumnTemplate>
</QueryBuilderTemplates>
</QueryBuilderColumn>
</QueryBuilderColumns>
</SfQueryBuilder>
</div>
```

```

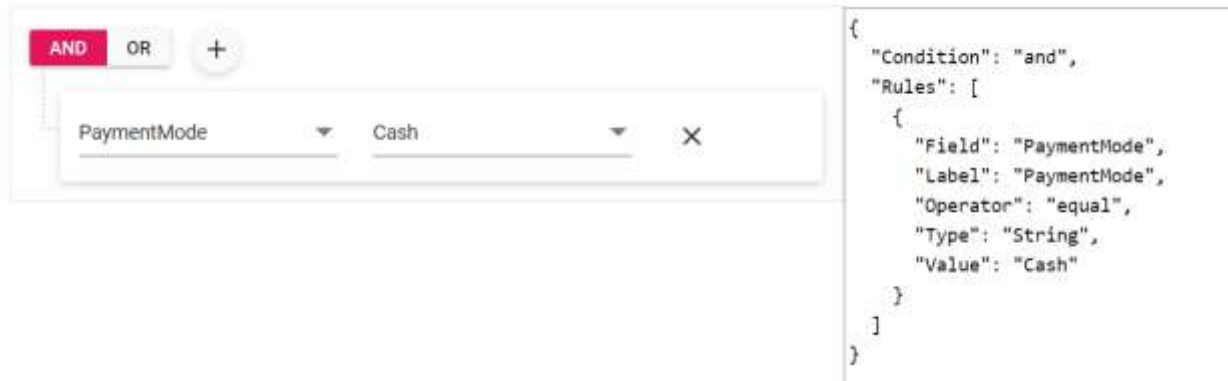
</QueryBuilderColumn>
<QueryBuilderColumn Field="Category" Label="Category"
Type=Syncfusion.Blazor.QueryBuilder.ColumnType.String></QueryBuilderColumn>
<QueryBuilderColumn Field="Description" Label="Description"
Type=ColumnType.String></QueryBuilderColumn>
<QueryBuilderColumn Field="Amount" Label="Amount"
Type=Syncfusion.Blazor.QueryBuilder.ColumnType.Number></QueryBuilderColumn>
</QueryBuilderColumns>
</SfQueryBuilder>
</div>
<table class="col-lg-3">
<tbody>
<tr>
<td colspan="2">
<textarea id="table" readonly>@content</textarea>
</td>
</tr>
</tbody>
</table>
@code {
SfQueryBuilder<ExpenseData> qb;
private string content;
public List<ExpenseData> DataSource;
public string DefaultValue = "Cash";
public class ItemFields
{
public string Id { get; set; }
}
public List<ItemFields> Items = new List<ItemFields>() {
new ItemFields(){ Id= "Cash" },
new ItemFields(){ Id= "Debit Card" },
new ItemFields(){ Id= "Credit Card" },
new ItemFields(){ Id= "Net Banking" }
};
public void UpdateContent()
{
content = JsonConvert.SerializeObject(qb.GetValidRules(),
Formatting.Indented, new JsonSerializerSettings
{
NullValueHandling = NullValueHandling.Ignore
});
}
public void ValueCreated(RuleModel Rule)
{
Rule.Value = "Cash";
UpdateContent();
}
public void ChangeValue(Syncfusion.Blazor.DropDowns.ChangeEventArgs<string,
ItemFields> args, RuleModel Rule)
{
Rule.Value = args.Value;
UpdateContent();
}
private void ChangeField(Syncfusion.Blazor.DropDowns.ChangeEventArgs<string,
string> args, RuleModel Rule)
{
Rule.Field = args.Value;
}
}

```

```

Rule.Label = args.Value;
Rule.Operator = "equal";
UpdateContent();
}
public class ExpenseData
{
    public string Category { get; set; }
    public string PaymentMode { get; set; }
    public string Description { get; set; }
    public int Amount { get; set; }
}
protected override void OnInitialized()
{
    DataSource = new List<ExpenseData>()
    {
        new ExpenseData() {Category= "Food", PaymentMode="Credit Card",
        Description="Boiled peanuts", Amount=100 },
        new ExpenseData() {Category= "Food", PaymentMode="Debit Card",
        Description="Boiled peanuts", Amount=200 },
        new ExpenseData() {Category= "Food", PaymentMode="Cash",
        Description="Confederate cash", Amount=300 },
        new ExpenseData() {Category= "Transportation", PaymentMode="Cash",
        Description="Public and other transportation", Amount=150 },
        new ExpenseData() {Category= "Transportation", PaymentMode="Debit Card",
        Description="Public and other transportation", Amount=250 }
    };
}
List<RuleModel> Rules = new List<RuleModel>()
{
    new RuleModel { Label="PaymentMode", Field="PaymentMode", Type="String",
    Operator="equal", Value="Cash" }
};
}
<style>
.e-query-builder {
margin: 0 auto;
float: left;
}
.e-rule-field .e-field,
.e-rule-field .e-value-field {
display: inline-block;
}
.e-rule-field .e-value-field {
padding-left: 20px;
}
#table {
width: 110%;
height: 500px;
border: 1px solid grey;
overflow: auto;
font-family: monospace;
}
</style>

```



Header Template

Header Template allows to define our own widgets for group header. Customize the AND/OR/NOT operators and add rules/groups buttons. Use the [HeaderTemplate](#) property to define templates.

In the following sample, the SFDropDownList component is used as the custom components for AND/OR operators and customized SfButtons for add rules and add groups button.

ASPX-CS

```
@using Syncfusion.Blazor.QueryBuilder
@using Syncfusion.Blazor.DropDowns
@using Syncfusion.Blazor.Buttons
<SfQueryBuilder TValue="EmployeeDetails" @ref="qbObj">
  <QueryBuilderRule Condition="and" Rules="Rules"></QueryBuilderRule>
  <QueryBuilderTemplates>
    <HeaderTemplate>
      <SfDropDownList ID="@context.ID" TValue="string" TItem="Operator"
        CssClass="e-outline" Width="100px" DataSource="@operData" @ref="ddlObj">
        <DropDownListFieldSettings Text="Text"></DropDownListFieldSettings>
        <DropDownListEvents TValue="string" TItem="Operator" Created="e =>
          ddlCreated(e, context)" ValueChange="e => onChange(e,
            context)"></DropDownListEvents>
      </SfDropDownList>
      <div id="addgrp">
        <SfButton Content="Add Group" CssClass="e-small e-outline" @onclick="e =>
          addGrpBtn(e, context)"></SfButton>
      </div>
      <div id="addgrp">
        <SfButton Content="Add Condition" CssClass="e-small e-outline" @onclick="e =>
          addCondBtn(e, context)"></SfButton>
      </div>
      @if (context.ID.Split("_")[1].IndexOf("0") < 0)
      {
        <div id="addgrp">
          <SfButton Content="Delete Group" CssClass="e-small e-outline e-danger"
            @onclick="e => deleteGrpBtn(e, context)"></SfButton>
        </div>
      }
    </HeaderTemplate>
  </QueryBuilderTemplates>
  <QueryBuilderColumns>
    <QueryBuilderColumn Field="EmployeeID" Label="Employee ID"
      Type="ColumnType.Number"></QueryBuilderColumn>
```

```

<QueryBuilderColumn Field="FirstName" Label="First Name"
Type="ColumnType.String"></QueryBuilderColumn>
<QueryBuilderColumn Field="TitleOfCourtesy" Label="Title of Courtesy"
Type="ColumnType.Boolean" Values="Values"></QueryBuilderColumn>
<QueryBuilderColumn Field="Title" Label="Title"
Type="ColumnType.String"></QueryBuilderColumn>
<QueryBuilderColumn Field="HireDate" Label="Hire Date"
Type="ColumnType.Date"></QueryBuilderColumn>
<QueryBuilderColumn Field="Country" Label="Country"
Type="ColumnType.String"></QueryBuilderColumn>
<QueryBuilderColumn Field="City" Label="City"
Type="ColumnType.String"></QueryBuilderColumn>
</QueryBuilderColumns>
</SfQueryBuilder>
@code {
SfDropDownList<string, Operator> ddlObj { set { ddlRefs.Add(value); } }
SfQueryBuilder<EmployeeDetails> qbObj;
List<SfDropDownList<string, Operator>> ddlRefs = new
List<SfDropDownList<string, Operator>>();
private string[] Values = new string[] { "Mr.", "Mrs." };
private string grpID;
List<RuleModel> Rules = new List<RuleModel>()
{
new RuleModel { Field="Country", Label="Country", Type="String",
Operator="equal", Value = "England" },
new RuleModel { Field="EmployeeID", Label="EmployeeID", Type="Number",
Operator="notequal", Value = 1001 },
new RuleModel { Condition="or", Rules=new List<RuleModel>() { new
RuleModel() { Field="FirstName", Label="First Name", Type = "String",
Operator = "equal", Value="Mark" } }}
};
public class Operator
{
public string Text { get; set; }
}
public class EmployeeDetails
{
public int EmployeeID { get; set; }
public string FirstName { get; set; }
public bool TitleOfCourtesy { get; set; }
public string Title { get; set; }
public DateTime HireDate { get; set; }
public string Country { get; set; }
public string City { get; set; }
}
List<Operator> operData = new List<Operator>()
{
new Operator() { Text = "AND" },
new Operator() { Text = "OR" }
};
private void ddlCreated(Object obj, HeaderTemplateModel model)
{
foreach (SfDropDownList<string, Operator> ddlObj in ddlRefs)
{
if (ddlObj.ID == model.ID)
{
ddlObj.Value = model.Condition.ToUpper();
}
}
}
}

```

```

}
}
}
private void onChange(ChangeEventArgs<string, Operator> args,
HeaderTemplateModel model)
{
if (args.IsInteracted)
{
model.Condition = args.Value.ToLower();
}
}
private void addGrpBtn(MouseEventArgs args, HeaderTemplateModel model)
{
grpID = model.ID.Split("_")[1];
qbObj.AddGroup(new RuleModel { Condition = "or", Rules = new
List<RuleModel>() { new RuleModel() } }, grpID);
}
private void addCondBtn(MouseEventArgs args, HeaderTemplateModel model)
{
grpID = model.ID.Split("_")[1];
qbObj.AddRule(new RuleModel(), grpID);
}
private void deleteGrpBtn(MouseEventArgs args, HeaderTemplateModel model)
{
grpID = model.ID.Split("_")[1];
qbObj.DeleteGroup(grpID);
}
}
<style>
#addgrp {
display: inline-block;
padding: 0 0 0 12px;
}
</style>

```

The screenshot displays the Blazor Query Builder interface. It features two main sections for building queries. The first section, under the 'AND' operator, contains two conditions: 'Country' Equal 'England' and 'Employee ID' Not Equal '1001'. The second section, under the 'OR' operator, contains one condition: 'First Name' Equal 'Mark'. Each condition is represented by a row with dropdowns for the field name, the comparison operator, and the value, followed by a delete button (X). The 'OR' section also includes a 'DELETE GROUP' button. The interface is clean and uses a light gray color scheme.

You can also explore our [Blazor Query Builder example](#) to know how to render and configure the query builder.

Importing and Exporting in Blazor QueryBuilder Component

Importing allows to view or edit the predefined conditions which is available in List rules or SQL rules. You can import the conditions as either initial rendering or post rendering.

Initial rendering

To apply the conditions initially, you can define the condition and rules in **QueryBuilderRule** . Here, the list of rules can be imported by defining the **QueryBuilderRule**.

ASPX-CS

```
@using Syncfusion.Blazor.QueryBuilder
<SfQueryBuilder DataSource="@EmployeeData" @ref="QueryBuilderObj">
  <QueryBuilderRule Condition="or" Rules="Rules"></QueryBuilderRule>
  <QueryBuilderColumns>
    <QueryBuilderColumn Field="EmployeeID" Label="Employee ID"
      Type="ColumnType.Number"></QueryBuilderColumn>
    <QueryBuilderColumn Field="FirstName" Label="First Name"
      Type="ColumnType.String"></QueryBuilderColumn>
    <QueryBuilderColumn Field="TitleOfCourtesy" Label="Title Of Courtesy"
      Type="ColumnType.Boolean"></QueryBuilderColumn>
    <QueryBuilderColumn Field="HireDate" Label="Hire Date"
      Type="ColumnType.Date"></QueryBuilderColumn>
    <QueryBuilderColumn Field="Country" Label="Country"
      Type="ColumnType.String"></QueryBuilderColumn>
    <QueryBuilderColumn Field="City" Label="City"
      Type="ColumnType.String"></QueryBuilderColumn>
  </QueryBuilderColumns>
</SfQueryBuilder>
@code {
  SfQueryBuilder<EmployeeDetails> QueryBuilderObj;
  List<RuleModel> Rules = new List<RuleModel>()
  {
    new RuleModel { Field="Country", Label="Country", Type="String",
      Operator="equal", Value = "England" },
    new RuleModel { Field="EmployeeID", Label="EmployeeID", Type="Number",
      Operator="notequal", Value = 1001 }
  };
  public List<EmployeeDetails> EmployeeData = new List<EmployeeDetails>
  {
    new EmployeeDetails{ FirstName = "Martin", EmployeeID = 1001, Country =
      "England", City = "Manchester", HireDate = new DateTime(2014, 4, 23) },
    new EmployeeDetails{ FirstName = "Benjamin", EmployeeID = 1002, Country =
      "England", City = "Birmingham", HireDate = new DateTime(2014, 6, 19) },
    new EmployeeDetails{ FirstName = "Stuart", EmployeeID = 1003, Country =
      "England", City = "London", HireDate = new DateTime(2014, 7, 4) },
    new EmployeeDetails{ FirstName = "Ben", EmployeeID = 1004, Country = "USA",
      City = "California", HireDate = new DateTime(2014, 8, 15) },
    new EmployeeDetails{ FirstName = "Joseph", EmployeeID = 1005, Country =
      "Spain", City = "Madrid", HireDate = new DateTime(2014, 8, 29) }
  };
  public class EmployeeDetails
  {
    public int EmployeeID { get; set; }
    public string FirstName { get; set; }
    public bool TitleOfCourtesy { get; set; }
    public string Title { get; set; }
  }
}
```

```
public DateTime HireDate { get; set; }
public string Country { get; set; }
public string City { get; set; }
}
```



Post rendering

Importing from SQL

You can set the conditions from SQL query through the [SetRulesFromSql](#) method.

ASPX-CS

```
@using Syncfusion.Blazor.QueryBuilder
@using Syncfusion.Blazor.Buttons
<SfQueryBuilder DataSource="@EmployeeData" @ref="QueryBuilderObj">
  <QueryBuilderColumns>
    <QueryBuilderColumn Field="EmployeeID" Label="Employee ID"
      Type="ColumnType.Number"></QueryBuilderColumn>
    <QueryBuilderColumn Field="FirstName" Label="First Name"
      Type="ColumnType.String"></QueryBuilderColumn>
    <QueryBuilderColumn Field="TitleOfCourtesy" Label="Title Of Courtesy"
      Type="ColumnType.Boolean"></QueryBuilderColumn>
    <QueryBuilderColumn Field="HireDate" Label="Hire Date"
      Type="ColumnType.Date"></QueryBuilderColumn>
    <QueryBuilderColumn Field="Country" Label="Country"
      Type="ColumnType.String"></QueryBuilderColumn>
    <QueryBuilderColumn Field="City" Label="City"
      Type="ColumnType.String"></QueryBuilderColumn>
  </QueryBuilderColumns>
</SfQueryBuilder>
<SfButton CssClass="e-primary" @onclick="setRules">Set Rules</SfButton>
@code {
  SfQueryBuilder<EmployeeDetails> QueryBuilderObj;
  public List<EmployeeDetails> EmployeeData = new List<EmployeeDetails>
  {
    new EmployeeDetails{ FirstName = "Martin", EmployeeID = 1001, Country =
      "England", City = "Manchester", HireDate = new DateTime(2014, 4, 23) },
    new EmployeeDetails{ FirstName = "Benjamin", EmployeeID = 1002, Country =
      "England", City = "Birmingham", HireDate = new DateTime(2014, 6, 19) },
    new EmployeeDetails{ FirstName = "Stuart", EmployeeID = 1003, Country =
      "England", City = "London", HireDate = new DateTime(2014, 7, 4) },
    new EmployeeDetails{ FirstName = "Ben", EmployeeID = 1004, Country = "USA",
      City = "California", HireDate = new DateTime(2014, 8, 15) },
  }
```



```

new EmployeeDetails{ FirstName = "Joseph", EmployeeID = 1005, Country =
"Spain", City = "Madrid", HireDate = new DateTime(2014, 8, 29) }
};
public class EmployeeDetails
{
public int EmployeeID { get; set; }
public string FirstName { get; set; }
public bool TitleOfCourtesy { get; set; }
public string Title { get; set; }
public DateTime HireDate { get; set; }
public string Country { get; set; }
public string City { get; set; }
}
private void setRules()
{
QueryBuilderObj.SetRulesFromSql("EmployeeID = 1001 AND City LIKE
('Manchester%')");
}
}

```

The screenshot shows a web-based query builder interface. At the top, there are three buttons: 'AND' (highlighted in red), 'OR', and a '+' icon. Below these are two rule rows. The first row has a dropdown for 'Employee ID', an operator dropdown set to 'Equal', a text input with '1,001', and a delete 'X' button. The second row has a dropdown for 'City', an operator dropdown set to 'Starts With', a text input with 'Manchester', and a delete 'X' button. At the bottom left, there is a red button labeled 'SET RULES'.

Exporting

Exporting allows to save or maintain the created conditions through the [Blazor Query Builder](#). You can export the defined conditions by the following ways.

Exporting to SQL

The defined conditions can be exported to the SQL query through the [GetSqlFromRules](#) method.

ASPX-CS

```

@using Syncfusion.Blazor.QueryBuilder
@using Syncfusion.Blazor.Buttons
<SfQueryBuilder DataSource="@EmployeeDetails" @ref="QueryBuilderObj">
<QueryBuilderColumns>
<QueryBuilderColumn Field="EmployeeID" Label="Employee ID"
Type="ColumnType.Number"></QueryBuilderColumn>
<QueryBuilderColumn Field="FirstName" Label="First Name"
Type="ColumnType.String"></QueryBuilderColumn>
<QueryBuilderColumn Field="TitleOfCourtesy" Label="Title Of Courtesy"
Type="ColumnType.Boolean"></QueryBuilderColumn>
<QueryBuilderColumn Field="HireDate" Label="Hire Date"
Type="ColumnType.Date"></QueryBuilderColumn>
<QueryBuilderColumn Field="Country" Label="Country"
Type="ColumnType.String"></QueryBuilderColumn>

```

```

<QueryBuilderColumn Field="City" Label="City"
Type="ColumnType.String"></QueryBuilderColumn>
</QueryBuilderColumns>
<QueryBuilderRule Condition="or" Rules="Rules"></QueryBuilderRule>
</SfQueryBuilder>
<SfButton CssClass="e-primary" @onclick="getSql">Get SQL</SfButton>
@code {
SfQueryBuilder<Employee> QueryBuilderObj;
public List<Employee> EmployeeDetails = new List<Employee>
{
new Employee{ FirstName = "Martin", EmployeeID = "1001", Country =
"England", City = "Manchester", HireDate = "23/04/2014" },
new Employee{ FirstName = "Benjamin", EmployeeID = "1002", Country =
"England", City = "Birmingham", HireDate = "19/06/2014" },
new Employee{ FirstName = "Stuart", EmployeeID = "1003", Country =
"England", City = "London", HireDate = "04/07/2014"},
new Employee{ FirstName = "Ben", EmployeeID = "1004", Country = "USA", City
= "California", HireDate = "15/08/2014" },
new Employee{ FirstName = "Joseph", EmployeeID = "1005", Country = "Spain",
City = "Madrid", HireDate = "29/08/2014" }
};
public class Employee {
public string FirstName { get; set; }
public string EmployeeID { get; set; }
public string Country { get; set; }
public string City { get; set; }
public string HireDate { get; set; }
}
public List<RuleModel> Rules = new List<RuleModel>
{
new RuleModel { Field = "EmployeeID", Type="Number", Value = 1001, Operator
= "notequal" },
new RuleModel { Field = "Country", Type="String", Value = "England",
Operator = "equal" }
};
private void getSql()
{
QueryBuilderObj.GetSqlFromRules();
}
}

```

You can also explore our [Blazor Query Builder example](#) to know how to render and configure the query builder.

Localization in Blazor QueryBuilder Component

The **Localization** library allows to localize default text content of the Query Builder. The [Blazor Query Builder](#) has static text that can be changed to other cultures (Arabic, Deutsch, French, etc.). In the following sample the text is changed to **German** culture.

Resource file (.resx) is used to translate the static text of the query builder.

Use **Resource** file to translate the static text of the QueryBuilder. The Resource file is an XML file which contains the strings (key and value pairs) that has to be translated into different language. Refer [Localization](#) link to know more about how to configure and use localization in the Blazor Server and Web Assembly project for Syncfusion Blazor components.

The following list of properties and its values are used in the Query Builder.

Locale key	en-US (default)
QueryBuilder_AddGroup	Add Group
QueryBuilder_AddCondition	Add Condition
QueryBuilder_DeleteRule	Remove this condition
QueryBuilder_DeleteGroup	Delete group
QueryBuilder_Edit	EDIT
QueryBuilder_SelectField	Select a field
QueryBuilder_SelectOperator	Select operator
QueryBuilder_StartsWith	Starts With
QueryBuilder_EndsWith	Ends With
QueryBuilder_Contains	Contains
QueryBuilder_Equal	Equal
QueryBuilder_NotEqual	Not Equal
QueryBuilder_LessThan	Less Than
QueryBuilder_LessThanOrEqual	Less Than Or Equal
QueryBuilder_GreaterThan	Greater Than
QueryBuilder_GreaterThanOrEqual	Greater Than Or Equal
QueryBuilder_Between	Between
QueryBuilder_NotBetween	Not Between
QueryBuilder_In	In
QueryBuilder_NotIn	Not In
QueryBuilder_Remove	REMOVE
QueryBuilder_ValidationMessage	This field is required
QueryBuilder_SummaryViewTitle	Summary View
QueryBuilder_OtherFields	Other Fields
QueryBuilder_AND	AND
QueryBuilder_OR	OR
QueryBuilder_SelectValue	Enter Value
QueryBuilder_IsEmpty	Is Empty
QueryBuilder_IsNotEmpty	Is Not Empty
QueryBuilder_IsNull	Is Null

| QueryBuilder_IsNotNull | Is Not Null |

ASPX-CS

```
@using Syncfusion.Blazor.QueryBuilder
<SfQueryBuilder TValue="EmployeeDetails">
  <QueryBuilderRule Condition="and" Rules="@Rules"></QueryBuilderRule>
  <QueryBuilderColumns>
    <QueryBuilderColumn Field="EmployeeID" Label="Employee ID"
      Type="ColumnType.Number"></QueryBuilderColumn>
    <QueryBuilderColumn Field="FirstName" Label="First Name"
      Type="ColumnType.String"></QueryBuilderColumn>
    <QueryBuilderColumn Field="TitleOfCourtesy" Label="Title of Courtesy"
      Type="ColumnType.Boolean" Values="Values"></QueryBuilderColumn>
    <QueryBuilderColumn Field="Title" Label="Title"
      Type="ColumnType.String"></QueryBuilderColumn>
    <QueryBuilderColumn Field="HireDate" Label="Hire Date"
      Type="ColumnType.Date"></QueryBuilderColumn>
    <QueryBuilderColumn Field="Country" Label="Country"
      Type="ColumnType.String"></QueryBuilderColumn>
    <QueryBuilderColumn Field="City" Label="City"
      Type="ColumnType.String"></QueryBuilderColumn>
  </QueryBuilderColumns>
</SfQueryBuilder>
@code {
  private string[] Values = new string[] { "Mr.", "Mrs." };
  List<RuleModel> Rules = new List<RuleModel>()
  {
    new RuleModel { Field="EmployeeID", Label="Employee ID", Type="Number",
      Operator="equal", Value = 2345 }
  };
  public class EmployeeDetails
  {
    public int EmployeeID { get; set; }
    public string FirstName { get; set; }
    public bool TitleOfCourtesy { get; set; }
    public string Title { get; set; }
    public DateTime HireDate { get; set; }
    public string Country { get; set; }
    public string City { get; set; }
  }
}
```



You can also explore our [Blazor Query Builder example](#) to know how to render and configure the query builder.

Styles and Appearances in Blazor QueryBuilder Component

To modify the QueryBuilder appearance, you need to override the default CSS of QueryBuilder component. Please find the list of CSS classes and its corresponding section in QueryBuilder component. Also, you have an option to create your own custom theme for the controls using our [Theme Studio](#).

Class	Purpose of Class
-----	-----
<code>.e-query-builder .e-btn-group input + label.e-btn</code>	To customize the condition button in querybuilder.
<code>.e-query-builder .e-btn-group input:checked + label.e-btn</code>	To customize the selected condition button in querybuilder.
<code>.e-query-builder .e-group-body .e-rule-container</code>	To customize the querybuilder rule container.
<code>.e-query-builder .e-group-action .e-btn</code>	To customize the querybuilder add group/condition button.
<code>.e-query-builder .e-remove-rule.e-btn.e-round</code>	To customize the querybuilder Delete button.
<code>.e-query-builder .e-rule-list > ::after, .e-query-builder .e-rule-list > ::before</code>	To customize the querybuilder group joining line.
<code>.e-query-builder .e-rule-container.e-joined-rule</code>	To customize the querybuilder condition joining line.

How To

Blazor QueryBuilder Component in WebAssembly App using Visual Studio

This article provides step-by-step instructions to configure Syncfusion [Blazor Query Builder](#) in a simple Blazor WebAssembly application using [Visual Studio 2019](#).

Starting with version 17.4.0.39 (2019 Volume 4), you need to include a valid license key (either paid or trial key) within your applications. Please refer to this [help topic](#) for more information.

Prerequisites

- [Visual Studio 2019](#)
- [.NET Core SDK 3.1.3](#)

.NET Core SDK 3.1.3 requires Visual Studio 2019 16.6 or later. Syncfusion Blazor components are compatible with .NET Core 5.0 Preview 6 and it requires Visual Studio 16.7 Preview 1 or later.

Create a Blazor WebAssembly project in Visual Studio 2019

1. Install the essential project templates in the Visual Studio 2019 by running the below command line in the command prompt.

ASPX-CS

```
@using Syncfusion.Blazor
@using Syncfusion.Blazor.QueryBuilder.
```

5. Open the `~/Program.cs` file and register the Syncfusion Blazor Service.

C#

```
using Syncfusion.Blazor;  
namespace WebApplication1  
{  
    public class Program  
    {  
        public static async Task Main(string[] args)  
        {  
            ....  
            ....  
            builder.Services.AddSyncfusionBlazor();  
            await builder.Build().RunAsync();  
        }  
    }  
}
```

6. Add the Syncfusion bootstrap4 theme in the element of the ~/wwwroot/index.html page.

HTML

```
<head>  
    ....  
    ....  
    <link href="_content/Syncfusion.Blazor/styles/bootstrap4.css"  
    rel="stylesheet" />  
</head>
```

The same theme file can be referred through the CDN version by using

<https://cdn.syncfusion.com/blazor/{{ site.blazorversion }}/styles/bootstrap4.css>.

To use manual scripts other than the scripts from NuGet package, register the Blazor service in ~/Program.cs file by using true parameter as mentioned below.

C#

```
using Syncfusion.Blazor;  
namespace WebApplication1  
{  
    public class Program  
    {  
        public static async Task Main(string[] args)  
        {  
            ....  
            ....  
            builder.Services.AddSyncfusionBlazor(true);  
            await builder.Build().RunAsync();  
        }  
    }  
}
```

Adding Query Builder component to the application

Now, add the [Blazor Query Builder](#) component in razor page in the **Pages** folder. For example, the Query Builder component is added in the `~/Pages/Index.razor` page.

ASPX-CS

```
@using Syncfusion.Blazor.QueryBuilder
<SfQueryBuilder TValue="EmployeeDetails">
  <QueryBuilderColumns>
    <QueryBuilderColumn Field="EmployeeID" Label="Employee ID"
      Type="ColumnType.Number"></QueryBuilderColumn>
    <QueryBuilderColumn Field="FirstName" Label="First Name"
      Type="ColumnType.String"></QueryBuilderColumn>
    <QueryBuilderColumn Field="TitleOfCourtesy" Label="Title of Courtesy"
      Type="ColumnType.Boolean" Values="Values"></QueryBuilderColumn>
    <QueryBuilderColumn Field="Title" Label="Title"
      Type="ColumnType.String"></QueryBuilderColumn>
    <QueryBuilderColumn Field="HireDate" Label="Hire Date"
      Type="ColumnType.Date"></QueryBuilderColumn>
    <QueryBuilderColumn Field="Country" Label="Country"
      Type="ColumnType.String"></QueryBuilderColumn>
    <QueryBuilderColumn Field="City" Label="City"
      Type="ColumnType.String"></QueryBuilderColumn>
  </QueryBuilderColumns>
</SfQueryBuilder>
@code {
  private string[] Values = new string[] { "Mr.", "Mrs." };
  public class EmployeeDetails
  {
    public int EmployeeID { get; set; }
    public string FirstName { get; set; }
    public bool TitleOfCourtesy { get; set; }
    public string Title { get; set; }
    public DateTime HireDate { get; set; }
    public string Country { get; set; }
    public string City { get; set; }
  }
}
```

Run the application

After successful compilation of the application, simply press F5 to run the application. The [Blazor Query Builder](#) component will render in the web browser.

Change Display Mode in Blazor QueryBuilder Component

The QueryBuilder allows to view Vertically or Horizontally. To enable this feature, you can set the [DisplayMode](#) property.

ASPX-CS

```
@using Syncfusion.Blazor.QueryBuilder
<SfQueryBuilder DataSource="@EmployeeData" Width="400px"
DisplayMode="Syncfusion.Blazor.QueryBuilder.DisplayMode.Vertical">
<QueryBuilderRule Condition="and" Rules="Rules"></QueryBuilderRule>
<QueryBuilderColumns>
<QueryBuilderColumn Field="EmployeeID" Label="Employee ID"
Type="ColumnType.Number"></QueryBuilderColumn>
<QueryBuilderColumn Field="FirstName" Label="First Name"
Type="ColumnType.String"></QueryBuilderColumn>
<QueryBuilderColumn Field="TitleOfCourtesy" Label="Title Of Courtesy"
Type="ColumnType.Boolean"></QueryBuilderColumn>
<QueryBuilderColumn Field="HireDate" Label="Hire Date"
Type="ColumnType.Date"></QueryBuilderColumn>
<QueryBuilderColumn Field="Country" Label="Country"
Type="ColumnType.String"></QueryBuilderColumn>
<QueryBuilderColumn Field="City" Label="City"
Type="ColumnType.String"></QueryBuilderColumn>
</QueryBuilderColumns>
</SfQueryBuilder>
@code {
List<RuleModel> Rules = new List<RuleModel>()
{
new RuleModel() { Field="HireDate", Label="Hire Date", Type="Date",
Operator="greaterthan", Value=new DateTime(2019, 7, 10)}
};
//Local datasource
public List<EmployeeDetails> EmployeeData = new List<EmployeeDetails>
{
```



```

new EmployeeDetails{ FirstName = "Martin", EmployeeID = 1001, Country =
"England", City = "Manchester", HireDate = new DateTime(2014, 4, 23) },
new EmployeeDetails{ FirstName = "Benjamin", EmployeeID = 1002, Country =
"England", City = "Birmingham", HireDate = new DateTime(2014, 6, 19) },
new EmployeeDetails{ FirstName = "Stuart", EmployeeID = 1003, Country =
"England", City = "London", HireDate = new DateTime(2014, 7, 4) },
new EmployeeDetails{ FirstName = "Ben", EmployeeID = 1004, Country = "USA",
City = "California", HireDate = new DateTime(2014, 8, 15) },
new EmployeeDetails{ FirstName = "Joseph", EmployeeID = 1005, Country =
"Spain", City = "Madrid", HireDate = new DateTime(2014, 8, 29) }
};
public class EmployeeDetails
{
public string FirstName { get; set; }
public int EmployeeID { get; set; }
public string Country { get; set; }
public string City { get; set; }
public DateTime HireDate { get; set; }
}
}

```

The default view in the desktop mode is Horizontal. The default view in the mobile mode is Vertical.

Show Not Operator in Blazor QueryBuilder Component

The QueryBuilder provides **Not** operator along with AND, OR operators to filter records based on more than one condition. You can enable this feature by setting the [EnableNotCondition](#) property to **true**.

By default **EnableNotCondition** set as false.

ASPX-CS

```

@using Syncfusion.Blazor.QueryBuilder
<SfQueryBuilder TValue="@EmployeeDetails" EnableNotCondition="true">
<QueryBuilderRule Condition="or" Not="false"
Rules="@Rules"></QueryBuilderRule>
<QueryBuilderColumns>
<QueryBuilderColumn Field="EmployeeID" Label="Employee ID"
Type="ColumnType.Number"></QueryBuilderColumn>

```

```

<QueryBuilderColumn Field="FirstName" Label="First Name"
Type="ColumnType.String"></QueryBuilderColumn>
<QueryBuilderColumn Field="HireDate" Label="Hire Date"
Type="ColumnType.Date"></QueryBuilderColumn>
<QueryBuilderColumn Field="Country" Label="Country"
Type="ColumnType.String"></QueryBuilderColumn>
</QueryBuilderColumns>
</SfQueryBuilder>
@code {
List<RuleModel> Rules = new List<RuleModel>()
{
    new RuleModel { Field="EmployeeID", Label="EmployeeID", Type="Number",
Operator="notequal", Value = 1001 },
    new RuleModel { Condition = "and", Not=true, Rules = new List<RuleModel>(){
        new RuleModel() { Field = "FirstName", Label = "First Name", Type =
"String", Operator = "contains", Value = "mark" } } }
};
public class EmployeeDetails
{
    public string FirstName { get; set; }
    public int EmployeeID { get; set; }
    public string Country { get; set; }
    public string City { get; set; }
    public DateTime HireDate { get; set; }
}
}

```

The screenshot shows a Blazor QueryBuilder interface. At the top, there are buttons for logical operators: NOT, AND, OR, and a plus sign (+). Below this, there are two rule boxes. The first rule box contains 'Employee ID' in a dropdown, 'Not Equal' in a dropdown, and '1001' in a text input. The second rule box contains 'First Name' in a dropdown, 'Contains' in a dropdown, and 'mark' in a text input. Between the two rule boxes, there is a button for the logical operator 'AND'. Each rule box has a close button (X) on the right.

Maintain the State Persistence in Blazor QueryBuilder Component

State persistence allows the QueryBuilder to retain the current QueryBuilder state in the browser local storage for state maintenance. This action is handled through the [EnablePersistence](#) property which is set to false by default. When it is set to true, the QueryBuilder **Rules** will be retained even after refreshing the page.

The state will be persisted based on ID property. So, it is recommended to explicitly set the ID property for QueryBuilder.

ASPX-CS

```
@using Syncfusion.Blazor.QueryBuilder
```

```

<SfQueryBuilder ID="querybuilder" DataSource="@EmployeeData"
EnablePersistence="true">
  <QueryBuilderRule Condition="or" Rules="@Rules"></QueryBuilderRule>
  <QueryBuilderColumns>
    <QueryBuilderColumn Field="EmployeeID" Label="Employee ID"
    Type="ColumnType.Number"></QueryBuilderColumn>
    <QueryBuilderColumn Field="FirstName" Label="First Name"
    Type="ColumnType.String"></QueryBuilderColumn>
    <QueryBuilderColumn Field="HireDate" Label="Hire Date"
    Type="ColumnType.Date"></QueryBuilderColumn>
    <QueryBuilderColumn Field="Country" Label="Country"
    Type="ColumnType.String"></QueryBuilderColumn>
  </QueryBuilderColumns>
</SfQueryBuilder>
@code {
List<RuleModel> Rules = new List<RuleModel>()
{
  new RuleModel { Field="Country", Label="Country", Type="String",
  Operator="equal", Value = "England" },
  new RuleModel { Field="EmployeeID", Label="EmployeeID", Type="Number",
  Operator="notequal", Value = 1001 }
};
public List<EmployeeDetails> EmployeeData = new List<EmployeeDetails>
{
  new EmployeeDetails{ FirstName = "Martin", EmployeeID = 1001, Country =
  "England", City = "Manchester", HireDate = new DateTime(2014, 4, 23) },
  new EmployeeDetails{ FirstName = "Benjamin", EmployeeID = 1002, Country =
  "England", City = "Birmingham", HireDate = new DateTime(2014, 6, 19) },
  new EmployeeDetails{ FirstName = "Stuart", EmployeeID = 1003, Country =
  "England", City = "London", HireDate = new DateTime(2014, 7, 4) },
  new EmployeeDetails{ FirstName = "Ben", EmployeeID = 1004, Country = "USA",
  City = "California", HireDate = new DateTime(2014, 8, 15) },
  new EmployeeDetails{ FirstName = "Joseph", EmployeeID = 1005, Country =
  "Spain", City = "Madrid", HireDate = new DateTime(2014, 8, 29) }
};
public class EmployeeDetails
{
  public string FirstName { get; set; }
  public int EmployeeID { get; set; }
  public string Country { get; set; }
  public string City { get; set; }
  public DateTime HireDate { get; set; }
}
}

```

Readonly in Blazor QueryBuilder Component

The readonly property in the query builder disables the user interactions on the component. You can enable this feature by setting the [Readonly](#) property to `true`.

ASPX-CS

```

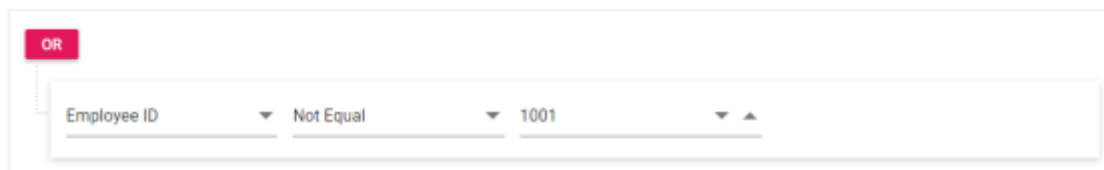
@using Syncfusion.Blazor.QueryBuilder
<SfQueryBuilder TValue="@EmployeeDetails" Readonly="true">
  <QueryBuilderRule Condition="or" Rules="@Rules"></QueryBuilderRule>
  <QueryBuilderColumns>

```

```

<QueryBuilderColumn Field="EmployeeID" Label="Employee ID"
Type="ColumnType.Number"></QueryBuilderColumn>
<QueryBuilderColumn Field="FirstName" Label="First Name"
Type="ColumnType.String"></QueryBuilderColumn>
<QueryBuilderColumn Field="HireDate" Label="Hire Date"
Type="ColumnType.Date"></QueryBuilderColumn>
<QueryBuilderColumn Field="Country" Label="Country"
Type="ColumnType.String"></QueryBuilderColumn>
</QueryBuilderColumns>
</SfQueryBuilder>
@code {
List<RuleModel> Rules = new List<RuleModel>()
{
new RuleModel { Field="EmployeeID", Label="EmployeeID", Type="Number",
Operator="notequal", Value = 1001 }
};
public class EmployeeDetails
{
public string FirstName { get; set; }
public int EmployeeID { get; set; }
public string Country { get; set; }
public string City { get; set; }
public DateTime HireDate { get; set; }
}
}

```



Right to Left in Blazor QueryBuilder Component

RTL provides an option to switch the text direction and layout of the Query Builder 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 this feature, set the [EnableRtl](#) to true.

ASPX-CS

```

@using Syncfusion.Blazor.QueryBuilder
<SfQueryBuilder DataSource="@EmployeeData" EnableRtl="true">
<QueryBuilderRule Condition="or" Rules="@Rules"></QueryBuilderRule>
<QueryBuilderColumns>
<QueryBuilderColumn Field="EmployeeID" Label="Employee ID"
Type="ColumnType.Number"></QueryBuilderColumn>
<QueryBuilderColumn Field="FirstName" Label="First Name"
Type="ColumnType.String"></QueryBuilderColumn>
<QueryBuilderColumn Field="HireDate" Label="Hire Date"
Type="ColumnType.Date"></QueryBuilderColumn>
<QueryBuilderColumn Field="Country" Label="Country"
Type="ColumnType.String"></QueryBuilderColumn>
</QueryBuilderColumns>

```

```

</SfQueryBuilder>
@code {
List<RuleModel> Rules = new List<RuleModel>()
{
    new RuleModel { Field="Country", Label="Country", Type="String",
    Operator="equal", Value = "England" },
    new RuleModel { Field="EmployeeID", Label="EmployeeID", Type="Number",
    Operator="notequal", Value = 1001 }
};
public List<EmployeeDetails> EmployeeData = new List<EmployeeDetails>
{
    new EmployeeDetails{ FirstName = "Martin", EmployeeID = 1001, Country =
    "England", City = "Manchester", HireDate = new DateTime(2014, 4, 23) },
    new EmployeeDetails{ FirstName = "Benjamin", EmployeeID = 1002, Country =
    "England", City = "Birmingham", HireDate = new DateTime(2014, 6, 19) },
    new EmployeeDetails{ FirstName = "Stuart", EmployeeID = 1003, Country =
    "England", City = "London", HireDate = new DateTime(2014, 7, 4) },
    new EmployeeDetails{ FirstName = "Ben", EmployeeID = 1004, Country = "USA",
    City = "California", HireDate = new DateTime(2014, 8, 15) },
    new EmployeeDetails{ FirstName = "Joseph", EmployeeID = 1005, Country =
    "Spain", City = "Madrid", HireDate = new DateTime(2014, 8, 29) }
};
public class EmployeeDetails
{
    public string FirstName { get; set; }
    public int EmployeeID { get; set; }
    public string Country { get; set; }
    public string City { get; set; }
    public DateTime HireDate { get; set; }
}
}

```

The screenshot shows a Blazor QueryBuilder interface. At the top right, there are buttons for adding a new rule (+), switching between OR and AND operators, and a red button labeled 'AND'. Below these, there are two filter rules displayed in a list. The first rule has a field 'Customerid', a value '1,001', and an operator 'Equal'. The second rule has a field 'Employeeid', a value 'FLKIN', and an operator 'Equal'. Each rule has a close button (X) on the left. The rules are connected by an 'AND' operator.

Restrict the Groups in Blazor QueryBuilder Component

The QueryBuilder allows to restrict the groups from creation based on group count. You can enable this feature by setting the [MaxGroupCount](#) property.

By default, MaxGroupCount is set as 5.

ASPX-CS

```
@using Syncfusion.Blazor.QueryBuilder
```

```

<SfQueryBuilder DataSource="@EmployeeData" MaxGroupCount="2">
  <QueryBuilderRule Condition="or" Rules="@Rules"></QueryBuilderRule>
  <QueryBuilderColumns>
    <QueryBuilderColumn Field="EmployeeID" Label="Employee ID"
      Type="ColumnType.Number"></QueryBuilderColumn>
    <QueryBuilderColumn Field="FirstName" Label="First Name"
      Type="ColumnType.String"></QueryBuilderColumn>
    <QueryBuilderColumn Field="HireDate" Label="Hire Date"
      Type="ColumnType.Date"></QueryBuilderColumn>
    <QueryBuilderColumn Field="Country" Label="Country"
      Type="ColumnType.String"></QueryBuilderColumn>
  </QueryBuilderColumns>
</SfQueryBuilder>
@code {
List<RuleModel> Rules = new List<RuleModel>()
{
  new RuleModel { Field="Country", Label="Country", Type="String",
    Operator="equal", Value = "England" },
  new RuleModel { Field="EmployeeID", Label="EmployeeID", Type="Number",
    Operator="notequal", Value = 1001 }
};
public List<EmployeeDetails> EmployeeData = new List<EmployeeDetails>
{
  new EmployeeDetails{ FirstName = "Martin", EmployeeID = 1001, Country =
    "England", City = "Manchester", HireDate = new DateTime(2014, 4, 23) },
  new EmployeeDetails{ FirstName = "Benjamin", EmployeeID = 1002, Country =
    "England", City = "Birmingham", HireDate = new DateTime(2014, 6, 19) },
  new EmployeeDetails{ FirstName = "Stuart", EmployeeID = 1003, Country =
    "England", City = "London", HireDate = new DateTime(2014, 7, 4) },
  new EmployeeDetails{ FirstName = "Ben", EmployeeID = 1004, Country = "USA",
    City = "California", HireDate = new DateTime(2014, 8, 15) },
  new EmployeeDetails{ FirstName = "Joseph", EmployeeID = 1005, Country =
    "Spain", City = "Madrid", HireDate = new DateTime(2014, 8, 29) }
};
public class EmployeeDetails
{
  public string FirstName { get; set; }
  public int EmployeeID { get; set; }
  public string Country { get; set; }
  public string City { get; set; }
  public DateTime HireDate { get; set; }
}
}

```

You can use this property in the mobile mode to restrict the nested group creation.

Sort the Columns in Blazor QueryBuilder Component

The QueryBuilder has options to sort the column fields by **ascending** or **descending** order. To sort the columns, you need to set [SortDirection](#) property.

ASPX-CS

```

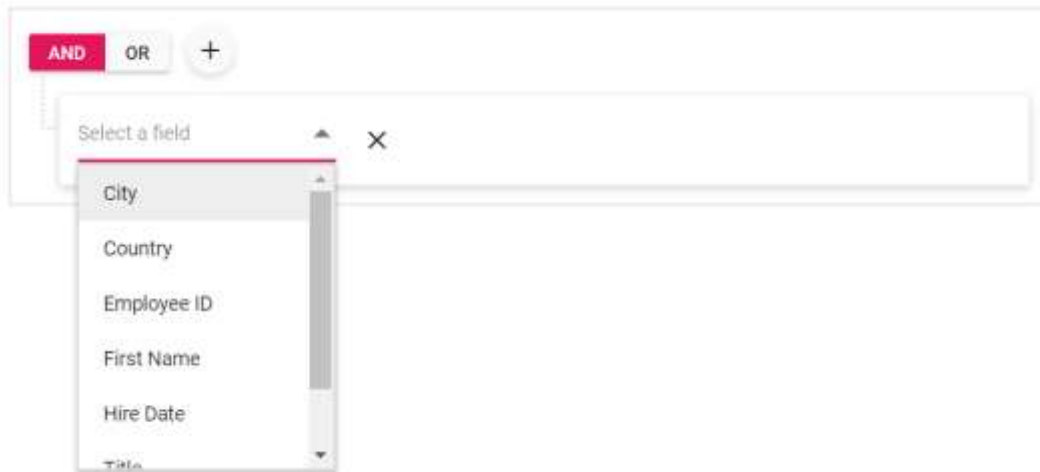
@using Syncfusion.Blazor.QueryBuilder
<SfQueryBuilder DataSource="@EmployeeData"
  SortDirection="SortDirection.Ascending">
  <QueryBuilderRule Condition="or" Rules="@Rules"></QueryBuilderRule>

```

```

<QueryBuilderColumns>
<QueryBuilderColumn Field="EmployeeID" Label="Employee ID"
Type="ColumnType.Number"></QueryBuilderColumn>
<QueryBuilderColumn Field="FirstName" Label="First Name"
Type="ColumnType.String"></QueryBuilderColumn>
<QueryBuilderColumn Field="TitleOfCourtesy" Label="Title of Courtesy"
Type="ColumnType.Boolean" Values="Values"></QueryBuilderColumn>
<QueryBuilderColumn Field="Title" Label="Title"
Type="ColumnType.String"></QueryBuilderColumn>
<QueryBuilderColumn Field="HireDate" Label="Hire Date"
Type="ColumnType.Date"></QueryBuilderColumn>
<QueryBuilderColumn Field="Country" Label="Country"
Type="ColumnType.String"></QueryBuilderColumn>
<QueryBuilderColumn Field="City" Label="City"
Type="ColumnType.String"></QueryBuilderColumn>
</QueryBuilderColumns>
</SfQueryBuilder>
@code {
private string[] Values = new string[] { "Mr.", "Mrs." };
List<RuleModel> Rules = new List<RuleModel>()
{
new RuleModel { Field="Country", Label="Country", Type="String",
Operator="equal", Value = "England" },
new RuleModel { Field="EmployeeID", Label="EmployeeID", Type="Number",
Operator="notequal", Value = 1001 }
};
public List<EmployeeDetails> EmployeeData = new List<EmployeeDetails>
{
new EmployeeDetails{ FirstName = "Martin", EmployeeID = 1001, Country =
"England", City = "Manchester", HireDate = new DateTime(2014, 4, 23) },
new EmployeeDetails{ FirstName = "Benjamin", EmployeeID = 1002, Country =
"England", City = "Birmingham", HireDate = new DateTime(2014, 6, 19) },
new EmployeeDetails{ FirstName = "Stuart", EmployeeID = 1003, Country =
"England", City = "London", HireDate = new DateTime(2014, 7, 4) },
new EmployeeDetails{ FirstName = "Ben", EmployeeID = 1004, Country = "USA",
City = "California", HireDate = new DateTime(2014, 8, 15) },
new EmployeeDetails{ FirstName = "Joseph", EmployeeID = 1005, Country =
"Spain", City = "Madrid", HireDate = new DateTime(2014, 8, 29) }
};
public class EmployeeDetails
{
public string FirstName { get; set; }
public int EmployeeID { get; set; }
public string Country { get; set; }
public string City { get; set; }
public DateTime HireDate { get; set; }
}
}

```



RadioButton

<!-- markdownlint-disable MD024 -->

Getting Started with Blazor RadioButton Component

This section briefly explains about how to include [Blazor Radio Button](#) component in your Blazor server-side application. You can refer [Getting Started with Syncfusion Blazor for Server-side in Visual Studio 2019 page](#) for the introduction and configuring the common specifications.

To get start quickly with Radio Button Component using Blazor, you can check on this video:

{% youtube

"youtube:https://www.youtube.com/watch?v=a6HR1QGAvLo"%}

Importing Syncfusion Blazor component in the application

1. Install the **Syncfusion.Blazor** NuGet package to the application by using the **NuGet Package Manager**.
2. You can add the client-side style resource through [CDN](#) or from [NuGet](#) package in the element of the `~/Pages/_Host.cshtml` page.

Please ensure to check the **Include prerelease** option.

HTML

```
<head>
<link href="_content/Syncfusion.Blazor/styles/bootstrap4.css"
rel="stylesheet" />
@*<link href="https://cdn.syncfusion.com/blazor/{{ site.blazorversion
}}/styles/bootstrap4.css" rel="stylesheet" />*@
</head>
```

For Internet Explorer 11 kindly refer the polyfills. Refer the [documentation](#) for more information.

HTML

```
<head>
```



```
<link href="_content/Syncfusion.Blazor/styles/bootstrap4.css"
rel="stylesheet" />
<script
src="https://github.com/Daddoon/Blazor.Polyfill/releases/download/3.0.1/blaz
or.polyfill.min.js"></script>
</head>
```

Adding component package to the application

Open `/_Imports.razor` file and import the **Syncfusion.Blazor.Buttons** package.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
```

Add SyncfusionBlazor service in Startup.cs

Open the **Startup.cs** file and add services required by Syncfusion components.

Add **services.AddSyncfusionBlazor()** method in the ConfigureServices function as follows.

C#

```
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void ConfigureServices(IServiceCollection services)
        {
            ....
            ....
            services.AddSyncfusionBlazor();
        }
    }
}
```

To enable custom client side resource loading from CRG or CDN. You need to disable resource loading by **AddSyncfusionBlazor(true)** and load the scripts in the HEAD element of the `~/Pages/_Host.cshtml` page.

ASPX-CS

```
<head>
<environment include="Development">
<script src="https://cdn.syncfusion.com/blazor/{ site.blazorversion
}/syncfusion-blazor.min.js">
</script>
</environment>
</head>
```

Adding Radio Button component to the application

Now, add the Syncfusion Blazor Radio Button component in `razor` page in the `Pages` folder. For example, the Radio Button component is added in the `~/Pages/Index.razor` page.

ASPX-CS

```
<SfRadioButton Label="Option 1" Name="options" Value="card" @bind-
Checked="stringChecked"></SfRadioButton>
<SfRadioButton Label="Option 2" Name="options" Value="cash" @bind-
Checked="stringChecked"></SfRadioButton>
@code {
private string stringChecked = "cash";
}
```

Run the application

After successful compilation of your application, simply press F5 to run the application. The Blazor Radio Button component will render in the web browser.



You can also explore our [Blazor RadioButton example](#) that shows how to configure the radio button in Blazor.

See Also

- [Getting Started with Syncfusion Blazor for Client-Side in .NET Core CLI](#)
- [Getting Started with Syncfusion Blazor for Server-Side in Visual Studio 2019 Preview](#)
- [Getting Started with Syncfusion Blazor for Server-Side in .NET Core CLI](#)

Native Events in Blazor RadioButton Component

You can define the native event using an `event` attribute in component. The value of attribute is treated as an event handler. The event specific data will be available in event arguments.

The different event argument types for each event are,

- Focus Events - `UIFocusEventArgs`
- Mouse Events - `UIMouseEventArgs`
- Keyboard Events - `UIKeyboardEventArgs`
- Touch Events - `UITouchEventArgs`

List of Native events supported

The following native event support have been provided to the Radio Button component:

List of Native events				
onchange	oninput	onblur	onfocus	onfocusout
onfocusin	onclick	onkeydown	onkeyup	onkeypress

How to bind click event to Radio Button

The `onclick` attribute is used to bind the click event for Radio Button.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
<SfRadioButton Label="Credit/Debit Card" Name="payment" Value="credit/debit"
@onclick="onClick" @bind-Checked="stringChecked"></SfRadioButton><br />
```

```
<SfRadioButton Label="Net Banking" Name="payment" Value="netbanking"
@onclick="onClick" @bind-Checked="stringChecked"></SfRadioButton>
@code {
private string stringChecked = "netbanking";
private void onClick(Microsoft.AspNetCore.Components.Web.MouseEventArgs
args){
//onclick Event triggered
}
}
```

Label and Size in Blazor RadioButton Component

This section explains the different sizes and labels.

Label

Radio Button caption can be defined by using the [Label](#) property. This reduces the manual addition of label for Radio Button. You can customize the label position before or after the Radio Button through the [LabelPosition](#) property.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
<SfRadioButton Label="Left Side Label" Name="position"
LabelPosition="LabelPosition.Before" Value="Left" @bind-
Checked="stringChecked"></SfRadioButton><br />
<SfRadioButton Label="Right Side Label" Name="position"
LabelPosition="LabelPosition.After" Value="Right" @bind-
Checked="stringChecked"></SfRadioButton>
@code {
private string stringChecked = "Right";
}
```

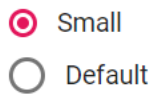


Size

The different Radio Button sizes available are default and small. To reduce the size of the default Radio Button to small, set the [CssClass](#) property to `e-small`.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
<SfRadioButton Label="Small" Name="size" CssClass="e-small" Value="Small"
@bind-Checked="stringChecked"></SfRadioButton><br />
<SfRadioButton Label="Default" Name="size" Value="Default" @bind-
Checked="stringChecked"></SfRadioButton>
@code {
private string stringChecked = "Default";
}
```



See Also

- [How to customize the Radio Button appearance](#)

Styles and Appearances in Blazor RadioButton Component

To modify the RadioButton appearance, you need to override the default CSS of RadioButton component. Please find the list of CSS classes and its corresponding section in RadioButton. Also, you have an option to create your own custom theme for the controls using our [Theme Studio](#).

| CSS Class | Purpose of Class |

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

| .e-radio-wrapper | To customize the radio button wrapper. |

| .e-radio + label:hover::before | To customize the radiobutton on hover. |

| .e-radio:checked + label::after, e-radio:checked + label::before | To customize the checked radiobutton. |

| .e-radio:checked:focus + label::before, .e-radio:checked + label:hover::before | To customize the checked radiobutton on hover. |

How To

Customize Blazor RadioButton Appearance

The appearance of the Radio Button component can be customized by 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 Radio Button is customized through the custom classes to create the primary, success, info, warning, and danger type of Radio Button.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
<SfRadioButton Label="Primary" Name="custom" CssClass="e-primary"
Value="Primary" @bind-Checked="stringChecked"></SfRadioButton><br />
<SfRadioButton Label="Success" Name="custom" CssClass="e-success"
Value="Success" @bind-Checked="stringChecked"></SfRadioButton><br />
<SfRadioButton Label="Info" Name="custom" CssClass="e-info" Value="Info"
@bind-Checked="stringChecked"></SfRadioButton><br />
<SfRadioButton Label="Warning" Name="custom" CssClass="e-warning"
Value="Warning" @bind-Checked="stringChecked"></SfRadioButton><br />
<SfRadioButton Label="Danger" Name="custom" CssClass="e-danger"
Value="Danger" @bind-Checked="stringChecked"></SfRadioButton>
@code {
private string stringChecked = "Danger";
}
</style>
```

```

.e-radio-wrapper.e-success .e-radio:checked + label::after { /* csslint
allow: adjoining-classes */
background-color: #689f38;
color: #689f38;
}
.e-radio-wrapper.e-success .e-radio:checked:focus + label::after,
.e-radio-wrapper.e-success .e-radio:checked + label:hover::after { /*
csslint allow: adjoining-classes */
background-color: #449d44;
}
.e-radio-wrapper.e-success .e-radio:checked + ::before {
border-color: #689f38;
background-color: #fff;
}
.e-radio-wrapper.e-success .e-radio:checked:focus + ::before,
.e-radio-wrapper.e-success .e-radio:checked + label:hover::before { /*
csslint allow: adjoining-classes */
border-color: #449d44;
}
.e-radio-wrapper.e-success .e-radio + label:hover::before {
border-color: #blafaf;
}
.e-radio-wrapper.e-info .e-radio:checked + label::after { /* csslint allow:
adjoining-classes */
background-color: #2196f3;
color: #2196f3;
}
.e-radio-wrapper.e-info .e-radio:checked:focus + label::after,
.e-radio-wrapper.e-info .e-radio:checked + label:hover::after { /* csslint
allow: adjoining-classes */
background-color: #0b7dda;
}
.e-radio-wrapper.e-info .e-radio:checked + label::before {
border-color: #2196f3;
background-color: #fff;
}
.e-radio-wrapper.e-info .e-radio:checked:focus + label::before,
.e-radio-wrapper.e-info .e-radio:checked + label:hover::before {
border-color: #0b7dda;
}
.e-radio-wrapper.e-info .e-radio + label:hover::before {
border-color: #blafaf;
}
.e-radio-wrapper.e-warning .e-radio:checked + label::after { /* csslint
allow: adjoining-classes */
background-color: #ef6c00;
color: #ef6c00;
}
.e-radio-wrapper.e-warning .e-radio:checked:focus + label::after,
.e-radio-wrapper.e-warning .e-radio:checked + label:hover::after { /*
csslint allow: adjoining-classes */
background-color: #cc5c00;
}
.e-radio-wrapper.e-warning .e-radio:checked + label::before {
border-color: #ef6c00;
background-color: #fff;
}

```

```

.e-radio-wrapper.e-warning .e-radio:checked:focus + label::before,
.e-radio-wrapper.e-warning .e-radio:checked + label:hover::before {
border-color: #cc5c00;
}
.e-radio-wrapper.e-warning .e-radio + label:hover::before {
border-color: #blafaf;
}
.e-radio-wrapper.e-danger .e-radio:checked + label::after { /* csslint
allow: adjoining-classes */
background-color: #d84315;
color: #d84315;
}
.e-radio-wrapper.e-danger .e-radio:checked:focus + label::after,
.e-radio-wrapper.e-danger .e-radio:checked + label:hover::after { /* csslint
allow: adjoining-classes */
background-color: #ba330a;
}
.e-radio-wrapper.e-danger .e-radio:checked + label::before {
border-color: #d84315;
background-color: #fff;
}
.e-radio-wrapper.e-danger .e-radio:checked:focus + label::before,
.e-radio-wrapper.e-danger .e-radio:checked + label:hover::before {
border-color: #ba330a;
}
.e-radio-wrapper.e-danger .e-radio + label:hover::before {
border-color: #blafaf
}
</style>

```



Radio Button Model Binding in Blazor RadioButton Component

This section demonstrates the strongly typed extension support in Radio Button. The view that can bind with any model is called as strongly typed view. You can bind any class as model to view. You can access model properties on that view. You can use data associated with model to render the component.

In this sample, first check the male value and click the submit button to post the selected value in the Radio Button. When female value is checked, validation error message will be shown below the Radio Button.

ASPX-CS

```

@using Syncfusion.Blazor.Buttons
@using System.ComponentModel.DataAnnotations
<EditForm Model="Annotate">

```

```

<DataAnnotationsValidator></DataAnnotationsValidator>
<div class="form-group">
<SfRadioButton Label="Male" Name="Gender" Value="male" @bind-
Checked="@Annotate.Gender"></SfRadioButton>
<SfRadioButton Label="Female" Name="Gender" Value="female" @bind-
Checked="@Annotate.Gender"></SfRadioButton>
<ValidationMessage For="@(() => Annotate.Gender)" />
</div>
</EditForm>
@code {
public Annotation Annotate = new Annotation();
public class Annotation
{
[Range(typeof(string), "male", "male", ErrorMessage = "Male gender is
required.")]
public string Gender { get; set; } = "male";
}
}

```



Right-To-Left in Blazor RadioButton Component

Radio Button component has RTL support. This can be achieved by setting [EnableRtl](#) as `true`.

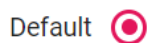
The following example illustrates how to enable right-to-left support in Radio Button component.

ASPX-CS

```

@using Syncfusion.Blazor.Buttons
<SfRadioButton Label="Default" EnableRtl="true" Value="Default" @bind-
Checked="stringChecked"></SfRadioButton>
@code {
private string stringChecked = "Default";
}

```



Set the disabled state in Blazor RadioButton Component

Radio Button component can be enabled/disabled by giving [Disabled](#) property. To disable Radio Button component, the `Disabled` property can be set as `true`.

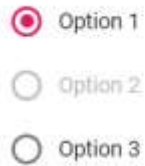
ASPX-CS

```

@using Syncfusion.Blazor.Buttons

```

```
<SfRadioButton Label="Option 1" Name="default" Value="Checked" @bind-
Checked="stringChecked"></SfRadioButton><br />
<SfRadioButton Label="Option 2" Name="default" Value="Disable" @bind-
Checked="stringChecked" Disabled="true"></SfRadioButton><br />
<SfRadioButton Label="Option 3" Name="default" Value="None" @bind-
Checked="stringChecked"></SfRadioButton>
@code {
private string stringChecked = "Checked";
}
```



Range Selector

Blazor Range Selector Component in Server Side App using Visual Studio

This section briefly explains how to include a Range Navigator component in the Blazor server-side application. Refer to the [Getting Started with Syncfusion Blazor for server-side in Visual Studio](#) page for introduction and configuring common specifications.

Importing Syncfusion Blazor Range Navigator component in the application

1. Install **Syncfusion.Blazor** NuGet package to the application by using the **NuGet Package Manager**.
2. Add the client-side resources through [CDN](#) or from [NuGet](#) package in the **HEAD** element of the `~/Pages/_Host.cshtml` page.

HTML

```
<head>
<link href="_content/Syncfusion.Blazor.Themes/bootstrap4.css"
rel="stylesheet" />
<!--CDN-->
@*<link href="https://cdn.syncfusion.com/blazor/{{ site.blazorversion
}}/styles/bootstrap4.css" rel="stylesheet" />*@
</head>
```

For Internet Explorer 11, kindly refer the polyfills. Refer the [documentation](#) for more information.

HTML

```
<head>
<link href="https://cdn.syncfusion.com/blazor/{{ site.blazorversion
}}/styles/bootstrap4.css" rel="stylesheet" />
```



```
<script  
src="https://github.com/Daddoon/Blazor.Polyfill/releases/download/3.0.1/blaz  
or.polyfill.min.js"></script>  
</head>
```

Adding component package to the application

Open the `~/Imports.razor` file and include the **Syncfusion.Blazor.Charts** namespace.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
```

Adding SyncfusionBlazor service in Startup.cs

Open the **Startup.cs** file and add services required by the Syncfusion components using **services.AddSyncfusionBlazor()** method. Add this method in the **ConfigureServices** function as follows.

CSHARP

```
using Syncfusion.Blazor;  
namespace BlazorApplication  
{  
    public class Startup  
    {  
        ....  
        ....  
        public void ConfigureServices(IServiceCollection services)  
        {  
            ....  
            ....  
            services.AddSyncfusionBlazor();  
        }  
    }  
}
```

To enable custom client-side source loading from CRG or CDN, please refer to the section about [custom resources in Blazor application](#).

Adding Range Navigator component

To initialize the Range Navigator component, add the below code to the **Index.razor** view page under `~/Pages` folder. In a new application, if **Index.razor** page has any default content template, then those content can be completely removed and following code can be added.

ASPX-CS

```
@page "/"  
<SfRangeNavigator>  
</SfRangeNavigator>
```

Populate Range Navigator with Data

To bind the data for the Range Navigator component, assign a **IEnumerable** object to the [DataSource](#) property. It can also be provided as an instance of the [DataManager](#).

ASPX-CS

```
@code {
public class StockPrice
{
public DateTime Date { get; set; }
public double Close { get; set; }
}
public List<StockPrice> StockDetails = new List<StockPrice>
{
new StockPrice { Date = new DateTime(2005, 01, 01), Close = 21 },
new StockPrice { Date = new DateTime(2006, 01, 01), Close = 24 },
new StockPrice { Date = new DateTime(2007, 01, 01), Close = 36 },
new StockPrice { Date = new DateTime(2008, 01, 01), Close = 38 },
new StockPrice { Date = new DateTime(2009, 01, 01), Close = 54 },
new StockPrice { Date = new DateTime(2010, 01, 01), Close = 57 },
new StockPrice { Date = new DateTime(2011, 01, 01), Close = 62 }
};
}
```

Now map `Date` and `Close` fields from the datasource to [XName](#) and [YName](#) properties of the [RangeNavigatorSeries](#) and then set the `StockDetails` to [DataSource](#) property.

ASPX-CS

```
<SfRangeNavigator ValueType="RangeValueType.DateTime"
IntervalType="RangeIntervalType.Years" LabelFormat="yyyy">
<RangeNavigatorSeriesCollection>
<RangeNavigatorSeries DataSource="@StockDetails" XName="Date"
Type="RangeNavigatorType.Area" YName="Close"></RangeNavigatorSeries>
</RangeNavigatorSeriesCollection>
</SfRangeNavigator>
```

On successful compilation of the application, the Syncfusion Blazor Range Navigator component will render in the web browser.



See also

- [Getting Started with Syncfusion Blazor for Client-Side in .NET Core CLI](#)
- [Getting Started with Syncfusion Blazor for Server-Side in Visual Studio](#)
- [Getting Started with Syncfusion Blazor for Server-Side in .NET Core CLI](#)

Range in Blazor Range Selector Component

The Range Selector's left and right thumbs are used to indicate the selected range in the large collection of data. A range can be selected in the following ways:

- By dragging the thumbs.
- By tapping on the labels.
- By setting the start and the end through the [Value](#) property.

<!-- markdownlint-disable MD036 -->

Value Binding

This section describes how to bind the value to the Range Selector component in the following ways:

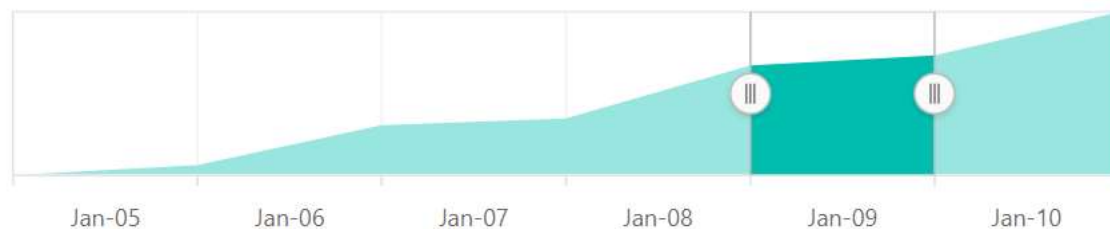
- One-way binding
- Two-way binding

One-way binding

As shown in the following example, the [Value](#) property can be used directly as an object or from code-behind for the Range Selector.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfRangeNavigator Value="@Value" ValueType="RangeValueType.DateTime"
LabelFormat="MMM-yy" IntervalType="RangeIntervalType.Years" Interval="1">
<RangeNavigatorRangeTooltipSettings
Enable="true"></RangeNavigatorRangeTooltipSettings>
<RangeNavigatorSeriesCollection>
<RangeNavigatorSeries DataSource="@StockInfo" XName="Date"
Type="RangeNavigatorType.Area" YName="Close">
</RangeNavigatorSeries>
</RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code {
public class StockDetails
{
public DateTime Date { get; set; }
public double Close { get; set; }
}
public DateTime[] Value = new DateTime[] { new DateTime(2009, 01, 01), new
DateTime(2010, 01, 01) };
public List<StockDetails> StockInfo = new List<StockDetails>
{
new StockDetails { Date = new DateTime(2005, 01, 01), Close = 21 },
new StockDetails { Date = new DateTime(2006, 01, 01), Close = 24 },
new StockDetails { Date = new DateTime(2007, 01, 01), Close = 36 },
new StockDetails { Date = new DateTime(2008, 01, 01), Close = 38 },
new StockDetails { Date = new DateTime(2009, 01, 01), Close = 54 },
new StockDetails { Date = new DateTime(2010, 01, 01), Close = 57 },
new StockDetails { Date = new DateTime(2011, 01, 01), Close = 70 }
};
}
```



Two-way binding

The **@bind-Value** code-behind attribute in the Range Selector can be used to achieve two-way binding. The following example shows how to achieve two-way binding for the Range Selector.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfRangeNavigator @bind-Value="@Value" ValueType="RangeValueType.DateTime"
LabelFormat="MMM-yy" IntervalType="RangeIntervalType.Years" Interval="1">
  <RangeNavigatorRangeTooltipSettings
  Enable="true"></RangeNavigatorRangeTooltipSettings>
  <RangeNavigatorSeriesCollection>
    <RangeNavigatorSeries DataSource="@StockInfo" XName="Date"
    Type="RangeNavigatorType.Area" YName="Close">
  </RangeNavigatorSeries>
  </RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code {
  public class StockDetails
  {
    public DateTime Date { get; set; }
    public double Close { get; set; }
  }
  public object Value = new DateTime[] { new DateTime(2009, 01, 01), new
  DateTime(2010, 01, 01) };
  public List<StockDetails> StockInfo = new List<StockDetails>
  {
    new StockDetails { Date = new DateTime(2005, 01, 01), Close = 21 },
    new StockDetails { Date = new DateTime(2006, 01, 01), Close = 24 },
    new StockDetails { Date = new DateTime(2007, 01, 01), Close = 36 },
    new StockDetails { Date = new DateTime(2008, 01, 01), Close = 38 },
    new StockDetails { Date = new DateTime(2009, 01, 01), Close = 54 },
    new StockDetails { Date = new DateTime(2010, 01, 01), Close = 57 },
    new StockDetails { Date = new DateTime(2011, 01, 01), Close = 70 }
  };
}
```

*Lightweight in Blazor Range Selector Component*

By default, when the [DataSource](#) for [RangeNavigatorSeries](#) is empty, a lightweight Range Selector will be shown without Chart.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfRangeNavigator Value="@Value" ValueType="RangeValueType.DateTime"
DataSource="@DataSource" XName="X">
```

```

YName="Y" IntervalType="@RangeIntervalType.Years"
LabelIntersectAction="RangeLabelIntersectAction.Hide">
<RangeNavigatorRangeTooltipSettings
Enable="true"></RangeNavigatorRangeTooltipSettings>
</SfRangeNavigator>
@code {
public class SampleData
{
public DateTime X { get; set; }
public double Y { get; set; }
}
public DateTime[] Value = new DateTime[] { new DateTime(2007, 01, 01), new
DateTime(2009, 01, 01) };
public List<SampleData> DataSource = new List<SampleData>
{
new SampleData { X = new DateTime(2005, 01, 01), Y = 21 },
new SampleData { X = new DateTime(2006, 01, 01), Y = 24 },
new SampleData { X = new DateTime(2007, 01, 01), Y = 36 },
new SampleData { X = new DateTime(2008, 01, 01), Y = 38 },
new SampleData { X = new DateTime(2009, 01, 01), Y = 54 },
new SampleData { X = new DateTime(2010, 01, 01), Y = 57 },
new SampleData { X = new DateTime(2011, 01, 01), Y = 70 }
};
}

```



See Also

- [Period Selector](#)

Series Type in Blazor Range Selector Component

To render the data, the Range Selector supports three types of series.

<!-- markdownlint-disable MD036 -->

Line

<!-- markdownlint-disable MD036 -->

To render a line series, use series [Type](#) as **Line**. By default, the line series is rendered in the Range Selector.

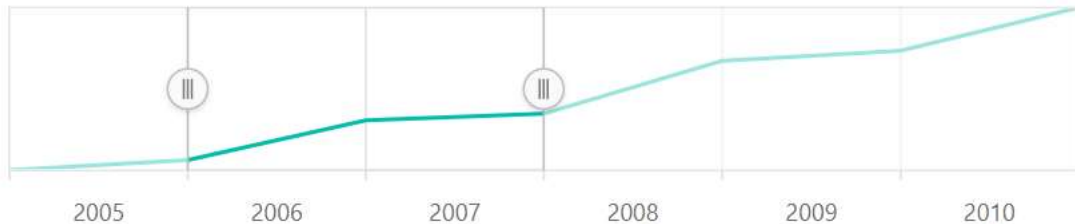
ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfRangeNavigator Value="@Value" ValueType="RangeValueType.DateTime"
IntervalType="RangeIntervalType.Years">
<RangeNavigatorSeriesCollection>
<RangeNavigatorSeries DataSource="@StockInfo" XName="Date"
Type="RangeNavigatorType.Line" YName="Close">
</RangeNavigatorSeries>
</RangeNavigatorSeriesCollection>
</SfRangeNavigator>

```

```
@code {
public class StockDetails
{
public DateTime Date { get; set; }
public double Close { get; set; }
}
public DateTime[] Value = new DateTime[] { new DateTime(2006, 01, 01), new
DateTime(2008, 01, 01) };
public List<StockDetails> StockInfo = new List<StockDetails>
{
new StockDetails { Date = new DateTime(2005, 01, 01), Close = 21 },
new StockDetails { Date = new DateTime(2006, 01, 01), Close = 24 },
new StockDetails { Date = new DateTime(2007, 01, 01), Close = 36 },
new StockDetails { Date = new DateTime(2008, 01, 01), Close = 38 },
new StockDetails { Date = new DateTime(2009, 01, 01), Close = 54 },
new StockDetails { Date = new DateTime(2010, 01, 01), Close = 57 },
new StockDetails { Date = new DateTime(2011, 01, 01), Close = 70 }
};
}
```



Area

To render an area series, use series [Type](#) as **Area**.

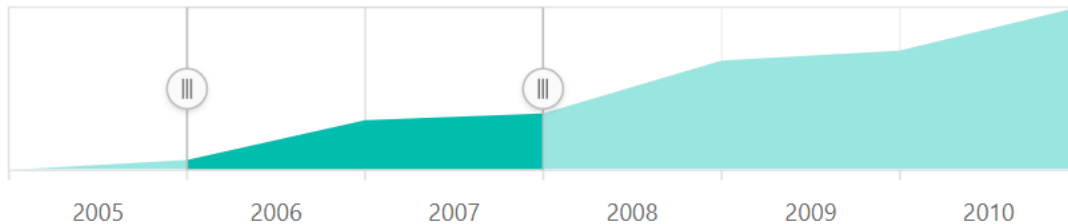
ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfRangeNavigator Value="@Value" ValueType="RangeValueType.DateTime"
IntervalType="RangeIntervalType.Years">
<RangeNavigatorSeriesCollection>
<RangeNavigatorSeries DataSource="@StockInfo" XName="Date"
Type="RangeNavigatorType.Area" YName="Close">
</RangeNavigatorSeries>
</RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code {
public class StockDetails
{
public DateTime Date { get; set; }
public double Close { get; set; }
}
public DateTime[] Value = new DateTime[] { new DateTime(2006, 01, 01), new
DateTime(2008, 01, 01) };
public List<StockDetails> StockInfo = new List<StockDetails>
{
new StockDetails { Date = new DateTime(2005, 01, 01), Close = 21 },
new StockDetails { Date = new DateTime(2006, 01, 01), Close = 24 },
new StockDetails { Date = new DateTime(2007, 01, 01), Close = 36 },
}
```

```

new StockDetails { Date = new DateTime(2008, 01, 01), Close = 38 },
new StockDetails { Date = new DateTime(2009, 01, 01), Close = 54 },
new StockDetails { Date = new DateTime(2010, 01, 01), Close = 57 },
new StockDetails { Date = new DateTime(2011, 01, 01), Close = 70 }
};
}

```



Step Line

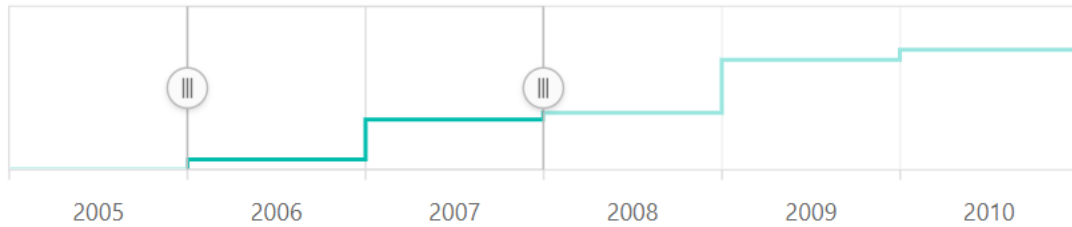
To render a Step line series, use series [Type](#) as **Step Line**.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfRangeNavigator Value="@Value" ValueType="RangeValueType.DateTime"
IntervalType="RangeIntervalType.Years">
  <RangeNavigatorSeriesCollection>
    <RangeNavigatorSeries DataSource="@StockInfo" XName="Date"
Type="RangeNavigatorType.StepLine" YName="Close">
  </RangeNavigatorSeries>
</RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code {
public class StockDetails
{
public DateTime Date { get; set; }
public double Close { get; set; }
}
public DateTime[] Value = new DateTime[] { new DateTime(2006, 01, 01), new
DateTime(2008, 01, 01) };
public List<StockDetails> StockInfo = new List<StockDetails>
{
new StockDetails { Date = new DateTime(2005, 01, 01), Close = 21 },
new StockDetails { Date = new DateTime(2006, 01, 01), Close = 24 },
new StockDetails { Date = new DateTime(2007, 01, 01), Close = 36 },
new StockDetails { Date = new DateTime(2008, 01, 01), Close = 38 },
new StockDetails { Date = new DateTime(2009, 01, 01), Close = 54 },
new StockDetails { Date = new DateTime(2010, 01, 01), Close = 57 },
new StockDetails { Date = new DateTime(2011, 01, 01), Close = 70 }
};
}

```



<!-- markdownlint-disable MD036 -->

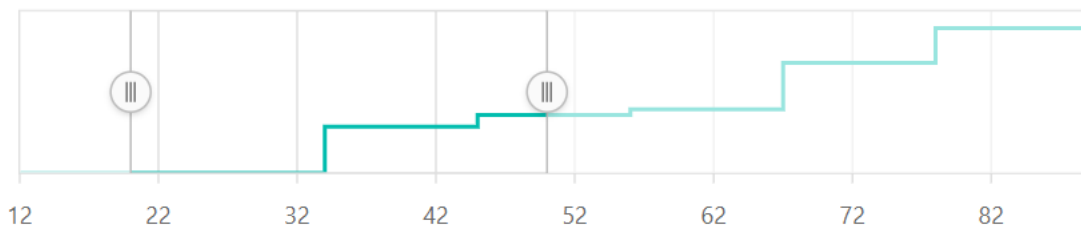
Type of Data in Blazor Range Selector Component

Numeric

The numeric scale is used to represent the numeric values of data in a Range Selector. By default, the [ValueType](#) of a Range Selector is **Double**.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfRangeNavigator Value="@Value" ValueType=RangeValueType.Double>
<RangeNavigatorSeriesCollection>
<RangeNavigatorSeries DataSource="@StockInfo" XName="Day"
Type="RangeNavigatorType.StepLine" YName="Close">
</RangeNavigatorSeries>
</RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code{
public class StockDetails
{
public double Day { get; set; }
public double Close { get; set; }
}
public List<StockDetails> StockInfo = new List<StockDetails>
{
new StockDetails { Day = 12, Close = 28 },
new StockDetails { Day = 34, Close = 44 },
new StockDetails { Day = 45, Close = 48 },
new StockDetails { Day = 56, Close = 50 },
new StockDetails { Day = 67, Close = 66 },
new StockDetails { Day = 78, Close = 78 },
new StockDetails { Day = 89, Close = 84 }
};
public int[] Value = new int[] { 20, 50 };
}
```

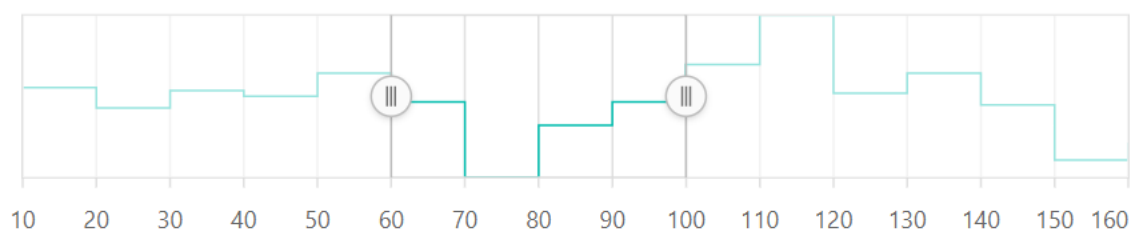


Range

The minimum and the maximum of the scale will be calculated automatically based on the provided data. It can be customized by using the [Minimum](#), [Maximum](#), and [Interval](#) properties.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfRangeNavigator Interval="10" Value="@Value"
ValueType=RangeValueType.Double>
<RangeNavigatorSeriesCollection>
<RangeNavigatorSeries DataSource="@StockInfo" XName="Date"
Type="RangeNavigatorType.StepLine" YName="Close">
</RangeNavigatorSeries>
</RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code {
public class StockDetails
{
public double Date { get; set; }
public double Close { get; set; }
}
public int[] Value = new int[] { 60, 100 };
public List<StockDetails> StockInfo = new List<StockDetails>
{
new StockDetails { Date = 10, Close = 35 },
new StockDetails { Date = 20, Close = 28 },
new StockDetails { Date = 30, Close = 34 },
new StockDetails { Date = 40, Close = 32 },
new StockDetails { Date = 50, Close = 40 },
new StockDetails { Date = 60, Close = 30 },
new StockDetails { Date = 70, Close = 4 },
new StockDetails { Date = 80, Close = 22 },
new StockDetails { Date = 90, Close = 22 },
new StockDetails { Date = 100, Close = 30 },
new StockDetails { Date = 110, Close = 43 },
new StockDetails { Date = 120, Close = 60 },
new StockDetails { Date = 130, Close = 33 },
new StockDetails { Date = 140, Close = 40 },
new StockDetails { Date = 150, Close = 29 },
new StockDetails { Date = 160, Close = 10 },
new StockDetails { Date = 160, Close = 16 },
};
}
```

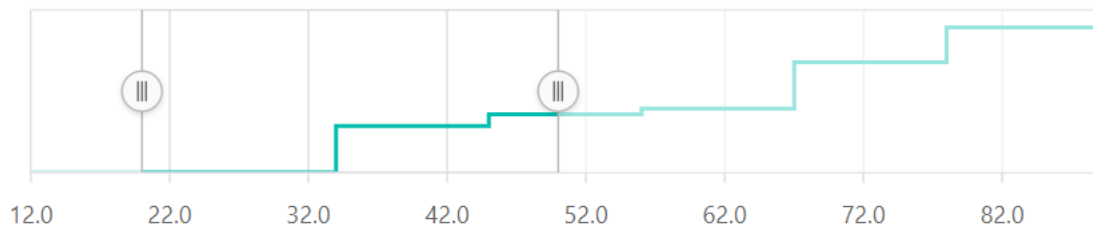


Label Format

The numeric labels can be formatted using the [LabelFormat](#) property and it supports all the globalized formats.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfRangeNavigator Value="@Value" ValueType=RangeValueType.Double
LabelFormat="n1">
<RangeNavigatorSeriesCollection>
<RangeNavigatorSeries DataSource="@StockInfo" XName="Day"
Type="RangeNavigatorType.StepLine" YName="Close">
</RangeNavigatorSeries>
</RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code{
public class StockDetails
{
public double Day { get; set; }
public double Close { get; set; }
}
public List<StockDetails> StockInfo = new List<StockDetails>
{
new StockDetails { Day = 12, Close = 28 },
new StockDetails { Day = 34, Close = 44 },
new StockDetails { Day = 45, Close = 48 },
new StockDetails { Day = 56, Close = 50 },
new StockDetails { Day = 67, Close = 66 },
new StockDetails { Day = 78, Close = 78 },
new StockDetails { Day = 89, Close = 84 }
};
public int[] Value = new int[] { 20, 50 };
}
```



The following table shows the results of applying some commonly used label formats to 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.

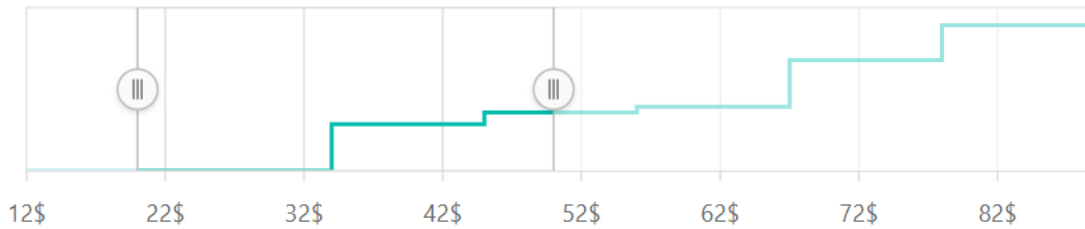
0.01	p1	1.0%	The number is converted to percentage with 1 decimal place.
0.01	p2	1.00%	The number is converted to percentage with 2 decimal places.
0.01	p3	1.000%	The number is converted to percentage with 3 decimal places.
1000	c1	\$1,000.0	The currency symbol is appended to number and the number is rounded to 1 decimal place.
1000	c2	\$1,000.00	The currency symbol is appended to number and the number is rounded to 2 decimal places.

Custom Label Format

The Range Selector also supports the Custom Label formats using the placeholders such as **{value}\$**, in which the value represents the axis label, e.g. 20\$.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfRangeNavigator Value="@Value" ValueType=RangeValueType.Double
LabelFormat="{value}$">
<RangeNavigatorSeriesCollection>
<RangeNavigatorSeries DataSource="@StockInfo" XName="Day"
Type="RangeNavigatorType.StepLine" YName="Close">
</RangeNavigatorSeries>
</RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code{
public class StockDetails
{
public double Day { get; set; }
public double Close { get; set; }
}
public List<StockDetails> StockInfo = new List<StockDetails>
{
new StockDetails { Day = 12, Close = 28 },
new StockDetails { Day = 34, Close = 44 },
new StockDetails { Day = 45, Close = 48 },
new StockDetails { Day = 56, Close = 50 },
new StockDetails { Day = 67, Close = 66 },
new StockDetails { Day = 78, Close = 78 },
new StockDetails { Day = 89, Close = 84 }
};
public int[] Value = new int[] { 20, 50 };
}
```



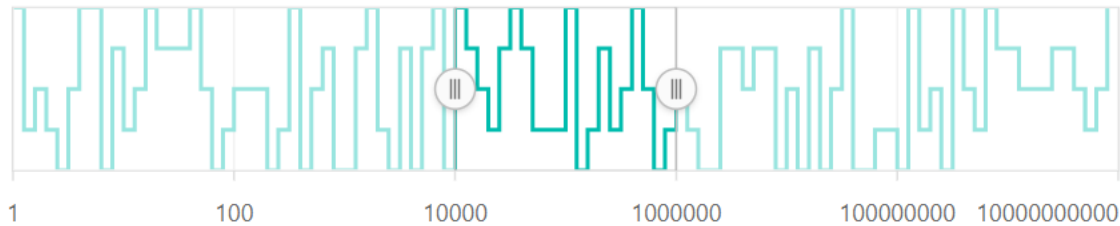
Logarithmic

<!-- markdownlint-disable MD033 -->

The Logarithmic supports the logarithmic scale, and it is used to visualize the data when the Range Selector has numerical values in both the lower (e.g.: 10-6) and the higher (e.g.: 106) orders of the magnitude.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfRangeNavigator Value="@Value" Interval="2"
ValueType="RangeValueType.Logarithmic">
<RangeNavigatorSeriesCollection>
<RangeNavigatorSeries DataSource="@StockInfo" XName="Day"
Type="RangeNavigatorType.StepLine" YName="Close" Width="2">
</RangeNavigatorSeries>
</RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code {
public class StockDetails
{
public double Day { get; set; }
public double Close { get; set; }
}
public int[] Value = new int[] { 4, 6 };
private Random random = new Random();
public List<StockDetails> StockInfo;
protected override async Task OnInitializedAsync()
{
StockInfo = this.GetData();
}
public List<StockDetails> GetData()
{
List<StockDetails> data = new List<StockDetails>();
for (int i = 0; i < 100; i++)
{
data.Add(new StockDetails
{
Day = Math.Pow(10, i * 0.1),
Close = (random.Next(5, 10) * (80 - 30 + 1)) + 30,
});
}
return data;
}
}
```

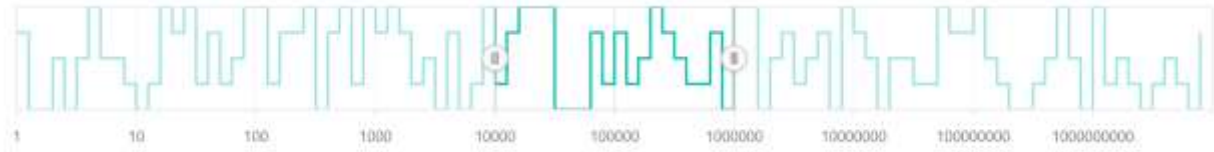


Range

The minimum and the maximum of the Range Selector will be calculated automatically based on the provided data. It can be customized by using the [Minimum](#), the [Maximum](#), and the [Interval](#) properties.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfRangeNavigator Value="@Value" Interval="1"
ValueType="RangeValueType.Logarithmic">
<RangeNavigatorSeriesCollection>
<RangeNavigatorSeries DataSource="@StockInfo" XName="Day"
Type="RangeNavigatorType.StepLine" YName="Close" Width="2">
</RangeNavigatorSeries>
</RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code {
public class StockDetails
{
public double Day { get; set; }
public double Close { get; set; }
}
public int[] Value = new int[] { 4, 6 };
private Random random = new Random();
public List<StockDetails> StockInfo;
protected override async Task OnInitializedAsync()
{
StockInfo = this.GetData();
}
public List<StockDetails> GetData()
{
List<StockDetails> data = new List<StockDetails>();
for (int i = 0; i < 100; i++)
{
data.Add(new StockDetails
{
Day = Math.Pow(10, i * 0.1),
Close = (random.Next(5, 10) * (80 - 30 + 1)) + 30,
});
}
return data;
}
}
```

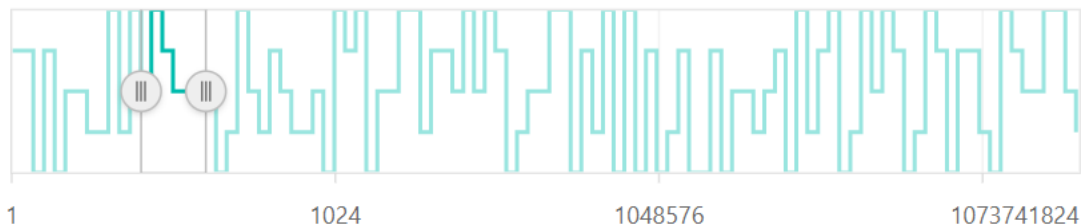


Logarithmic Base

The Logarithmic Base can be customized using the [LogBase](#) property. The default value of this property is **10**.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfRangeNavigator Value="@Value" LogBase="2"
ValueType="RangeValueType.Logarithmic">
<RangeNavigatorSeriesCollection>
<RangeNavigatorSeries DataSource="@StockInfo" XName="Day"
Type="RangeNavigatorType.StepLine" YName="Close" Width="2">
</RangeNavigatorSeries>
</RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code {
public class StockDetails
{
public double Day { get; set; }
public double Close { get; set; }
}
public int[] Value = new int[] { 4, 6 };
private Random random = new Random();
public List<StockDetails> StockInfo;
protected override async Task OnInitializedAsync()
{
StockInfo = this.GetData();
}
public List<StockDetails> GetData()
{
List<StockDetails> data = new List<StockDetails>();
for (int i = 0; i < 100; i++)
{
data.Add(new StockDetails
{
Day = Math.Pow(10, i * 0.1),
Close = (random.Next(5, 10) * (80 - 30 + 1)) + 30,
});
}
return data;
}
}
```

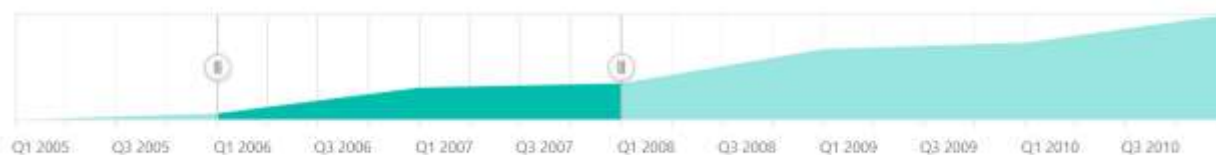


DateTime

The Range Selector supports the DateTime scale and displays the DateTime values as labels in the specified format.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfRangeNavigator Value="@Value" ValueType="RangeValueType.DateTime">
  <RangeNavigatorSeriesCollection>
    <RangeNavigatorSeries DataSource="@StockInfo" XName="Date"
      Type="RangeNavigatorType.Area" YName="Y">
    </RangeNavigatorSeries>
  </RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code{
public class StockDetails
{
    public DateTime Date { get; set; }
    public double Y { get; set; }
}
public DateTime[] Value = new DateTime[] { new DateTime(2006, 01, 01), new
DateTime(2008, 01, 01) };
public List<StockDetails> StockInfo = new List<StockDetails>
{
    new StockDetails { Date = new DateTime(2005, 01, 01), Y = 21 },
    new StockDetails { Date = new DateTime(2006, 01, 01), Y = 24 },
    new StockDetails { Date = new DateTime(2007, 01, 01), Y = 36 },
    new StockDetails { Date = new DateTime(2008, 01, 01), Y = 38 },
    new StockDetails { Date = new DateTime(2009, 01, 01), Y = 54 },
    new StockDetails { Date = new DateTime(2010, 01, 01), Y = 57 },
    new StockDetails { Date = new DateTime(2011, 01, 01), Y = 70 }
};
}
```



Interval Customization

The DateTime intervals can be customized using the [Interval](#) and the [IntervalType](#) properties of the Range Selector. For example, if the [Interval](#) is set to 2 and the [IntervalType](#) is set to years, the interval will be considered to be 2 years.

DateTime supports the following interval types:

- Auto
- Years
- Quarter
- Months
- Weeks
- Days
- Hours
- Minutes

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfRangeNavigator Value="@Value" ValueType="RangeValueType.DateTime"
IntervalType="RangeIntervalType.Months" Interval="2">
<RangeNavigatorSeriesCollection>
<RangeNavigatorSeries DataSource="@StockInfo" XName="Date"
Type="RangeNavigatorType.Area" YName="Y">
</RangeNavigatorSeries>
</RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code{
public class StockDetails
{
public DateTime Date { get; set; }
public double Y { get; set; }
}
public List<StockDetails> StockInfo = new List<StockDetails>
{
new StockDetails { Date = new DateTime(2005, 01, 01), Y = 21 },
new StockDetails { Date = new DateTime(2006, 01, 01), Y = 24 },
new StockDetails { Date = new DateTime(2007, 01, 01), Y = 36 },
new StockDetails { Date = new DateTime(2008, 01, 01), Y = 38 },
new StockDetails { Date = new DateTime(2009, 01, 01), Y = 54 },
new StockDetails { Date = new DateTime(2010, 01, 01), Y = 57 },
new StockDetails { Date = new DateTime(2011, 01, 01), Y = 70 }
};
public DateTime[] Value = new DateTime[] { new DateTime(2006, 01, 01), new
DateTime(2008, 01, 01) };
}
```



Label Format

The [LabelFormat](#) property is used to format and parse the date to all globalize format.

ASPX-CS

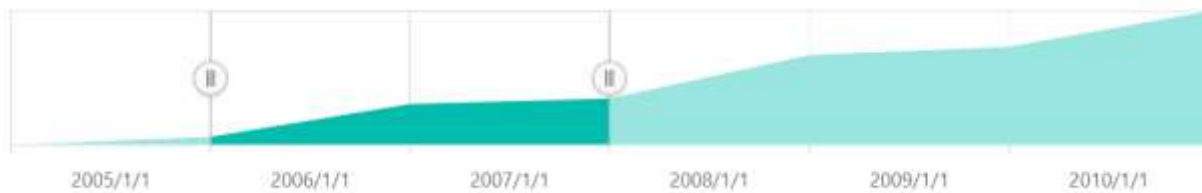
```
@using Syncfusion.Blazor.Charts
<SfRangeNavigator Value="@Value" ValueType="RangeValueType.DateTime"
LabelFormat="y/M/d">
<RangeNavigatorSeriesCollection>
```



```

<RangeNavigatorSeries DataSource="@StockInfo" XName="Date"
Type="RangeNavigatorType.Area" YName="Y">
</RangeNavigatorSeries>
</RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code{
public class StockDetails
{
public DateTime Date { get; set; }
public double Y { get; set; }
}
public List<StockDetails> StockInfo = new List<StockDetails>
{
new StockDetails { Date = new DateTime(2005, 01, 01), Y = 21 },
new StockDetails { Date = new DateTime(2006, 01, 01), Y = 24 },
new StockDetails { Date = new DateTime(2007, 01, 01), Y = 36 },
new StockDetails { Date = new DateTime(2008, 01, 01), Y = 38 },
new StockDetails { Date = new DateTime(2009, 01, 01), Y = 54 },
new StockDetails { Date = new DateTime(2010, 01, 01), Y = 57 },
new StockDetails { Date = new DateTime(2011, 01, 01), Y = 70 }
};
public DateTime[] Value = new DateTime[] { new DateTime(2006, 01, 01), new
DateTime(2008, 01, 01) };
}

```



The following table shows the results of applying some common DateTime 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 the day format.
new Date(2000, 03, 10)	yMd	04/10/2000	The date is displayed in the 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	The 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.

Period Selector in Blazor Range Selector Component

The period selector allows to choose a time range with specific periods.

Periods

An array of objects that allows the users to specify pre-defined time intervals. The [Interval](#) property specifies the count value of the button, the [Selected](#) property allows the users to select the period button initially, and the [Text](#) property specifies the text to be displayed on the button. The [IntervalType](#) property allows the users to customize the interval type and it supports the following types:

- Auto
- Years
- Quarter
- Months
- Weeks
- Days
- Hours
- Minutes
- Seconds

ASPX-CS

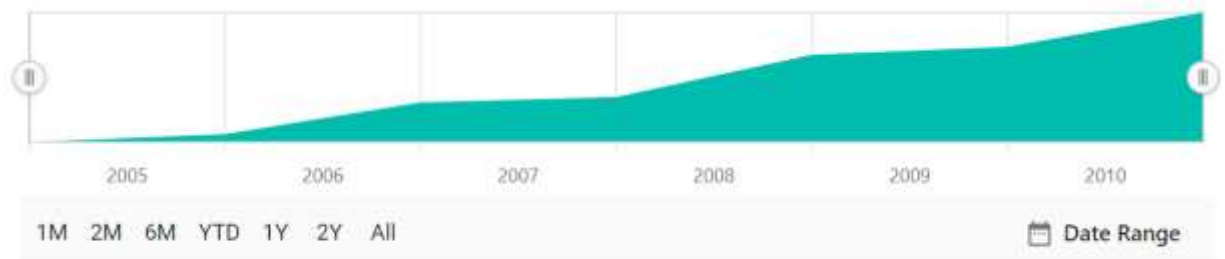
```
@using Syncfusion.Blazor.Charts
<SfRangeNavigator ValueType="RangeValueType.DateTime">
  <RangeNavigatorPeriodSelectorSettings>
    <RangeNavigatorPeriods>
      <RangeNavigatorPeriod Interval="1" IntervalType="RangeIntervalType.Months"
        Text="1M"></RangeNavigatorPeriod>
      <RangeNavigatorPeriod Interval="2" IntervalType="RangeIntervalType.Months"
        Text="2M"></RangeNavigatorPeriod>
      <RangeNavigatorPeriod Interval="6" IntervalType="RangeIntervalType.Months"
        Text="6M"></RangeNavigatorPeriod>
      <RangeNavigatorPeriod Text="YTD"></RangeNavigatorPeriod>
      <RangeNavigatorPeriod Interval="1" IntervalType="RangeIntervalType.Years"
        Text="1Y"></RangeNavigatorPeriod>
      <RangeNavigatorPeriod Interval="2" IntervalType="RangeIntervalType.Years"
        Text="2Y"></RangeNavigatorPeriod>
      <RangeNavigatorPeriod Text="All"></RangeNavigatorPeriod>
    </RangeNavigatorPeriods>
  </RangeNavigatorPeriodSelectorSettings>
  <RangeNavigatorSeriesCollection>
    <RangeNavigatorSeries DataSource="@StockInfo" XName="Date"
      Type="RangeNavigatorType.Area" YName="Close"></RangeNavigatorSeries>
  </RangeNavigatorSeriesCollection>
</SfRangeNavigator>

@code {
  public class StockDetails
  {
    public DateTime Date { get; set; }
    public double Close { get; set; }
  }
  public DateTime[] Value = new DateTime[] { new DateTime(2009, 01, 01), new
  DateTime(2010, 01, 01) };
  public List<StockDetails> StockInfo = new List<StockDetails>
  {
```

```

new StockDetails { Date = new DateTime(2005, 01, 01), Close = 21 },
new StockDetails { Date = new DateTime(2006, 01, 01), Close = 24 },
new StockDetails { Date = new DateTime(2007, 01, 01), Close = 36 },
new StockDetails { Date = new DateTime(2008, 01, 01), Close = 38 },
new StockDetails { Date = new DateTime(2009, 01, 01), Close = 54 },
new StockDetails { Date = new DateTime(2010, 01, 01), Close = 57 },
new StockDetails { Date = new DateTime(2011, 01, 01), Close = 70 }
};
}

```



Position

The [Position](#) property allows the users to position the period selector at the **Top** or **Bottom**.

ASPX-CS

```

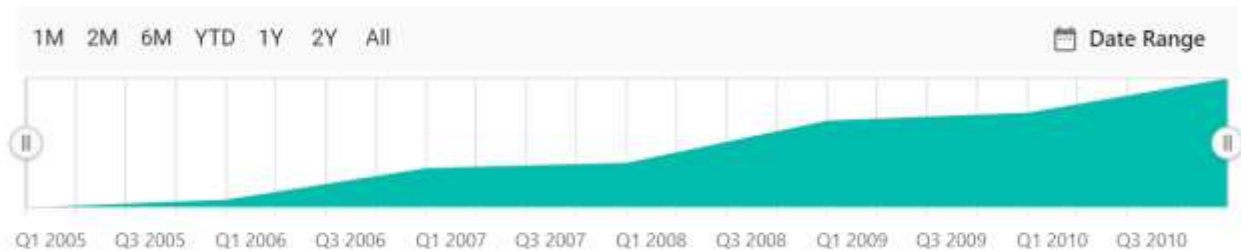
@using Syncfusion.Blazor.Charts
<SfRangeNavigator ValueType="RangeValueType.DateTime">
  <RangeNavigatorPeriodSelectorSettings Position="PeriodSelectorPosition.Top">
    <RangeNavigatorPeriods>
      <RangeNavigatorPeriod Interval="1" IntervalType="RangeIntervalType.Months"
        Text="1M"></RangeNavigatorPeriod>
      <RangeNavigatorPeriod Interval="2" IntervalType="RangeIntervalType.Months"
        Text="2M"></RangeNavigatorPeriod>
      <RangeNavigatorPeriod Interval="6" IntervalType="RangeIntervalType.Months"
        Text="6M"></RangeNavigatorPeriod>
      <RangeNavigatorPeriod Text="YTD"></RangeNavigatorPeriod>
      <RangeNavigatorPeriod Interval="1" IntervalType="RangeIntervalType.Years"
        Text="1Y"></RangeNavigatorPeriod>
      <RangeNavigatorPeriod Interval="2" IntervalType="RangeIntervalType.Years"
        Text="2Y"></RangeNavigatorPeriod>
      <RangeNavigatorPeriod Text="All"></RangeNavigatorPeriod>
    </RangeNavigatorPeriods>
  </RangeNavigatorPeriodSelectorSettings>
  <RangeNavigatorSeriesCollection>
    <RangeNavigatorSeries DataSource="@StockInfo" XName="Date"
      Type="RangeNavigatorType.Area" YName="Close"></RangeNavigatorSeries>
  </RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code {
  public class StockDetails
  {
    public DateTime Date { get; set; }
    public double Close { get; set; }
  }
  public DateTime[] Value = new DateTime[] { new DateTime(2009, 01, 01), new
  DateTime(2010, 01, 01) };
}

```

```

public List<StockDetails> StockInfo = new List<StockDetails>
{
    new StockDetails { Date = new DateTime(2005, 01, 01), Close = 21 },
    new StockDetails { Date = new DateTime(2006, 01, 01), Close = 24 },
    new StockDetails { Date = new DateTime(2007, 01, 01), Close = 36 },
    new StockDetails { Date = new DateTime(2008, 01, 01), Close = 38 },
    new StockDetails { Date = new DateTime(2009, 01, 01), Close = 54 },
    new StockDetails { Date = new DateTime(2010, 01, 01), Close = 57 },
    new StockDetails { Date = new DateTime(2011, 01, 01), Close = 70 }
};
}

```



Height

The [Height](#) property allows the users to specify the height of the period selector. The default value of the height property is **43px**.

ASPX-CS

```

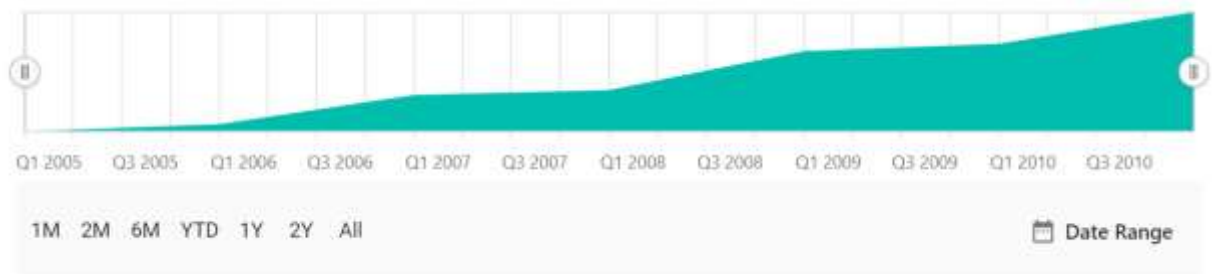
@using Syncfusion.Blazor.Charts
<SfRangeNavigator ValueType="RangeValueType.DateTime">
  <RangeNavigatorPeriodSelectorSettings Height="65">
    <RangeNavigatorPeriods>
      <RangeNavigatorPeriod Interval="1" IntervalType="RangeIntervalType.Months"
        Text="1M"></RangeNavigatorPeriod>
      <RangeNavigatorPeriod Interval="2" IntervalType="RangeIntervalType.Months"
        Text="2M"></RangeNavigatorPeriod>
      <RangeNavigatorPeriod Interval="6" IntervalType="RangeIntervalType.Months"
        Text="6M"></RangeNavigatorPeriod>
      <RangeNavigatorPeriod Text="YTD"></RangeNavigatorPeriod>
      <RangeNavigatorPeriod Interval="1" IntervalType="RangeIntervalType.Years"
        Text="1Y"></RangeNavigatorPeriod>
      <RangeNavigatorPeriod Interval="2" IntervalType="RangeIntervalType.Years"
        Text="2Y"></RangeNavigatorPeriod>
      <RangeNavigatorPeriod Text="All"></RangeNavigatorPeriod>
    </RangeNavigatorPeriods>
  </RangeNavigatorPeriodSelectorSettings>
  <RangeNavigatorSeriesCollection>
    <RangeNavigatorSeries DataSource="@StockInfo" XName="Date"
      Type="RangeNavigatorType.Area" YName="Close"></RangeNavigatorSeries>
  </RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code {
    public class StockDetails
    {
        public DateTime Date { get; set; }
        public double Close { get; set; }
    }
}

```

```

public DateTime[] Value = new DateTime[] { new DateTime(2009, 01, 01), new
DateTime(2010, 01, 01) };
public List<StockDetails> StockInfo = new List<StockDetails>
{
    new StockDetails { Date = new DateTime(2005, 01, 01), Close = 21 },
    new StockDetails { Date = new DateTime(2006, 01, 01), Close = 24 },
    new StockDetails { Date = new DateTime(2007, 01, 01), Close = 36 },
    new StockDetails { Date = new DateTime(2008, 01, 01), Close = 38 },
    new StockDetails { Date = new DateTime(2009, 01, 01), Close = 54 },
    new StockDetails { Date = new DateTime(2010, 01, 01), Close = 57 },
    new StockDetails { Date = new DateTime(2011, 01, 01), Close = 70 }
};
}

```



Visibility

The [DisableRangeSelector](#) property allows the users to display only the period selector and not the Range Selector.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfRangeNavigator DisableRangeSelector="true"
ValueType="RangeValueType.DateTime">
<RangeNavigatorPeriodSelectorSettings>
<RangeNavigatorPeriods>
<RangeNavigatorPeriod Interval="1" IntervalType="RangeIntervalType.Months"
Text="1M"></RangeNavigatorPeriod>
<RangeNavigatorPeriod Interval="2" IntervalType="RangeIntervalType.Months"
Text="2M"></RangeNavigatorPeriod>
<RangeNavigatorPeriod Interval="6" IntervalType="RangeIntervalType.Months"
Text="6M"></RangeNavigatorPeriod>
<RangeNavigatorPeriod Text="YTD"></RangeNavigatorPeriod>
<RangeNavigatorPeriod Interval="1" IntervalType="RangeIntervalType.Years"
Text="1Y"></RangeNavigatorPeriod>
<RangeNavigatorPeriod Interval="2" IntervalType="RangeIntervalType.Years"
Text="2Y"></RangeNavigatorPeriod>
<RangeNavigatorPeriod Text="All"></RangeNavigatorPeriod>
</RangeNavigatorPeriods>
</RangeNavigatorPeriodSelectorSettings>
<RangeNavigatorSeriesCollection>
<RangeNavigatorSeries DataSource="@StockInfo" XName="Date"
Type="RangeNavigatorType.Area" YName="Close"></RangeNavigatorSeries>
</RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code {
public class StockDetails

```

```
{
public DateTime Date { get; set; }
public double Close { get; set; }
}
public DateTime[] Value = new DateTime[] { new DateTime(2009, 01, 01), new
DateTime(2010, 01, 01) };
public List<StockDetails> StockInfo = new List<StockDetails>
{
new StockDetails { Date = new DateTime(2005, 01, 01), Close = 21 },
new StockDetails { Date = new DateTime(2006, 01, 01), Close = 24 },
new StockDetails { Date = new DateTime(2007, 01, 01), Close = 36 },
new StockDetails { Date = new DateTime(2008, 01, 01), Close = 38 },
new StockDetails { Date = new DateTime(2009, 01, 01), Close = 54 },
new StockDetails { Date = new DateTime(2010, 01, 01), Close = 57 },
new StockDetails { Date = new DateTime(2011, 01, 01), Close = 70 }
};
}
```

See Also

- [Disable Range Selector](#)

Labels in Blazor Range Selector Component

Multi-level labels

The multi-level labels for the Range Selector can be enabled by setting the [EnableGrouping](#) property to **true**. This is restricted to the DateTime axis alone.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfRangeNavigator DataSource="@StockInfo" XName="X" YName="Y"
LabelPosition="AxisPosition.Outside"
EnableGrouping="true" IntervalType="RangeIntervalType.Quarter"
Value="@Value" ValueType="RangeValueType.DateTime">
<RangeNavigatorRangeTooltipSettings
Enable="true"></RangeNavigatorRangeTooltipSettings>
</SfRangeNavigator>
@code {
public class StockDetails
{
public DateTime X { get; set; }
public double Y { get; set; }
}
public DateTime[] Value = new DateTime[] { new DateTime(2005, 11, 01), new
DateTime(2006, 01, 01) };
public List<StockDetails> StockInfo = new List<StockDetails> {
new StockDetails { X = new DateTime(2005, 01, 01), Y = 21 },
new StockDetails { X = new DateTime(2005, 03, 01), Y = 24 },
new StockDetails { X = new DateTime(2005, 05, 01), Y = 36 },
new StockDetails { X = new DateTime(2006, 07, 01), Y = 38 },
new StockDetails { X = new DateTime(2006, 08, 01), Y = 54 },
new StockDetails { X = new DateTime(2006, 09, 01), Y = 57 },
new StockDetails { X = new DateTime(2006, 11, 01), Y = 70 }
};
}
```

```
}

```



Grouping

The multi-level labels can be grouped using the [GroupBy](#) property with the following interval types:

- Auto
- Years
- Quarter
- Months
- Weeks
- Days
- Hours
- Minutes
- Seconds

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfRangeNavigator DataSource="@StockInfo" XName="X" YName="Y"
LabelPosition="AxisPosition.Outside"
EnableGrouping="true" GroupBy="RangeIntervalType.Years"
IntervalType="RangeIntervalType.Quarter"
Value="@Value" ValueType="RangeValueType.DateTime">
  <RangeNavigatorRangeTooltipSettings
  Enable="true"></RangeNavigatorRangeTooltipSettings>
</SfRangeNavigator>
@code {
public class StockDetails
{
public DateTime X { get; set; }
public double Y { get; set; }
}
public DateTime[] Value = new DateTime[] { new DateTime(2005, 11, 01), new
DateTime(2006, 01, 01) };
public List<StockDetails> StockInfo = new List<StockDetails> {
new StockDetails { X = new DateTime(2005, 01, 01), Y = 21 },
new StockDetails { X = new DateTime(2005, 03, 01), Y = 24 },
new StockDetails { X = new DateTime(2005, 05, 01), Y = 36 },
new StockDetails { X = new DateTime(2006, 07, 01), Y = 38 },
new StockDetails { X = new DateTime(2006, 08, 01), Y = 54 },
new StockDetails { X = new DateTime(2006, 09, 01), Y = 57 },
new StockDetails { X = new DateTime(2006, 11, 01), Y = 70 }
};
}
```



Smart labels

The [LabelIntersectAction](#) property is used to avoid overlapping of labels. The following code sample shows the setting of [LabelIntersectAction](#) property to **Hide**.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfRangeNavigator Value="@Value" ValueType="RangeValueType.DateTime"
LabelFormat="yyyy/M/d"
LabelIntersectAction="RangeLabelIntersectAction.Hide">
<RangeNavigatorRangeTooltipSettings
Enable="true"></RangeNavigatorRangeTooltipSettings>
<RangeNavigatorSeriesCollection>
<RangeNavigatorSeries DataSource="@DataSource" XName="X"
Type="RangeNavigatorType.StepLine"
YName="Y" Width="2"></RangeNavigatorSeries>
</RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code {
public class SampleData
{
public DateTime X { get; set; }
public double Y { get; set; }
}
public DateTime[] Value = new DateTime[] { new DateTime(2006, 01, 01), new
DateTime(2008, 01, 01) };
public List<SampleData> DataSource = new List<SampleData> {
new SampleData { X = new DateTime(2005, 01, 01), Y = 21 },
new SampleData { X = new DateTime(2006, 01, 01), Y = 24 },
new SampleData { X = new DateTime(2007, 01, 01), Y = 36 },
new SampleData { X = new DateTime(2008, 01, 01), Y = 38 },
new SampleData { X = new DateTime(2009, 01, 01), Y = 54 },
new SampleData { X = new DateTime(2010, 01, 01), Y = 57 },
new SampleData { X = new DateTime(2011, 01, 01), Y = 70 }
};
}
```



Position

By default, the labels can be placed outside the Range Selector. It can also be placed inside the Range Selector using the [LabelPosition](#) property.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
```



```

<SfRangeNavigator Value="@Value" ValueType="RangeValueType.DateTime"
LabelPosition="AxisPosition.Inside">
  <RangeNavigatorSeriesCollection>
    <RangeNavigatorSeries DataSource="@StockInfo" XName="X"
Type="RangeNavigatorType.StepLine" YName="Y">
    </RangeNavigatorSeries>
  </RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code {
public class StockDetails
{
public DateTime X { get; set; }
public double Y { get; set; }
}
public DateTime[] Value = new DateTime[] { new DateTime(2006, 01, 01), new
DateTime(2008, 01, 01) };
public List<StockDetails> StockInfo = new List<StockDetails> {
new StockDetails { X = new DateTime(2005, 01, 01), Y = 21 },
new StockDetails { X = new DateTime(2006, 01, 01), Y = 24 },
new StockDetails { X = new DateTime(2007, 01, 01), Y = 36 },
new StockDetails { X = new DateTime(2008, 01, 01), Y = 38 },
new StockDetails { X = new DateTime(2009, 01, 01), Y = 54 },
new StockDetails { X = new DateTime(2010, 01, 01), Y = 57 },
new StockDetails { X = new DateTime(2011, 01, 01), Y = 70 }
};
}

```



Labels Customization

The font size, color, family, etc. can be customized using the [RangeNavigatorLabelStyle](#) setting.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfRangeNavigator DataSource="@StockInfo" XName="X" YName="Y" Value="@Value"
LabelFormat="MMM"
IntervalType="RangeIntervalType.Months" ValueType="RangeValueType.DateTime">
  <RangeNavigatorLabelStyle Color="red" Size="10"></RangeNavigatorLabelStyle>
</SfRangeNavigator>
@code {
public class StockDetails
{
public DateTime X { get; set; }
public double Y { get; set; }
}
public DateTime[] Value = new DateTime[] { new DateTime(2005, 11, 01), new
DateTime(2006, 01, 01) };
public List<StockDetails> StockInfo = new List<StockDetails> {
new StockDetails { X = new DateTime(2005, 01, 01), Y = 21 },

```

```

new StockDetails { X = new DateTime(2005, 03, 01), Y = 24 },
new StockDetails { X = new DateTime(2005, 05, 01), Y = 36 },
new StockDetails { X = new DateTime(2006, 07, 01), Y = 38 },
new StockDetails { X = new DateTime(2006, 08, 01), Y = 54 },
new StockDetails { X = new DateTime(2006, 09, 01), Y = 57 },
new StockDetails { X = new DateTime(2006, 11, 01), Y = 70 }
};
}

```



Grid Lines and Tick Lines in Blazor Range Selector Component

Gridline Customization

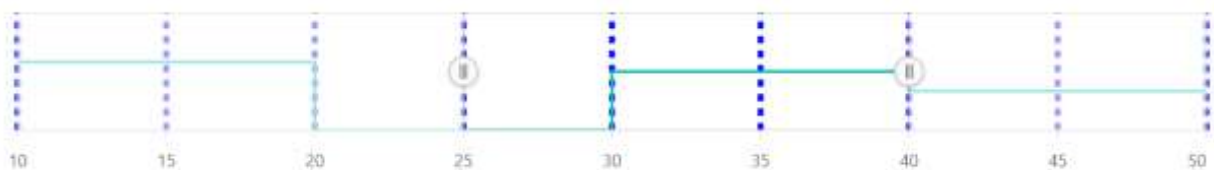
The gridlines indicate axis divisions by drawing the chart plot. Gridlines include helpful cues to the user, particularly for large or complicated charts. The [Width](#), [Color](#), and [DashArray](#) of the major gridlines can be customized by using the [RangeNavigatorMajorGridLines](#) setting.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfRangeNavigator Value="@Value" ValueType="RangeValueType.Double">
  <RangeNavigatorMajorGridLines Color="Blue" Width="4"
  DashArray="5,5"></RangeNavigatorMajorGridLines>
  <RangeNavigatorSeriesCollection>
    <RangeNavigatorSeries DataSource="@StockInfo" XName="Date"
    Type="RangeNavigatorType.StepLine" YName="Close">
  </RangeNavigatorSeries>
  </RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code {
  public class StockDetails
  {
    public double Date { get; set; }
    public double Close { get; set; }
  }
  public int[] Value = new int[] { 25, 40 };
  public List<StockDetails> StockInfo = new List<StockDetails> {
    new StockDetails { Date= 10, Close= 35 },
    new StockDetails { Date= 20, Close= 28 },
    new StockDetails { Date= 30, Close= 34 },
    new StockDetails { Date= 40, Close= 32 },
    new StockDetails { Date= 50, Close= 40 }
  };
}

```



Tickline Customization

Ticklines are the small lines which is drawn on the axis line representing the axis labels. Ticklines will be drawn outside the axis by default. The [Width](#), [Color](#), and [Height](#) of the major ticklines can be customized by using the [RangeNavigatorMajorTickLines](#) setting.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfRangeNavigator Value="@Value" ValueType="RangeValueType.Double">
  <RangeNavigatorMajorTickLines Color="Red"
  Width="3"></RangeNavigatorMajorTickLines>
  <RangeNavigatorSeriesCollection>
    <RangeNavigatorSeries DataSource="@StockInfo" XName="Date"
    Type="RangeNavigatorType.StepLine" YName="Close">
  </RangeNavigatorSeries>
  </RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code {
public class StockDetails
{
public double Date { get; set; }
public double Close { get; set; }
}
public int[] Value = new int[] { 25, 40 };
public List<StockDetails> StockInfo = new List<StockDetails> {
new StockDetails { Date= 10, Close= 35 },
new StockDetails { Date= 20, Close= 28 },
new StockDetails { Date= 30, Close= 34 },
new StockDetails { Date= 40, Close= 32 },
new StockDetails { Date= 50, Close= 40 }
};
}
```



Customization in Blazor Range Selector Component

Navigator Appearance

The Range Selector can be customized by using the [RangeNavigatorStyleSettings](#). The [SelectedRegionColor](#) property is used to specify the color for the selected region, whereas the [UnselectedRegionColor](#) property is used to specify the color for the unselected region.

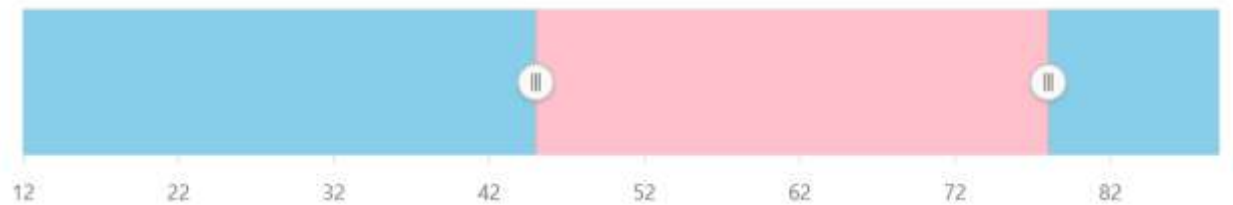
ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfRangeNavigator Value="@Value">
  <RangeNavigatorRangeTooltipSettings
  Enable="true"></RangeNavigatorRangeTooltipSettings>
  <RangeNavigatorStyleSettings UnselectedRegionColor="skyblue"
  SelectedRegionColor="pink"></RangeNavigatorStyleSettings>
  <RangeNavigatorSeriesCollection>
```

```

<RangeNavigatorSeries DataSource="@StockInfo" XName="Date"
Type="RangeNavigatorType.Area" YName="Close">
</RangeNavigatorSeries>
</RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code{
public class StockDetails
{
public double Date { get; set; }
public double Close { get; set; }
}
public List<StockDetails> StockInfo = new List<StockDetails> {
new StockDetails { Date = 12, Close = 28 },
new StockDetails { Date = 34, Close = 44 },
new StockDetails { Date = 45, Close = 48 },
new StockDetails { Date = 56, Close = 50 },
new StockDetails { Date = 67, Close = 66 },
new StockDetails { Date = 78, Close = 78 },
new StockDetails { Date = 89, Close = 84 },
};
public int[] Value = new int[] { 45, 78 };
}

```



Thumb

The [RangeNavigatorThumbSettings](#) allows to customize the border, fill color, size, and type of thumb using the [RangeNavigatorThumbBorder](#), [Fill](#), [Height](#), [Width](#), and [Type](#) properties. Thumbs can be of two shapes: **Circle** and **Rectangle**.

ASPX-CS

```

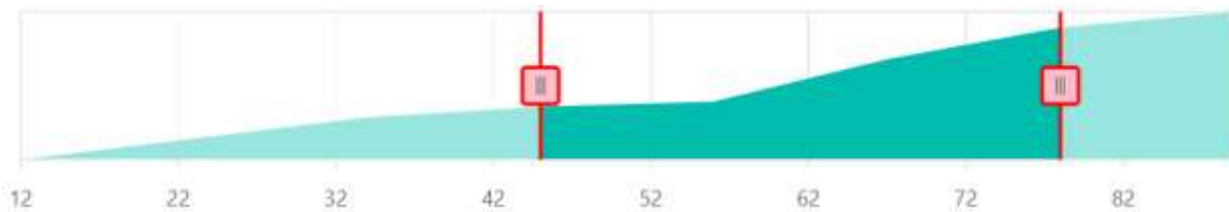
@using Syncfusion.Blazor.Charts
<SfRangeNavigator Value="@Value">
<RangeNavigatorStyleSettings>
<RangeNavigatorThumbSettings Type="ThumbType.Rectangle" Fill="pink">
<RangeNavigatorThumbBorder Width="2"
Color="red"></RangeNavigatorThumbBorder>
</RangeNavigatorThumbSettings>
</RangeNavigatorStyleSettings>
<RangeNavigatorRangeTooltipSettings
Enable="true"></RangeNavigatorRangeTooltipSettings>
<RangeNavigatorSeriesCollection>
<RangeNavigatorSeries DataSource="@StockInfo" XName="Date"
Type="RangeNavigatorType.Area" YName="Close">
</RangeNavigatorSeries>
</RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code{

```

```

public class StockDetails
{
    public double Date { get; set; }
    public double Close { get; set; }
}
public List<StockDetails> StockInfo = new List<StockDetails> {
    new StockDetails { Date = 12, Close = 28 },
    new StockDetails { Date = 34, Close = 44 },
    new StockDetails { Date = 45, Close = 48 },
    new StockDetails { Date = 56, Close = 50 },
    new StockDetails { Date = 67, Close = 66 },
    new StockDetails { Date = 78, Close = 78 },
    new StockDetails { Date = 89, Close = 84 },
};
public int[] Value = new int[] { 45, 78 };
}

```



Border

Using the [RangeNavigatorBorder](#), the [Width](#) and the [Color](#) of the Range Selector border can be customized.

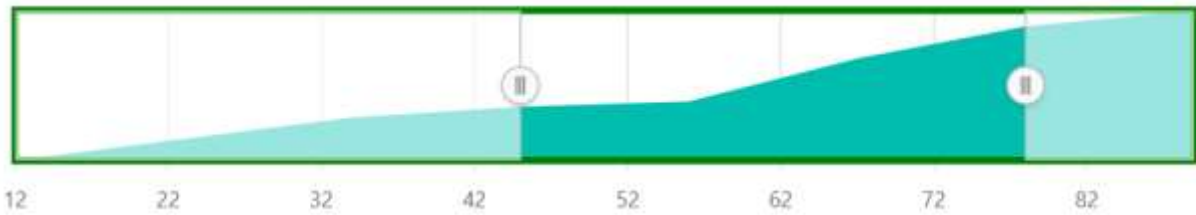
ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfRangeNavigator Value="@Value">
  <RangeNavigatorBorder Width="4" Color="Green"></RangeNavigatorBorder>
  <RangeNavigatorRangeTooltipSettings
    Enable="true"></RangeNavigatorRangeTooltipSettings>
  <RangeNavigatorSeriesCollection>
    <RangeNavigatorSeries DataSource="@StockInfo" XName="Date"
      Type="RangeNavigatorType.Area" YName="Close">
    </RangeNavigatorSeries>
  </RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code{
    public class StockDetails
    {
        public double Date { get; set; }
        public double Close { get; set; }
    }
    public List<StockDetails> StockInfo = new List<StockDetails> {
        new StockDetails { Date = 12, Close = 28 },
        new StockDetails { Date = 34, Close = 44 },
        new StockDetails { Date = 45, Close = 48 },
        new StockDetails { Date = 56, Close = 50 },
        new StockDetails { Date = 67, Close = 66 },
        new StockDetails { Date = 78, Close = 78 },
        new StockDetails { Date = 89, Close = 84 },
    }
}

```

```
};
public int[] Value = new int[] { 45, 78 };
}
```



Snapping

The [AllowSnapping](#) property toggles the placement of the slider exactly to the left or right at the nearest interval.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfRangeNavigator Value="@Value" AllowSnapping="true">
  <RangeNavigatorRangeTooltipSettings
    Enable="true"></RangeNavigatorRangeTooltipSettings>
  <RangeNavigatorSeriesCollection>
    <RangeNavigatorSeries DataSource="@StockInfo" XName="Date"
      Type="RangeNavigatorType.Area" YName="Close">
    </RangeNavigatorSeries>
  </RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code{
public class StockDetails
{
public double Date { get; set; }
public double Close { get; set; }
}
public List<StockDetails> StockInfo = new List<StockDetails> {
new StockDetails { Date = 12, Close = 28 },
new StockDetails { Date = 34, Close = 44 },
new StockDetails { Date = 45, Close = 48 },
new StockDetails { Date = 56, Close = 50 },
new StockDetails { Date = 67, Close = 66 },
new StockDetails { Date = 78, Close = 78 },
new StockDetails { Date = 89, Close = 84 },
};
public int[] Value = new int[] { 45, 78 };
}
```

Animation

Animation for the Range Selector is enabled by default. The speed of the animation can be controlled using the [AnimationDuration](#) property. The default value of the [AnimationDuration](#) property is 500 milliseconds.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfRangeNavigator Value="@Value" AnimationDuration="3000">
```

```

<RangeNavigatorRangeTooltipSettings
Enable="true"></RangeNavigatorRangeTooltipSettings>
<RangeNavigatorSeriesCollection>
<RangeNavigatorSeries DataSource="@StockInfo" XName="Date"
Type="RangeNavigatorType.Area" YName="Close">
</RangeNavigatorSeries>
</RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code{
public class StockDetails
{
public double Date { get; set; }
public double Close { get; set; }
}
public List<StockDetails> StockInfo = new List<StockDetails> {
new StockDetails { Date = 12, Close = 28 },
new StockDetails { Date = 34, Close = 44 },
new StockDetails { Date = 45, Close = 48 },
new StockDetails { Date = 56, Close = 50 },
new StockDetails { Date = 67, Close = 66 },
new StockDetails { Date = 78, Close = 78 },
new StockDetails { Date = 89, Close = 84 },
};
public int[] Value = new int[] { 45, 78 };
}

```

See Also

- [Grid and Tick Lines](#)
- [Labels](#)

Tooltip in Blazor Range Selector Component

<!-- markdownlint-disable MD036 -->

The tooltip for sliders are supported by the Range Selector. Sliders are used in the Range Selector to select data from a specific range. The tooltip displays the selected start and end values.

<!-- markdownlint-disable MD013 -->

Enable tooltip

The tooltip can be used to display information about the selected data and it is enabled by setting the [Enable](#) property to **true**.

ASPX-CS

```

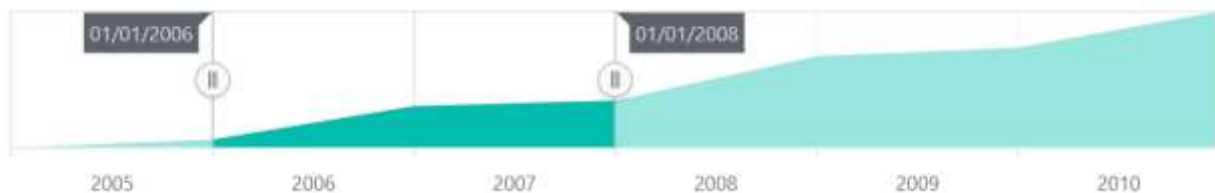
@using Syncfusion.Blazor.Charts
<SfRangeNavigator Value="@Value" ValueType="RangeValueType.DateTime">
<RangeNavigatorRangeTooltipSettings Enable="true"
DisplayMode="TooltipDisplayMode.Always">
</RangeNavigatorRangeTooltipSettings>
<RangeNavigatorSeriesCollection>
<RangeNavigatorSeries DataSource="@StockInfo" XName="Date"
Type="RangeNavigatorType.Area" YName="Close">
</RangeNavigatorSeries>
</RangeNavigatorSeriesCollection>

```

```

</SfRangeNavigator>
@code {
public class StockDetails
{
public DateTime Date { get; set; }
public double Close { get; set; }
}
public DateTime[] Value = new DateTime[] { new DateTime(2006, 01, 01), new
DateTime(2008, 01, 01) };
public List<StockDetails> StockInfo = new List<StockDetails>
{
new StockDetails { Date = new DateTime(2005, 01, 01), Close = 21 },
new StockDetails { Date = new DateTime(2006, 01, 01), Close = 24 },
new StockDetails { Date = new DateTime(2007, 01, 01), Close = 36 },
new StockDetails { Date = new DateTime(2008, 01, 01), Close = 38 },
new StockDetails { Date = new DateTime(2009, 01, 01), Close = 54 },
new StockDetails { Date = new DateTime(2010, 01, 01), Close = 57 },
new StockDetails { Date = new DateTime(2011, 01, 01), Close = 70 }
};
}

```



Tooltip Customization

The tooltip can be customized using the following properties:

- [Enable](#) - Customizes the visibility of the tooltip.
- [DisplayMode](#) - Customizes the display mode of the tooltip.
- [Fill](#) - Customizes the background color of the tooltip.
- [Opacity](#) - Customizes the opacity of the tooltip.
- [Format](#) - Customizes the format of the tooltip text.
- [RangeNavigatorTooltipTextStyle](#) - Customizes the [Size](#), the [Color](#), the [FontFamily](#), the [FontStyle](#), the [FontWeight](#), the [TextAlignment](#) and the [TextOverflow](#) of the tooltip text.
- [RangeNavigatorTooltipBorder](#) - Customizes the [Width](#) and the [Color](#) of the tooltip border.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfRangeNavigator Value="@Value" ValueType="RangeValueType.DateTime">
<RangeNavigatorRangeTooltipSettings Enable="true" Fill="red" Opacity="0.6"
DisplayMode="TooltipDisplayMode.Always">
<RangeNavigatorTooltipTextStyle Color="blue" Size="12px"
FontStyle="italic"></RangeNavigatorTooltipTextStyle>
</RangeNavigatorRangeTooltipSettings>
<RangeNavigatorSeriesCollection>
<RangeNavigatorSeries DataSource="@StockInfo" XName="Date"
Type="RangeNavigatorType.Area" YName="Close">
</RangeNavigatorSeries>

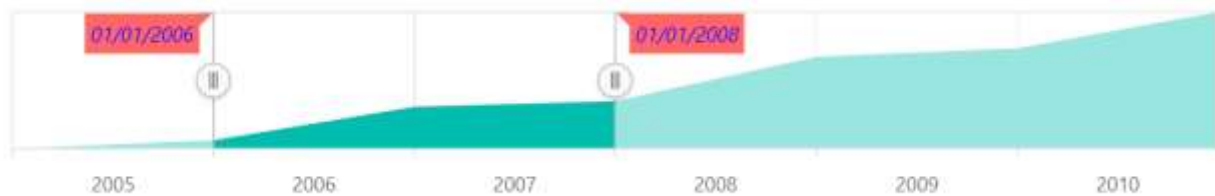
```



```

</RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code {
public class StockDetails
{
public DateTime Date { get; set; }
public double Close { get; set; }
}
public DateTime[] Value = new DateTime[] { new DateTime(2006, 01, 01), new
DateTime(2008, 01, 01) };
public List<StockDetails> StockInfo = new List<StockDetails>
{
new StockDetails { Date = new DateTime(2005, 01, 01), Close = 21 },
new StockDetails { Date = new DateTime(2006, 01, 01), Close = 24 },
new StockDetails { Date = new DateTime(2007, 01, 01), Close = 36 },
new StockDetails { Date = new DateTime(2008, 01, 01), Close = 38 },
new StockDetails { Date = new DateTime(2009, 01, 01), Close = 54 },
new StockDetails { Date = new DateTime(2010, 01, 01), Close = 57 },
new StockDetails { Date = new DateTime(2011, 01, 01), Close = 70 }
};
}

```



Label Format

The [LabelFormat](#) property in the tooltip is used to format and parse the date to all globalize formats.

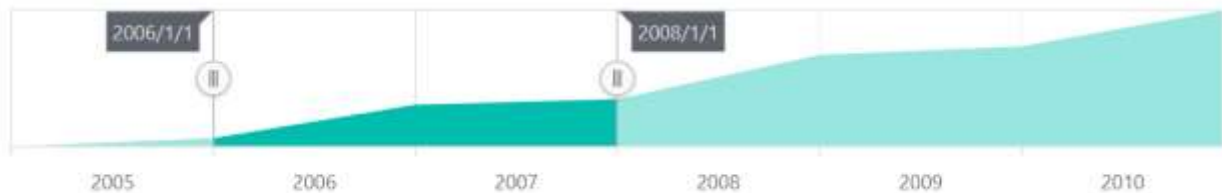
ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfRangeNavigator Value="@Value" ValueType="RangeValueType.DateTime">
<RangeNavigatorRangeTooltipSettings Enable="true"
DisplayMode="TooltipDisplayMode.Always" Format="y/M/d">
</RangeNavigatorRangeTooltipSettings>
<RangeNavigatorSeriesCollection>
<RangeNavigatorSeries DataSource="@StockInfo" XName="Date"
Type="RangeNavigatorType.Area" YName="Close">
</RangeNavigatorSeries>
</RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code {
public class StockDetails
{
public DateTime Date { get; set; }
public double Close { get; set; }
}
public DateTime[] Value = new DateTime[] { new DateTime(2006, 01, 01), new
DateTime(2008, 01, 01) };
public List<StockDetails> StockInfo = new List<StockDetails>
{
new StockDetails { Date = new DateTime(2005, 01, 01), Close = 21 },

```

```
new StockDetails { Date = new DateTime(2006, 01, 01), Close = 24 },
new StockDetails { Date = new DateTime(2007, 01, 01), Close = 36 },
new StockDetails { Date = new DateTime(2008, 01, 01), Close = 38 },
new StockDetails { Date = new DateTime(2009, 01, 01), Close = 54 },
new StockDetails { Date = new DateTime(2010, 01, 01), Close = 57 },
new StockDetails { Date = new DateTime(2011, 01, 01), Close = 70 }
};
}
```



The following table shows the results of applying some common date and 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 the day format.
new Date(2000, 03, 10)	yMd	04/10/2000	The date is displayed in the 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.

RTL in Blazor Range Selector Component

The Range Selector supports right-to-left (RTL), which can be enabled with the [EnableRtl](#) property.

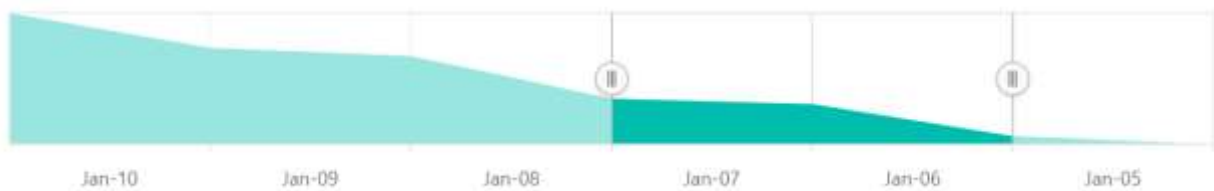
ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfRangeNavigator Value="@Value" ValueType="RangeValueType.DateTime"
LabelFormat="MMM-yy" EnableRtl="true">
<RangeNavigatorSeriesCollection>
<RangeNavigatorSeries DataSource="@StockInfo" XName="Date"
Type="RangeNavigatorType.Area" YName="Close">
</RangeNavigatorSeries>
</RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code {
```

```

public class StockDetails
{
    public DateTime Date { get; set; }
    public double Close { get; set; }
}
public DateTime[] Value = new DateTime[] { new DateTime(2006, 01, 01), new
DateTime(2008, 01, 01) };
public List<StockDetails> StockInfo = new List<StockDetails>
{
    new StockDetails { Date = new DateTime(2005, 01, 01), Close = 21, },
    new StockDetails { Date = new DateTime(2006, 01, 01), Close = 24, },
    new StockDetails { Date = new DateTime(2007, 01, 01), Close = 36, },
    new StockDetails { Date = new DateTime(2008, 01, 01), Close = 38, },
    new StockDetails { Date= new DateTime(2009, 01, 01), Close = 54, },
    new StockDetails { Date = new DateTime(2010, 01, 01), Close = 57, },
    new StockDetails { Date = new DateTime(2011, 01, 01), Close = 70, }
};
}

```



Print and Export in Blazor Range Selector Component

Print

The rendered Range Selector can be printed directly from the browser by calling the public method `PrintAsync`.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
@using Syncfusion.Blazor.Buttons
<SfRangeNavigator @ref="RangeObj" Value="@Value"
ValueType="RangeValueType.DateTime">
    <RangeNavigatorSeriesCollection>
        <RangeNavigatorSeries DataSource="@StockInfo" XName="Date"
Type="RangeNavigatorType.StepLine" YName="Close">
        </RangeNavigatorSeries>
    </RangeNavigatorSeriesCollection>
</SfRangeNavigator>
<SfButton Id="button" Content="Print" @onclick="@Click" IsPrimary="true"
CssClass="e-flat">
</SfButton>
@code {
    public SfRangeNavigator RangeObj;
    public class StockDetails
    {
        public DateTime Date { get; set; }
        public double Close { get; set; }
    }
}

```

```

public DateTime[] Value = new DateTime[] { new DateTime(2006, 01, 01), new
DateTime(2008, 01, 01) };
public List<StockDetails> StockInfo = new List<StockDetails> {
new StockDetails { Date = new DateTime(2005, 01, 01), Close = 21 },
new StockDetails { Date = new DateTime(2006, 01, 01), Close = 24 },
new StockDetails { Date = new DateTime(2007, 01, 01), Close = 36 },
new StockDetails { Date = new DateTime(2008, 01, 01), Close = 38 },
new StockDetails { Date = new DateTime(2009, 01, 01), Close = 54 },
new StockDetails { Date = new DateTime(2010, 01, 01), Close = 57 },
new StockDetails { Date = new DateTime(2011, 01, 01), Close = 70 }
};
public async Task Click(MouseEventArgs args)
{
await RangeObj.PrintAsync();
}
}

```

Export

The rendered Range Selector can be exported to **JPEG, PNG, SVG, or PDF** format by using the **ExportAsync** method in the Range Selector. This method contains the following parameters:

- **Type** - To specify the export type. The component can be exported to **JPEG, PNG, SVG, or PDF** format.
- **File name** - To specify the file name to export.
- **Orientation** - To specify the orientation type. This is applicable only for PDF export type. It is an optional parameter.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
@using Syncfusion.Blazor.Buttons
<SfRangeNavigator @ref="RangeObj" Value="@Value"
ValueType="RangeValueType.DateTime">
<RangeNavigatorSeriesCollection>
<RangeNavigatorSeries DataSource="@StockInfo" XName="Date"
Type="RangeNavigatorType.StepLine" YName="Close">
</RangeNavigatorSeries>
</RangeNavigatorSeriesCollection>
</SfRangeNavigator>
<SfButton Id="button" Content="Export" @onclick="@Click" IsPrimary="true"
CssClass="e-flat">
</SfButton>
@code {
public SfRangeNavigator RangeObj;
public class StockDetails
{
public DateTime Date { get; set; }
public double Close { get; set; }
}
public DateTime[] Value = new DateTime[] { new DateTime(2006, 01, 01), new
DateTime(2008, 01, 01) };
public List<StockDetails> StockInfo = new List<StockDetails> {
new StockDetails { Date = new DateTime(2005, 01, 01), Close = 21 },
new StockDetails { Date = new DateTime(2006, 01, 01), Close = 24 },

```

```

new StockDetails { Date = new DateTime(2007, 01, 01), Close = 36 },
new StockDetails { Date = new DateTime(2008, 01, 01), Close = 38 },
new StockDetails { Date = new DateTime(2009, 01, 01), Close = 54 },
new StockDetails { Date = new DateTime(2010, 01, 01), Close = 57 },
new StockDetails { Date = new DateTime(2011, 01, 01), Close = 70 }
};
public async Task Click(MouseEventArgs args)
{
    await RangeObj.ExportAsync(ExportType.PDF, "pngImage",
    Syncfusion.PdfExport.PdfPageOrientation.Landscape);
}
}

```

Events in Blazor Range Selector Component

This section describes about the Range Selector component's events, that is triggered when appropriate actions are performed. The events should be provided to the Range Selector through the **RangeNavigatorEvents** component.

The Range Selector component supports the following events.

- [Loaded](#)
- [OnPrintCompleted](#)
- [Changed](#)
- [Resized](#)
- [LabelRender](#)
- [TooltipRender](#)
- [SelectorRender](#)

Loaded

The [Loaded](#) event triggers, after the Range Selector is rendered.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfRangeNavigator Value="@Value" ValueType="RangeValueType.DateTime"
IntervalType="RangeIntervalType.Years">
<RangeNavigatorEvents Loaded="RangeNavigatorLoaded"></RangeNavigatorEvents>
<RangeNavigatorSeriesCollection>
<RangeNavigatorSeries DataSource="@StockInfo" XName="Date"
Type="RangeNavigatorType.Line" YName="Close">
</RangeNavigatorSeries>
</RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code {
    public class StockDetails
    {
        public DateTime Date { get; set; }
        public double Close { get; set; }
    }
    public List<StockDetails> StockInfo = new List<StockDetails> {
        new StockDetails { Date = new DateTime(2005, 01, 01), Close = 21 },
        new StockDetails { Date = new DateTime(2006, 01, 01), Close = 24 },
        new StockDetails { Date = new DateTime(2007, 01, 01), Close = 36 },
        new StockDetails { Date = new DateTime(2008, 01, 01), Close = 38 },
    }
}

```

```

new StockDetails { Date = new DateTime(2009, 01, 01), Close = 54 },
new StockDetails { Date = new DateTime(2010, 01, 01), Close = 57 },
new StockDetails { Date = new DateTime(2011, 01, 01), Close = 70 }
};
public DateTime[] Value = new DateTime[] {
new DateTime(2006, 01, 01), new DateTime(2008, 01, 01)
};
public void RangeNavigatorLoaded(RangeLoadedEventArgs args)
{
// Here you can customize your code.
}
}

```

OnPrintCompleted

The [OnPrintCompleted](#) event triggers, after the Range Selector is printed.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
@using Syncfusion.Blazor.Buttons
<SfRangeNavigator @ref="RangeObj" Value="@Value"
ValueType="RangeValueType.DateTime" IntervalType="RangeIntervalType.Years">
<RangeNavigatorEvents
OnPrintCompleted="PrintCompleted"></RangeNavigatorEvents>
<RangeNavigatorSeriesCollection>
<RangeNavigatorSeries DataSource="@StockInfo" XName="Date"
Type="RangeNavigatorType.Line" YName="Close">
</RangeNavigatorSeries>
</RangeNavigatorSeriesCollection>
</SfRangeNavigator>
<SfButton Id="button" Content="Print" @onclick="@Click" IsPrimary="true"
CssClass="e-flat">
</SfButton>
@code {
public SfRangeNavigator RangeObj;
public class StockDetails
{
public DateTime Date { get; set; }
public double Close { get; set; }
}
public List<StockDetails> StockInfo = new List<StockDetails> {
new StockDetails { Date = new DateTime(2005, 01, 01), Close = 21 },
new StockDetails { Date = new DateTime(2006, 01, 01), Close = 24 },
new StockDetails { Date = new DateTime(2007, 01, 01), Close = 36 },
new StockDetails { Date = new DateTime(2008, 01, 01), Close = 38 },
new StockDetails { Date = new DateTime(2009, 01, 01), Close = 54 },
new StockDetails { Date = new DateTime(2010, 01, 01), Close = 57 },
new StockDetails { Date = new DateTime(2011, 01, 01), Close = 70 }
};
public DateTime[] Value = new DateTime[] {
new DateTime(2006, 01, 01), new DateTime(2008, 01, 01)
};
public async Task Click(MouseEventArgs args)
{
await RangeObj.PrintAsync();
}
}

```

```
public void PrintCompleted(EventArgs args)
{
    // Here you can customize your code.
}
}
```

Changed

The [Changed](#) event triggers, whenever the slider position is changed. The following arguments are present in this event:

- [Start](#) - Specifies the start value.
- [End](#) - Specifies the end value.
- [ZoomFactor](#) - Specifies the zoom factor.
- [ZoomPosition](#) - Specifies the zoom position.
- [SelectedData](#) - The selected data collection can be accessed in this argument.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfRangeNavigator Value="@Value" ValueType="RangeValueType.DateTime"
IntervalType="RangeIntervalType.Years">
<RangeNavigatorEvents Changed="SliderChanged"></RangeNavigatorEvents>
<RangeNavigatorSeriesCollection>
<RangeNavigatorSeries DataSource="@StockInfo" XName="Date"
Type="RangeNavigatorType.Line" YName="Close">
</RangeNavigatorSeries>
</RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code {
public class StockDetails
{
    public DateTime Date { get; set; }
    public double Close { get; set; }
}
public List<StockDetails> StockInfo = new List<StockDetails> {
    new StockDetails { Date = new DateTime(2005, 01, 01), Close = 21 },
    new StockDetails { Date = new DateTime(2006, 01, 01), Close = 24 },
    new StockDetails { Date = new DateTime(2007, 01, 01), Close = 36 },
    new StockDetails { Date = new DateTime(2008, 01, 01), Close = 38 },
    new StockDetails { Date = new DateTime(2009, 01, 01), Close = 54 },
    new StockDetails { Date = new DateTime(2010, 01, 01), Close = 57 },
    new StockDetails { Date = new DateTime(2011, 01, 01), Close = 70 }
};
public DateTime[] Value = new DateTime[] {
    new DateTime(2006, 01, 01), new DateTime(2008, 01, 01)
};
public void SliderChanged(ChangedEventArgs args)
{
    // Here you can customize your code.
}
}
```

Resized

The [Resized](#) event triggers, when the browser window is resized. The following arguments are present in this event:

- [CurrentSize](#) - Specifies the current size for the Range Selector.
- [PreviousSize](#) - Specifies the previous size for the Range Selector.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfRangeNavigator Value="@Value" ValueType="RangeValueType.DateTime"
IntervalType="RangeIntervalType.Years">
<RangeNavigatorEvents
Resized="RangeNavigatorResized"></RangeNavigatorEvents>
<RangeNavigatorSeriesCollection>
<RangeNavigatorSeries DataSource="@StockInfo" XName="Date"
Type="RangeNavigatorType.Line" YName="Close">
</RangeNavigatorSeries>
</RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code {
public class StockDetails
{
public DateTime Date { get; set; }
public double Close { get; set; }
}
public List<StockDetails> StockInfo = new List<StockDetails> {
new StockDetails { Date = new DateTime(2005, 01, 01), Close = 21 },
new StockDetails { Date = new DateTime(2006, 01, 01), Close = 24 },
new StockDetails { Date = new DateTime(2007, 01, 01), Close = 36 },
new StockDetails { Date = new DateTime(2008, 01, 01), Close = 38 },
new StockDetails { Date = new DateTime(2009, 01, 01), Close = 54 },
new StockDetails { Date = new DateTime(2010, 01, 01), Close = 57 },
new StockDetails { Date = new DateTime(2011, 01, 01), Close = 70 }
};
public DateTime[] Value = new DateTime[] {
new DateTime(2006, 01, 01), new DateTime(2008, 01, 01)
};
public void RangeNavigatorResized(RangeResizeEventArgs args)
{
// Here you can customize your code.
}
}
```

LabelRender

Before rendering each axis label, the [LabelRender](#) event is triggered. The following arguments are present in this event:

- [Text](#) - Specifies the current axis label text.
- [Value](#) - Specifies the current axis label value.
- [Region](#) - Specifies the current axis label position.
- [LabelStyle](#) - Specifies the current axis label style.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfRangeNavigator Value="@Value" ValueType="RangeValueType.DateTime"
IntervalType="RangeIntervalType.Years">
  <RangeNavigatorEvents
    LabelRender="LabelCustomization"></RangeNavigatorEvents>
  <RangeNavigatorSeriesCollection>
    <RangeNavigatorSeries DataSource="@StockInfo" XName="Date"
      Type="RangeNavigatorType.Line" YName="Close">
    </RangeNavigatorSeries>
  </RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code {
public class StockDetails
{
public DateTime Date { get; set; }
public double Close { get; set; }
}
public List<StockDetails> StockInfo = new List<StockDetails> {
new StockDetails { Date = new DateTime(2005, 01, 01), Close = 21 },
new StockDetails { Date = new DateTime(2006, 01, 01), Close = 24 },
new StockDetails { Date = new DateTime(2007, 01, 01), Close = 36 },
new StockDetails { Date = new DateTime(2008, 01, 01), Close = 38 },
new StockDetails { Date = new DateTime(2009, 01, 01), Close = 54 },
new StockDetails { Date = new DateTime(2010, 01, 01), Close = 57 },
new StockDetails { Date = new DateTime(2011, 01, 01), Close = 70 }
};
public DateTime[] Value = new DateTime[] {
new DateTime(2006, 01, 01), new DateTime(2008, 01, 01)
};
public void LabelCustomization(RangeLabelRenderEventArgs args)
{
// Here you can customize your code.
}
}

```

TooltipRender

The [TooltipRender](#) event triggers before the tooltip is rendered. The following arguments are present in this event:

- [Text](#) - Specifies the current tooltip text.
- [TextStyle](#) - Specifies the current tooltip text style.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfRangeNavigator Value="@Value" ValueType="RangeValueType.DateTime"
IntervalType="RangeIntervalType.Years">
  <RangeNavigatorEvents
    TooltipRender="TooltipCustomization"></RangeNavigatorEvents>
  <RangeNavigatorSeriesCollection>
    <RangeNavigatorSeries DataSource="@StockInfo" XName="Date"
      Type="RangeNavigatorType.Line" YName="Close">
    </RangeNavigatorSeries>
  </RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code {
public class StockDetails
{
public DateTime Date { get; set; }
public double Close { get; set; }
}
public List<StockDetails> StockInfo = new List<StockDetails> {
new StockDetails { Date = new DateTime(2005, 01, 01), Close = 21 },
new StockDetails { Date = new DateTime(2006, 01, 01), Close = 24 },
new StockDetails { Date = new DateTime(2007, 01, 01), Close = 36 },
new StockDetails { Date = new DateTime(2008, 01, 01), Close = 38 },
new StockDetails { Date = new DateTime(2009, 01, 01), Close = 54 },
new StockDetails { Date = new DateTime(2010, 01, 01), Close = 57 },
new StockDetails { Date = new DateTime(2011, 01, 01), Close = 70 }
};
public DateTime[] Value = new DateTime[] {
new DateTime(2006, 01, 01), new DateTime(2008, 01, 01)
};
public void TooltipCustomization(RangeTooltipRenderEventArgs args)
{
// Here you can customize your code.
}
}

```

```

</RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code {
public class StockDetails
{
public DateTime Date { get; set; }
public double Close { get; set; }
}
public List<StockDetails> StockInfo = new List<StockDetails> {
new StockDetails { Date = new DateTime(2005, 01, 01), Close = 21 },
new StockDetails { Date = new DateTime(2006, 01, 01), Close = 24 },
new StockDetails { Date = new DateTime(2007, 01, 01), Close = 36 },
new StockDetails { Date = new DateTime(2008, 01, 01), Close = 38 },
new StockDetails { Date = new DateTime(2009, 01, 01), Close = 54 },
new StockDetails { Date = new DateTime(2010, 01, 01), Close = 57 },
new StockDetails { Date = new DateTime(2011, 01, 01), Close = 70 }
};
public DateTime[] Value = new DateTime[] {
new DateTime(2006, 01, 01), new DateTime(2008, 01, 01)
};
public void TooltipCustomization(RangeTooltipRenderEventArgs args)
{
// Here you can customize your code.
}
}

```

SelectorRender

The [SelectorRender](#) event triggers before the period selector is rendered. The following arguments are present in this event:

- [Content](#) - Specifies the content for the calendar in the period selector.
- [EnableCustomFormat](#) - Enables to show the content for the calendar in the period selector. By default it is true.
- [Selector](#) - Specifies the period selector collection.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfRangeNavigator Value="@Value" ValueType="RangeValueType.DateTime"
IntervalType="RangeIntervalType.Years">
<RangeNavigatorPeriodSelectorSettings>
<RangeNavigatorPeriods>
<RangeNavigatorPeriod Interval="1" IntervalType="RangeIntervalType.Months"
Text="1M"></RangeNavigatorPeriod>
<RangeNavigatorPeriod Interval="2" IntervalType="RangeIntervalType.Months"
Text="2M"></RangeNavigatorPeriod>
<RangeNavigatorPeriod Interval="6" IntervalType="RangeIntervalType.Months"
Text="6M"></RangeNavigatorPeriod>
<RangeNavigatorPeriod Text="YTD"></RangeNavigatorPeriod>
<RangeNavigatorPeriod Interval="1" IntervalType="RangeIntervalType.Years"
Text="1Y"></RangeNavigatorPeriod>
<RangeNavigatorPeriod Interval="2" IntervalType="RangeIntervalType.Years"
Text="2Y"></RangeNavigatorPeriod>
<RangeNavigatorPeriod Text="All"></RangeNavigatorPeriod>

```

```

</RangeNavigatorPeriods>
</RangeNavigatorPeriodSelectorSettings>
<RangeNavigatorEvents
SelectorRender="SelectorCustomization"></RangeNavigatorEvents>
<RangeNavigatorSeriesCollection>
<RangeNavigatorSeries DataSource="@StockInfo" XName="Date"
Type="RangeNavigatorType.Line" YName="Close">
</RangeNavigatorSeries>
</RangeNavigatorSeriesCollection>
</SfRangeNavigator>
@code {
public class StockDetails
{
public DateTime Date { get; set; }
public double Close { get; set; }
}
public List<StockDetails> StockInfo = new List<StockDetails> {
new StockDetails { Date = new DateTime(2005, 01, 01), Close = 21 },
new StockDetails { Date = new DateTime(2006, 01, 01), Close = 24 },
new StockDetails { Date = new DateTime(2007, 01, 01), Close = 36 },
new StockDetails { Date = new DateTime(2008, 01, 01), Close = 38 },
new StockDetails { Date = new DateTime(2009, 01, 01), Close = 54 },
new StockDetails { Date = new DateTime(2010, 01, 01), Close = 57 },
new StockDetails { Date = new DateTime(2011, 01, 01), Close = 70 }
};
public DateTime[] Value = new DateTime[] {
new DateTime(2006, 01, 01), new DateTime(2008, 01, 01)
};
public void SelectorCustomization(RangeSelectorRenderEventArgs args)
{
// Here you can customize your code.
}
}

```

Range Slider

<!-- markdownlint-disable MD024 -->

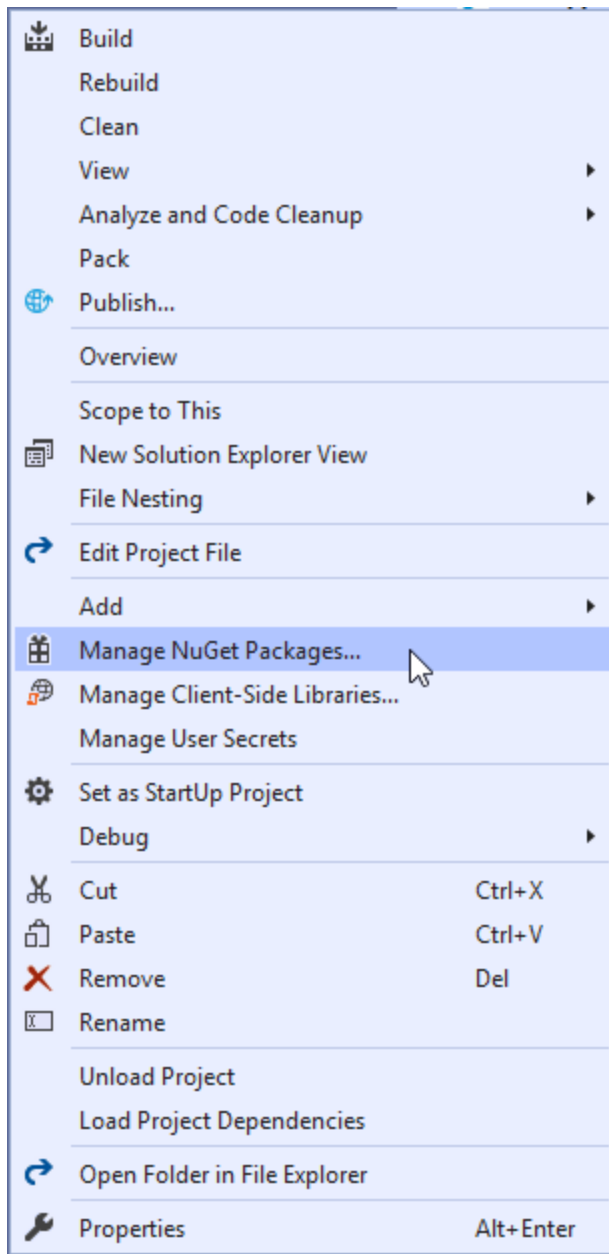
Getting Started with Blazor Range Slider Component

This section briefly explains how to include a simple **Range Slider** in your Blazor Server-Side application. Refer to the [Getting Started with Syncfusion Blazor for Server-Side in Visual Studio page](#) for the introduction and configuring the common specifications.

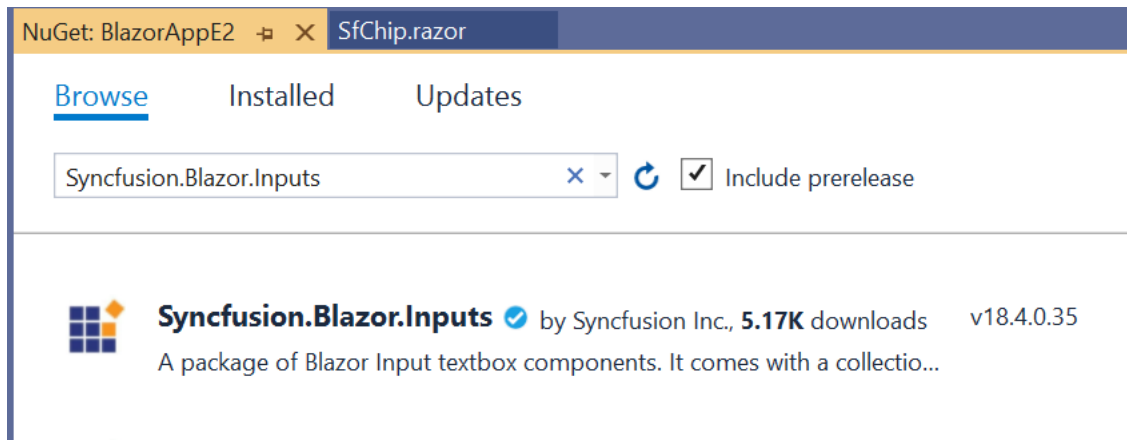
Importing Syncfusion Blazor component in the application

Using Syncfusion.Blazor NuGet Package [New standard]

1. Install **Syncfusion.Blazor.Inputs** NuGet package to the application by using the **NuGet Package Manager**. Refer to the Individual NuGet Packages section for the available NuGet packages.



2. Search Syncfusion.Blazor.Inputs keyword in the Browse tab and install Syncfusion.Blazor.Inputs NuGet package in the application.



- Once the installation process is completed, the Syncfusion Blazor Inputs package will be installed in the project. You can add the client-side style resources using NuGet package to the element of the `~/wwwroot/index.html` page in Blazor WebAssembly app or `~/Pages/_Host.cshtml` page in Blazor Server app.

HTML

```
<head>
....
....
<link href="_content/Syncfusion.Blazor/styles/bootstrap4.css"
rel="stylesheet" />
</head>
```

HTML

```
<head>
<link href="https://cdn.syncfusion.com/blazor/{{ site.blazorversion
}}/styles/bootstrap4.css" rel="stylesheet" />
</head>
```

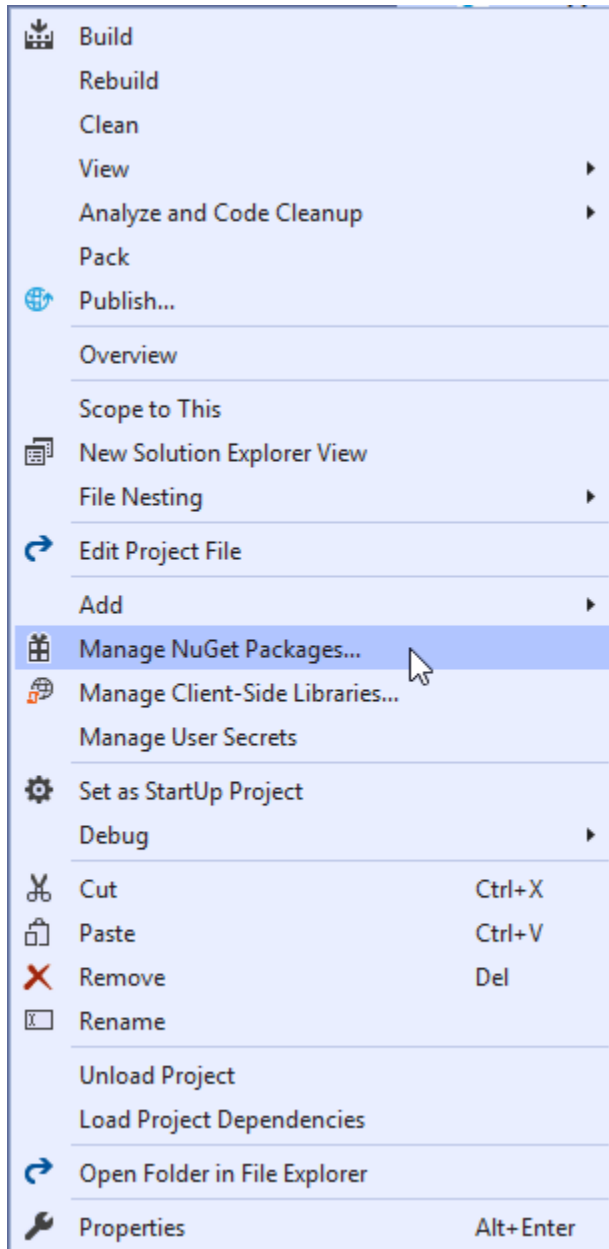
- For Internet Explorer 11 kindly refer the polyfills. Refer the [documentation](#) for more information.

HTML

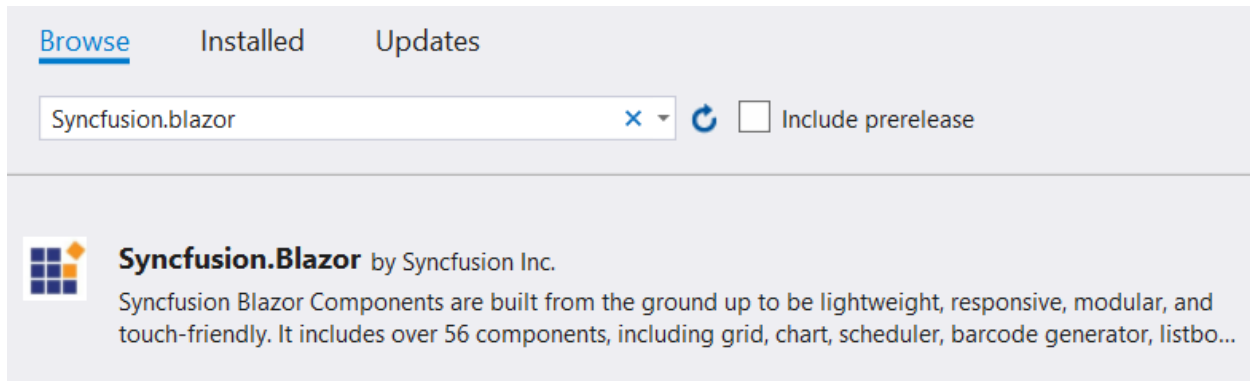
```
<head>
<link href="_content/Syncfusion.Blazor/styles/bootstrap4.css"
rel="stylesheet" />
<script
src="https://github.com/Daddoon/Blazor.Polyfill/releases/download/3.0.1/blaz
or.polyfill.min.js"></script>
</head>
```

Using Syncfusion.Blazor NuGet Package [Old standard]

1. Install **Syncfusion.Blazor** NuGet package to the application by using the **NuGet Package Manager**. Right-click the project and then select **Manage NuGet Packages**.



2. Search Syncfusion.Blazor keyword in the Browse tab and install Syncfusion.Blazor NuGet package in the application.



- Once the installation process is completed, the Syncfusion Blazor package will be installed in the project.

Warning: Syncfusion.Blazor package should not be installed along with [individual NuGet packages](#). Hence, you have to add the below Syncfusion.Blazor.Themes static web assets (styles) in the application.

You can add the client-side style resources through [CDN](#) or from [NuGet](#) package to the `<head>` element of the `~/wwwroot/index.html` page in Blazor WebAssembly app or `~/Pages/_Host.cshtml` page in Blazor Server app.

HTML

```
<head>
....
....
<link href="_content/Syncfusion.Blazor.Themes/bootstrap4.css"
rel="stylesheet" />
</head>
```

Warning: If you prefer the above new standard (individual NuGet packages), then skip this section. Using both old and new standards in the same application will throw ambiguous compilation errors.

Add Syncfusion Blazor service in Startup.cs (Server-side application)

Open the **Startup.cs** file and add services required by Syncfusion components using `services.AddSyncfusionBlazor()` method. Add this method in the `ConfigureServices` function as follows.

C#

```
using Syncfusion.Blazor;
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void ConfigureServices(IServiceCollection services)
        {
            ....
            ....
        }
    }
}
```

```
services.AddSyncfusionBlazor();
}
}
}
```

Add Syncfusion Blazor service in Program.cs (Client-side application)

Open the **Program.cs** file and add services required by Syncfusion components using `builder.services.AddSyncfusionBlazor()` method. Add this method in the **Main** function as follows.

C#

```
using Syncfusion.Blazor;
namespace BlazorApplication
{
    public class Program
    {
        public static async Task Main(string[] args)
        {
            ....
            ....
            builder.Services.AddSyncfusionBlazor();
        }
    }
}
```

To enable custom client side resource loading from CRG or CDN. You need to disable resource loading by **AddSyncfusionBlazor(true)** and load the scripts to the `<head>` element of the `~/wwwroot/index.html` page in Blazor WebAssembly app or `~/Pages/_Host.cshtml` page in Blazor Server app.

HTML

```
<head>
<script src="https://cdn.syncfusion.com/blazor/{{ site.blazorversion
}}/syncfusion-blazor.min.js"></script>
</head>
```

Adding component package to the application

Open `~/_Imports.razor` file and import the `Syncfusion.Blazor.Inputs` package.

ASPX-CS

```
@using Syncfusion.Blazor
@using Syncfusion.Blazor.Inputs
```

Adding Slider component to the application

Now, add the Syncfusion Blazor Slider component in any web page `razor` in the `Pages` folder. For example, the Slider component is added in the `~/Pages/Index.razor` page.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfSlider Value="30"></SfSlider>
```


Run the application

After successful compilation of your application, simply press **F5** to run the application.



See Also

[Slider Types](#)

[Slider Formatting](#)

[Orientation Slider](#)

[Ticks in Slider](#)

[Tooltip in Slider](#)

[Getting Started with Syncfusion Blazor for Client-Side in .NET Core CLI](#)

[Getting Started with Syncfusion Blazor for Client-Side in Visual Studio 2019](#)

[Getting Started with Syncfusion Blazor for Server-Side in .NET Core CLI](#)

Types in Blazor Range Slider Component

The types of Slider are as follows:

| **Types** | **Usage** |

| --- | --- |

| Default | Shows a default Slider to select a single value. |

| MinRange | Displays the shadow from the start value to the current selected value. |

| Range | Selects a range of values. It also displays the shadow in-between the selection range. |

Both the Default Slider and Min-Range Slider have same behavior that is used to select a single value.

In Min-Range Slider, a shadow is considered from the start value to current handle position. But the Range Slider

contains two handles that is used to select a range of values and a shadow is considered in between the two handles.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfSlider Value="30"></SfSlider>
<SfSlider Value="40" Type="SliderType.MinRange"></SfSlider>
<SfSlider Value=@RangeValue Type="SliderType.Range"></SfSlider>
@code{
public int[] RangeValue = { 30, 70 };
}
```

Default



MinRange



Range

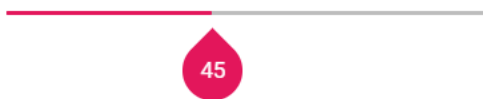


Tooltip in Blazor Range Slider Component

The Slider displays the tooltip to indicate the current value by clicking the Slider bar or drag the Slider handle. The Tooltip position can be customized by using the `Placement` property. Also decides the tooltip display mode on a page, i.e., on hovering, focusing, or clicking on the Slider handle and it always remains/displays on the page.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfSlider Min="0" Max="100" @bind-Value="@value">
  <SliderTooltip IsVisible="true" ShowOn="TooltipShowOn.Always"
    Placement="TooltipPlacement.After"></SliderTooltip>
</SfSlider>
@code {
  int value = 30;
}
```



Buttons

The Slider value can be changed by using the Increase and Decrease buttons. In Range Slider, by default the first handle value will be changed while clicking the button. Change the handle focus and press the button to change the last focused handle value.

After enabling the slider buttons if the 'Tab' key is pressed, the focus goes to the handle and not to the button.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfSlider @bind-Value="@value" ShowButtons="true" >
```

```
<SliderTooltip IsVisible="true" ShowOn="TooltipShowOn.Always"
Placement="TooltipPlacement.After"></SliderTooltip>
</SfSlider>
@code {
int value = 30;
}
```



Orientation in Blazor Range Slider Component

The Slider can be displayed, either in horizontal or vertical orientation. By default, the Slider renders in horizontal orientation.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfSlider ID="default" Value="40"
Orientation="SliderOrientation.Vertical"></SfSlider>
<style>
#default {
height: 300px;
}
</style>
```



Ticks in Blazor Range Slider Component

The Ticks in Slider supports to easily identify the current value/values of the Slider. It contains **SmallStep** and **LargeStep**. The value of the major ticks alone will be displayed in the slider. In order to enable/disable the small ticks, use the **ShowSmallTicks** property.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfSlider @bind-Value="@value">
```

```

<SliderTicks Placement="Placement.After" ShowSmallTicks="true"
LargeStep="20" SmallStep="10"></SliderTicks>
<SliderTooltip IsVisible="true" ShowOn="TooltipShowOn.Always"
Placement="TooltipPlacement.Before"></SliderTooltip>
</SfSlider>
@code {
int value = 30;
}

```



Step

When the Slider is moved, it increases/decreases the value based on the step value. By default, the value is increased/decreased by 1. Use the `Step` property to change the increment step value.

ASPX-CS

```

@using Syncfusion.Blazor.Inputs
<SfSlider Step="10" @bind-Value="@value">
<SliderTooltip IsVisible="true" ShowOn="TooltipShowOn.Always"
Placement="TooltipPlacement.Before"></SliderTooltip>
</SfSlider>
@code {
int value = 30;
}

```



Min and Max

Enables the minimum/starting and maximum/ending value of the Slider, by using the `Min` and `Max` property. By default, the minimum value is 1 and maximum value is 100. In the following sample the slider is rendered with the min value as 100 and max value as 1100.

ASPX-CS

```

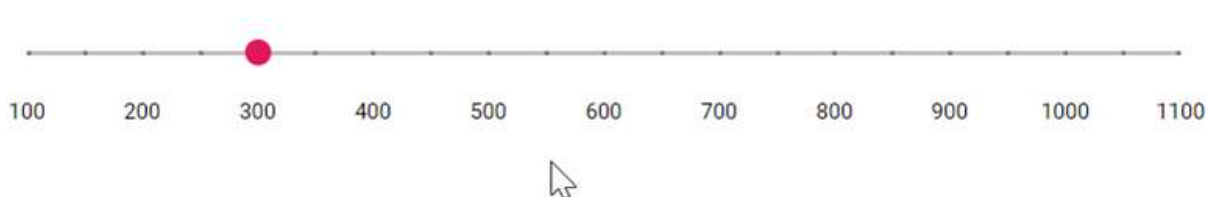
@using Syncfusion.Blazor.Inputs
<SfSlider @bind-Value="@value" Min="100" Max="1100">

```

```

<SliderTicks Placement="Placement.After" ShowSmallTicks="true"
LargeStep="100" SmallStep="50"></SliderTicks>
<SliderTooltip IsVisible="true" ShowOn="TooltipShowOn.Always"
Placement="TooltipPlacement.Before"></SliderTooltip>
</SfSlider>
@code {
int value = 300;
}

```



Formatting in Blazor Range Slider Component

The **Format** feature used to customize the units of Slider values to desired format. The formatted values will also be applied to the ARIA attributes of the slider. There are two ways of achieving formatting in slider.

Use the **Format** API of slider which utilizes our Internationalization to format values.

ASPX-CS

```

@using Syncfusion.Blazor.Inputs
<SfSlider @bind-Value="@CurrencyValue">
<SliderTooltip IsVisible="true" ShowOn="TooltipShowOn.Always" Format="C2"
Placement="TooltipPlacement.Before"></SliderTooltip>
<SliderTicks Placement="Placement.Before" Format="C2" ShowSmallTicks="true"
LargeStep="20" SmallStep="10"></SliderTicks>
</SfSlider>
@code {
int CurrencyValue = 30;
}

```



Using format API

Slider provides different predefined formatting styles like Numeric (N), Percentage (P), Currency (C) and # specifiers.

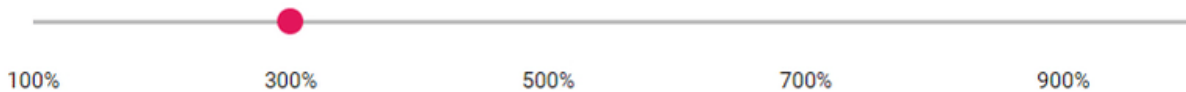
ASPX-CS

```

@using Syncfusion.Blazor.Inputs

```

```
<SfSlider Min="1" Max="10" @bind-Value="@PercentageValue">
  <SliderTicks Placement="Placement.After" Format="P0" ShowSmallTicks="true"
    LargeStep="2" SmallStep="1"></SliderTicks>
  <SliderTooltip IsVisible="true" ShowOn="TooltipShowOn.Always" Format="P0"
    Placement="TooltipPlacement.Before"></SliderTooltip>
</SfSlider>
@code {
  int PercentageValue = 3;
}
```



Limits in Blazor Range Slider Component

The slider limits restrict the slider thumb between a particular range. This is used if higher or lower value affects the process or product where the slider is being used.

The following are the six options in the slider's limits object. Each API in the limits object is optional.

- Enabled: Enables the limits in the Slider.
- MinStart: Sets the minimum limit for the first handle.
- MinEnd: Sets the maximum limit for the first handle.
- MaxStart: Sets the minimum limit for the second handle.
- MaxEnd: Sets the maximum limit for the second handle.
- StartHandleFixed: Locks the first handle.
- EndHandleFixed: Locks the second handle.

Default and MinRange slider limits

There is only one handle in the Default and MinRange Slider, so MinStart, MinEnd, and StartHandleFixed options can be used. When the limits are enabled in the Slider, the limited area becomes darkened. So you can differentiate the allowed and restricted area.

Refer to the following snippet to enable the limits in the Slider.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfSlider Value="30" Type="SliderType.MinRange">
  <SliderLimits MinStart="20" MinEnd="50" Enabled="true"></SliderLimits>
</SfSlider>
```



Range slider limits

In the range slider, both handles can be restricted and locked from the limit's object. In this sample, the first handle is limited between 10 and 40, and the second handle is limited between 60 and 90.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfSlider Value="@Value" Type="SliderType.Range">
  <SliderLimits Enabled="true" MinStart="10" MinEnd="40" MaxStart="60"
    MaxEnd="90"></SliderLimits>
</SfSlider>
@code{
public int[] Value = { 30, 70 };
}
```



Handle lock

The movement of slider handles can be locked by enabling the StartHandleFixed and EndHandleFixed properties in the limit's object.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfSlider Value="@Value" Type="SliderType.Range">
  <SliderLimits Enabled="true" StartHandleFixed="true" EndHandleFixed="true">
</SliderLimits>
</SfSlider>
@code{
public int[] Value = { 30, 70 };
}
```



CSS Structure in Blazor Range Slider 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 slider track

Use the following CSS to customize the slider track.

CSS

```
.e-control-wrapper.e-slider-container.e-horizontal .e-slider-track {  
  background: #007bff;  
  height: 3px;  
}
```

Customizing the slider handle

Use the following CSS to customize the slider handle properties.

CSS

```
.e-control-wrapper.e-slider-container .e-slider .e-handle {  
  background-color: #f9920b;  
  border-radius: 50%;  
  border: 0;  
}
```

Customizing the slider limits

Use the following CSS to customize the slider limits.

CSS

```
.e-control-wrapper.e-slider-container.e-horizontal .e-limits {  
  background-color: rgba(69, 100, 233, 0.46);  
}
```

Customizing the slider ticks

Use the following CSS to customize the slider ticks.

CSS

```
.e-scale .e-tick.e-custom::before {  
  content: '\e967';  
  position: absolute;  
}
```

Customizing the slider buttons

Use the following CSS to customize the slider buttons.

CSS

```
.e-control-wrapper.e-slider-container .e-slider-button {  
  background: #007bff;  
  height: 25px;  
  width: 25px;  
}
```


Localization in Blazor Range Slider Component

The **Localization** library allows to localize default text content of the Slider. The slider control has static text on some features (like increase and decrease button) that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the **Locale** value.

Blazor server side

Add **UseRequestLocalization** middle-ware in Configure method in **Startup.cs** file to get browser Culture Info.

CSHARP

```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Localization;
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            app.UseRequestLocalization();
            ....
            ....
        }
    }
}
```

The following sample, demonstrates how to enable **Localization** for Slider in server side Blazor samples. Here, Resource file is used to translate the static text.

The Resource file is an XML file which contains the strings(key and value pairs) that has to be translated into different languages. Refer Localization [link](#) to know more about how to configure and use localization in the Blazor framework.

Open the **Startup.cs** file and add the below configuration in the **ConfigureServices** function.

CSHARP

```
using Syncfusion.Blazor;
using System.Globalization;
using Microsoft.AspNetCore.Localization;
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void ConfigureServices(IServiceCollection services)
        {
            ....
            ....
            services.AddSyncfusionBlazor();
            services.AddLocalization(options => options.ResourcesPath = "Resources");
        }
    }
}
```

```

services.Configure<RequestLocalizationOptions>(options =>
{
    // define the list of cultures your app will support
    var supportedCultures = new List<CultureInfo>()
    {
        new CultureInfo("de")
    };
    // set the default culture
    options.DefaultRequestCulture = new RequestCulture("de");
    options.SupportedCultures = supportedCultures;
    options.SupportedUICultures = supportedCultures;
    options.RequestCultureProviders = new List<IRequestCultureProvider>() {
        new QueryStringRequestCultureProvider() // Here, You can also use other
        localization provider
    };
});
services.AddSingleton(typeof(ISyncfusionStringLocalizer),
    typeof(SampleLocalizer));
}
}
}

```

- Create `~/Shared/SyncfusionLocalizer.cs` file. Then, write a **class** by inheriting **ISyncfusionStringLocalizer** interface and override the Manager property to get the resource file details from the application end.

CSHARP

```

using Syncfusion.Blazor;
namespace BlazorApplication
{
    public class SampleLocalizer : ISyncfusionStringLocalizer
    {
        // To get the locale key from mapped resources file
        public string GetText(string key)
        {
            return this.ResourceManager.GetString(key);
        }
        // To access the resource file and get the exact value for locale key
        public System.Resources.ResourceManager ResourceManager
        {
            get
            {
                // Replace the ApplicationNamespace with your application name.
                return BlazorApplication.Resources.SyncfusionBlazorLocale.ResourceManager;
            }
        }
    }
}

```

- Add the resource files in the `~/Resources` folder. The locale resource files for different cultures are available in this [GitHub](#) repository. You can get any culture resource file from there and

utilize it in your application. Add **.resx** file to Resource folder and enter the key value (Locale Keywords) in the **Name** column and the translated string in the **Value** column as follows.

Name	Value (in Deutsch culture)
---	---
Slider_DecrementTitle	verringern
Slider_IncrementTitle	Erhöhen, ansteigen

- Finally, Specify the culture for Slider using **Locale** property.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfSlider Value="30" ShowButtons="true" Step="10" Locale="de"></SfSlider>
```

Blazor WebAssembly

The following sample, demonstrate how to enable **Localization** for Slider in client side Blazor samples.

- Open the **Program.cs** file and add the below configuration in the **Main** function as follows.

CSHARP

```
using Syncfusion.Blazor;
using System.Globalization;
namespace WebAssemblyLocale
{
    public class Program
    {
        public static async Task Main(string[] args)
        {
            ....
            ....
            builder.Services.AddSyncfusionBlazor();
            // Register the Syncfusion locale service to customize the SyncfusionBlazor
            // component locale culture
            builder.Services.AddSingleton(typeof(ISyncfusionStringLocalizer),
            typeof(SampleLocalizer));
            // Set the default culture of the application
            CultureInfo.DefaultThreadCurrentCulture = new CultureInfo("de");
            CultureInfo.DefaultThreadCurrentUICulture = new CultureInfo("de");
            await builder.Build().RunAsync();
        }
    }
}
```

- Then, create a **~/Shared/SampleLocalizer.cs** file and implement **ISyncfusionStringLocalizer** interface for the class and add Locale Keywords in the **.resx** file at Resource folder.

CSHARP

```

using Syncfusion.Blazor;
namespace WebAssemblyLocale
{
    public class SampleLocalizer : ISyncfusionStringLocalizer
    {
        // To get the locale key from mapped resources file
        public string GetText(string key)
        {
            return this.ResourceManager.GetString(key);
        }
        // To access the resource file and get the exact value for locale key
        public System.Resources.ResourceManager ResourceManager
        {
            get
            {
                // Replace the ApplicationNamespace with your application name.
                return WebAssemblyLocale.Resources.SyncfusionBlazorLocale.ResourceManager;
            }
        }
    }
}

```

For .NET 5.0 Blazor WebAssembly globalization, we should configure the `BlazorWebAssemblyLoadAllGlobalizationData` in the project file when the application uses large resources and dynamic culture changes.

XML

```

<PropertyGroup>
<BlazorWebAssemblyLoadAllGlobalizationData>true</BlazorWebAssemblyLoadAllGlobalizationData>
</PropertyGroup>

```

Refer [here](#) for more details.

- Now, Specify the culture for Slider using `Locale` property.

ASPX-CS

```

@using Syncfusion.Blazor.Inputs
<SfSlider Value="30" ShowButtons="true" Step="10" Locale="de"></SfSlider>

```



Accessibility in Blazor Range Slider Component

The Slider is characterized with complete ARIA Accessibility support that helps to access by on-screen readers and other assistive technology devices. This control is designed with the reference of guidelines document given in the [WAI ARAI Accessibility Practices](#).

The Slider control uses the Slider role and the following ARIA properties for its element based on the state.

| Properties | Functionalities |

| --- | --- |

| `aria-valuenow` | It indicates the current value of the slider. |

| `aria-valuetext` | It returns the current text of the slider. |

| `aria-valuemin` | It indicates the minimum value of the slider. |

| `aria-valuemax` | It indicates the maximum value of the slider. |

| `aria-orientation` | It indicates the Slider Orientation. |

| `aria-label` | Slider left and right button label text (increment and decrement). |

| `aria-labelledby` | It indicates the name of the Slider. |

Keyboard interaction

The Keyboard interaction of the Slider control is designed based on the [WAI-ARIA Practices](#) described for Slider. Users can use the following shortcut keys to interact with the Slider.

| Keyboard shortcuts | Actions |

| --- | --- |

| Right Arrow Up Arrow | Increase the Slider value. |

| Left Arrow Down Arrow | Decrease the Slider value. |

| Home | Moves to the start value (for Range Slider when the second thumb is focused and the Home key is pressed, it moves to the first thumb value). |

| End | Moves to the end value (for Range Slider when the first thumb is focused and the End key is pressed, it moves to the second thumb value). |

| Page Up | Increases the Slider by `LargeStep` value. |

| Page Down | Decreases the Slider by `LargeStep` value. |

How To

Customize the bar in Blazor Range Slider Component

Slider appearance can be customized through CSS. By overriding the slider CSS classes, you can customize the slider bar. The slider bar can be customized with different themes. By default, slider have class name `e-slider-track` for bar. The class can be overridden with our own color values like the following code snippet.

CSS

```
#gradient_slider.e-control.e-slider .e-range {
```

```

height: 6px;
top: calc(50% - 3px);
border-radius: 5px;
background: -webkit-linear-gradient(left, #e1451d 0, #fdff47 17%, #86f9fe 50%, #2900f8 65%, #6e00f8 74%, #e33df9 83%, #e14423 100%);
background: linear-gradient(left, #e1451d 0, #fdff47 17%, #86f9fe 50%, #2900f8 65%, #6e00f8 74%, #e33df9 83%, #e14423 100%);
background: -moz-linear-gradient(left, #e1451d 0, #fdff47 17%, #86f9fe 50%, #2900f8 65%, #6e00f8 74%, #e33df9 83%, #e14423 100%);
background:
url(data:image/svg+xml;base64,PD94bWwgdGVyc2lvbj0iMS4wIiA/Pgo8c3ZnIHhtbG5zPSJodHRwOi8vd3d3LnczLm9yZy8yMDAwL3N2ZyIgZD21kdGg9IjEwMCUiIGhlaWdodD0iMTAwJSIgdmlld0JveD0iMCAwIDEgMSIgcHJlc2VydmVBC3BlY3RSYXRpbz0ibm9uZSI+CiAgPGxpbmVhckdyYWRpZzW50IGlkPSJncmFkLXVjZ2ctZ2VuZXJhdGVkIiBncmFkaWVudFVuaXRzPSJlc2VyU3BhY2VPb1VzZSIgeDE9IjAlIiB5MT0iMCUiIHgyPSIyMDAlIiB5Mj0iMCUiPgogICAgPHN0b3Agb2Zmc2V0PSIwIiBzdG9wLWNvbG9yPSIjZTE0NTFkIiBzdG9wLW9wYWNpdHk9IjEiLz4KICAgIDxzdG9wIG9mZnNldD0iMTclIiBzdG9wLWNvbG9yPSIjZmRmZjQ3IiBzdG9wLW9wYWNpdHk9IjEiLz4KICAgIDxzdG9wIG9mZnNldD0iNTAlIiBzdG9wLWNvbG9yPSIjODZmOWZlIiBzdG9wLW9wYWNpdHk9IjEiLz4KICAgIDxzdG9wIG9mZnNldD0iNjUlIiBzdG9wLWNvbG9yPSIjMjkwMGY4IiBzdG9wLW9wYWNpdHk9IjEiLz4KICAgIDxzdG9wIG9mZnNldD0iNzQlIiBzdG9wLWNvbG9yPSIjNmUwMGY4IiBzdG9wLW9wYWNpdHk9IjEiLz4KICAgIDxzdG9wIG9mZnNldD0iODMlIiBzdG9wLWNvbG9yPSIjZTMzZGY5IiBzdG9wLW9wYWNpdHk9IjEiLz4KICAgIDxzdG9wIG9mZnNldD0iMTAwJSIgc3RvcC1jb2xvcj0iI2UxNDQyMyIgc3RvcC1vcGFjaXR5PSIiIi8+CiAgPC9saW5lYXJHcmFkaWVudD4KICA8cmVjdCB4PSIwIiB5PSIwIiB3aWR0aD0iMSIgaGVpZ2h0PSIiIiBmaWxsPSJlcwwoI2dyYWQtdWNNZy1nZW5lcmF0ZWQpIiAvPgo8L3N2Zz4=);
}

```

In this code snippet, it can be seen that the dynamic color of the slider is being changed.

ASPX-CS

```

<SfSlider @bind-Value="@Value" ID="dynamic_color_slider"
Type="SliderType.MinRange">
<SliderEvents TValue="int" ValueChange="@ (e => { OnChange(e.Value);
}) "></SliderEvents>
</SfSlider>

```

Color bar can be customized in quite a number of ways.

ASPX-CS

```

@using Syncfusion.Blazor.Inputs;
@using Syncfusion.Blazor.Buttons;
<div class="col-lg-12 control-section">
<div class="control-wrapper">
<div class="slider-content-wrapper">
<div class="slider_container">
<div class="slider-labeltext slider_userselect">Height</div>
<SfSlider Value="30" ID="height_slider">
</SfSlider>
</div>
<div class="slider_container">
<div class="slider-labeltext slider_userselect">Gradient color</div>
<SfSlider Value="50" ID="gradient_slider" Type="SliderType.MinRange">
</SfSlider>
</div>

```

```

<div class="slider_container">
<div class="slider-labeltext slider_userselect">Dynamic thumb and selection
bar color</div>
<SfSlider @bind-Value="@Value" ID="dynamic_color_slider"
Type="SliderType.MinRange" CssClass="@DynamicColor">
<SliderEvents TValue="int" ValueChange="@ (e => { OnChange(e.Value);
}) "></SliderEvents>
</SfSlider>
</div>
</div>
</div>
</div>
@code {
string DynamicColor = "e-slider-green";
int Value = 20;
void OnChange(int value)
{
if (value > 0 && value <= 25)
{
DynamicColor = "e-slider-green";
}
else if (value > 25 && value <= 50)
{
DynamicColor = "e-slider-royalblue";
}
else if (value > 50 && value <= 75)
{
DynamicColor = "e-slider-darkorange";
}
else if (value > 75 && value <= 100)
{
DynamicColor = "e-slider-red";
}
}
}
<style>
#dynamic_color_slider.e-slider-royalblue .e-range {
background-color: royalblue;
}
#dynamic_color_slider.e-slider-green .e-range {
background-color: green;
}
#dynamic_color_slider.e-slider-darkorange .e-range {
background-color: darkorange;
}
#dynamic_color_slider.e-slider-red .e-range {
background-color: red;
}
#dynamic_color_slider.e-slider-royalblue .e-handle {
background-color: royalblue;
}
#dynamic_color_slider.e-slider-green .e-handle {
background-color: green;
}
#dynamic_color_slider.e-slider-darkorange .e-handle {
background-color: darkorange;
}
}

```

```

#dynamic_color_slider.e-slider-red .e-handle {
background-color: red;
}
.slider-content-wrapper {
width: 40%;
margin: 0 auto;
min-width: 185px;
}
.slider-userselect {
-webkit-user-select: none;
/* Safari 3.1+ */
-moz-user-select: none;
/* Firefox 2+ */
-ms-user-select: none;
/* IE 10+ */
user-select: none;
/* Standard syntax */
}
.slider-labeltext {
text-align: -webkit-left;
font-weight: 500;
font-size: 13px;
padding-bottom: 10px;
}
.material .e-slider-container #height_slider.e-slider .e-handle,
.material.e-bigger .e-slider-container #height_slider.e-slider .e-handle,
.material .e-slider-container #gradient_slider.e-slider .e-handle,
.material.e-bigger .e-slider-container #gradient_slider.e-slider .e-handle {
height: 16px;
width: 16px;
}
.material .e-slider-container.e-horizontal #height_slider .e-handle,
.material.e-bigger .e-slider-container.e-horizontal #height_slider .e-
handle,
.material .e-slider-container.e-horizontal #gradient_slider .e-handle,
.material.e-bigger .e-slider-container.e-horizontal #gradient_slider .e-
handle {
margin-left: -8px;
top: calc(50% - 8px);
}
.e-bigger:not(.material) .e-slider-container.e-horizontal #height_slider.e-
slider .e-handle,
.e-bigger:not(.material) .e-slider-container.e-horizontal
#gradient_slider.e-slider .e-handle {
margin-left: -11px;
top: calc(50% - 11px);
}
.e-bigger:not(.material) .e-slider-container #height_slider.e-slider .e-
handle,
.e-bigger:not(.material) .e-slider-container #gradient_slider.e-slider .e-
handle {
height: 22px;
width: 22px;
}
.e-slider-container #height_slider.e-slider .e-handle,
.e-slider-container #gradient_slider.e-slider .e-handle {
height: 20px;

```



```

width: 20px;
}
.e-slider-container.e-horizontal #height_slider .e-handle,
.e-slider-container.e-horizontal #gradient_slider .e-handle {
margin-left: -10px;
top: calc(50% - 10px);
}
.slider_container {
margin-top: 40px;
}
.e-bigger .slider-content-wrapper {
width: 80%;
}
#height_slider .e-tab-handle::after {
background-color: #f9920b;
}
#height_slider.e-control.e-slider .e-slider-track {
height: 8px;
top: calc(50% - 4px);
border-radius: 0;
}
.highcontrast #height_slider.e-control.e-slider .e-slider-track {
height: 10px;
top: calc(50% - 5px);
border-radius: 0;
}
.fabric .slider_container .e-slider-hover .e-slider-track,
.fabric .slider_container .e-slider-container:active .e-slider-track,
.fabric .slider_container .e-slider-container .e-slider .e-tab-track {
background-color: #c8c8c8;
}
#gradient_slider.e-control.e-slider .e-range {
height: 6px;
top: calc(50% - 3px);
border-radius: 5px;
background: -webkit-linear-gradient(left, #e1451d 0, #fdff47 17%, #86f9fe
50%, #2900f8 65%, #6e00f8 74%, #e33df9 83%, #e14423 100%);
background: linear-gradient(left, #e1451d 0, #fdff47 17%, #86f9fe 50%,
#2900f8 65%, #6e00f8 74%, #e33df9 83%, #e14423 100%);
background: -moz-linear-gradient(left, #e1451d 0, #fdff47 17%, #86f9fe 50%,
#2900f8 65%, #6e00f8 74%, #e33df9 83%, #e14423 100%);
background:
url(data:image/svg+xml;base64,PD94bWwgdmVyc2lvdj0iMS4wIiA/Pgo8c3ZnIHhtbG5zPS
JodHRWoi8vd3d3LnczLm9yZy8yMDAwL3N2ZyIgd2lkdGg9IjEwMCUiIGhlaWdodD0iMTAwJSIgdml
ld0JveD0iMCAwIDEgMSIgcHJlc2VydmVbc3B1Y3RSYXRpbz0ibm9uZSI+CiAgPGxpbmVhckdyYW
RpZW50IGlkPSJncmFkLXVjZ2ctZ2VuZXJhdGVkIiBncmFkaWVudFVuaXRzPSJlc2VyU3BhY2VPbl
VzZSIgeDE9IjAlIiB5MT0iMCUiIHgyPSIxdMDAlIiB5Mj0iMCUiPgogICAgPHN0b3Agb2Zmc2V0PS
IwIiBzdG9wLWNvbG9yPSIjZTE0NTFkIiBzdG9wLW9wYWNpdHk9IjEiLz4KICAgIDxzdG9wIG9mZn
NldD0iMTclIiBzdG9wLWNvbG9yPSIjZmRmZjQ3IiBzdG9wLW9wYWNpdHk9IjEiLz4KICAgIDxzdG
9wIG9mZnNldD0iNTAlIiBzdG9wLWNvbG9yPSIjODZmOWZlIiBzdG9wLW9wYWNpdHk9IjEiLz4KIC
AgIDxzdG9wIG9mZnNldD0iNjUlIiBzdG9wLWNvbG9yPSIjMjkwMGY4IiBzdG9wLW9wYWNpdHk9Ij
EiLz4KICAgIDxzdG9wIG9mZnNldD0iNzQlIiBzdG9wLWNvbG9yPSIjNmUwMGY4IiBzdG9wLW9wYW
NpdHk9IjEiLz4KICAgIDxzdG9wIG9mZnNldD0iODMlIiBzdG9wLWNvbG9yPSIjZTMzZGY5IiBzdG
9wLW9wYWNpdHk9IjEiLz4KICAgIDxzdG9wIG9mZnNldD0iMTAwJSIgc3RvcC1jb2xvcj0iI2UxND
QyMyIgc3RvcC1vcGFjaXR5PSIxi8+CiAgPC9saW5lYXJhcmFkaWVudD4KICA8cmVjdCB4PSIwIi
B5PSIwIiB3aWR0aD0iMSIgaGVpZ2h0PSIxiBmaWxsPSJlcmlwI2dyYWQtdWNnZylnZW5lcmF0ZW
QpIiAvPgo8L3N2Zz4=);

```

```

}
.fabric .slider_container .e-slider-hover .e-slider-track,
.fabric .slider_container .e-slider-container:active .e-slider-track,
.fabric .slider_container .e-slider-container .e-slider .e-tab-track {
background-color: #c8c8c8;
}
#gradient_slider.e-control.e-slider .e-slider-track {
height: 8px;
top: calc(50% - 4px);
border-radius: 5px;
}
</style>

```

Height



Gradient color



Dynamic thumb and selection bar color



Customize the limits in Blazor Range Slider Component

Slider appearance can be customized via CSS. By overriding the slider CSS classes, the slider limit bar can be customized. Here, the limit bar is customized with different background color. By default, the slider has class `e-limits` for limits bar. You can override the class with our own color values as given in the following code snippet.

CSS

```

.e-control-wrapper.e-slider-container.e-horizontal .e-limits {
background-color: rgba(69, 100, 233, 0.46);
}

```

And on implementing the above code snippet in the below slider control's Blazor code, the overview would be like the one found below.

ASPX-CS

```

@using Syncfusion.Blazor.Inputs;
@using Syncfusion.Blazor.Buttons;
<div class="col-lg-8 control-section sb-property-border">
<div class="content-wrapper">
<div class='sliderwrap'>
<label class="userselect">MinRange Slider With Limits</label>
<SfSlider @bind-Value="@Default" Min="0" Max="100" Type=SliderType.MinRange>
<SliderTicks Placement="@Placement.Before" LargeStep="20" SmallStep="5"
ShowSmallTicks="true"></SliderTicks>
<SliderTooltip IsVisible="true" Placement="@TooltipPlacement.Before"
ShowOn="@TooltipShowOn.Focus"></SliderTooltip>
<SliderLimits Enabled="true"
MinStart="10"
MinEnd="40"
StartHandleFixed="false"></SliderLimits>
</SfSlider>
</div>
<div class='sliderwrap'>
<label class="userselect">Range Slider With Limits</label>
<SfSlider @bind-Value="@Range" Min="0" Max="100" Type=SliderType.Range>
<SliderTicks Placement="@Placement.Before" LargeStep="20" SmallStep="5"
ShowSmallTicks="true"></SliderTicks>
<SliderTooltip IsVisible="true" Placement="@TooltipPlacement.Before"
ShowOn="@TooltipShowOn.Focus"></SliderTooltip>
<SliderLimits Enabled="true"
MinStart="10"
MinEnd="40"
MaxStart="50"
MaxEnd="90"></SliderLimits>
</SfSlider>
</div>
</div>
</div>
</div>
@code{
int Default = 25;
int[] Range = { 25, 75 };
}
<style>
.content-wrapper {
width: 52%;
margin: 0 auto;
min-width: 185px;
}
.sliderwrap {
margin-top: 45px;
}
.e-bigger .content-wrapper {
width: 80%;
}
.e-control-wrapper.e-slider-container.e-horizontal .e-limits {
background-color: rgba(69, 100, 233, 0.46);
}
.sliderwrap label {
padding-bottom: 50px;
font-size: 13px;
font-weight: 500;
margin-top: 15px;

```

```

}
.userselect {
-webkit-user-select: none;
/* Safari 3.1+ */
-moz-user-select: none;
/* Firefox 2+ */
-ms-user-select: none;
/* IE 10+ */
user-select: none;
/* Standard syntax */
}
.property-custom td {
padding: 5px;
}
</style>

```

MinRange Slider With Limits



Range Slider With Limits



Customize the thumb in Blazor Range Slider Component

Slider appearance can be customized through CSS. By overriding the slider CSS classes, you can customize the thumb. By default, slider has unique class `e-handle` for slider thumb. You can override the following class as per your requirement. Here, in the sample, the slider thumb has been customized to square, circle, oval shapes, and background image has also been customized.

ASPX-CS

```

@using Syncfusion.Blazor.Inputs;
@using Syncfusion.Blazor.Buttons;

```

```
<div class="col-lg-12 control-section">
<div class="control-wrapper">
<div class="slider-content-wrapper">
<div class="slider_container">
<div class="labelText slider-userselect">Square</div>
<SfsSlider Value="30" ID="square_slider">
</SfsSlider>
</div>
<div class="slider_container">
<div class="labelText slider-userselect">Circle</div>
<SfsSlider Value="30" ID="circle_slider">
</SfsSlider>
</div>
<div class="slider_container">
<div class="labelText slider-userselect">Oval</div>
<SfsSlider Value="30" ID="oval_slider">
</SfsSlider>
</div>
<div class="slider_container">
<div class="labelText slider-userselect">Custom image</div>
<SfsSlider Value="30" ID="image_slider">
<SliderTicks Placement="@Placement.After" LargeStep="10" SmallStep="5"
ShowSmallTicks="true"></SliderTicks>
</SfsSlider>
</div>
</div>
</div>
</div>
</div>
<style>
.slider-content-wrapper {
width: 40%;
margin: 0 auto;
min-width: 185px;
}
.slider-userselect {
-webkit-user-select: none;
/* Safari 3.1+ */
-moz-user-select: none;
/* Firefox 2+ */
-ms-user-select: none;
/* IE 10+ */
user-select: none;
/* Standard syntax */
}
.labelText {
text-align: -webkit-left;
font-weight: 500;
font-size: 13px;
padding-bottom: 10px;
}
.slider_container {
margin-top: 40px;
}
.e-bigger .content-wrapper {
width: 80%;
}
#square_slider.e-control.e-slider .e-handle {
```

```

border-radius: 0%;
background-color: #f9920b;
border: 0;
}
#circle_slider.e-control.e-slider .e-handle {
background-color: #f9920b;
border-radius: 50%;
border: 0;
}
.material.e-bigger .e-slider-container.e-horizontal #image_slider.e-slider
.e-handle,
.material .e-slider-container.e-bigger.e-horizontal #image_slider.e-slider
.e-handle {
top: calc(50% - 7px);
}
.material.e-bigger .e-slider-container.e-horizontal #image_slider.e-slider
.e-handle.e-handle-active,
.material .e-slider-container.e-bigger.e-horizontal #image_slider.e-slider
.e-handle.e-handle-active {
top: calc(50% - 6px);
transform: scale(1.3) !important;
}
.e-bigger .e-slider-container.e-horizontal #image_slider.e-slider .e-handle,
.e-slider-container.e-bigger.e-horizontal #image_slider.e-slider .e-handle {
top: calc(50% - 9px);
}
#image_slider.e-control.e-slider .e-handle {
height: 25px;
width: 24px;
background-size: 24px;
}
.material #image_slider.e-control.e-slider .e-handle {
height: 20px;
width: 20px;
background-size: 20px;
}
.material #image_slider.e-control.e-slider .e-handle {
background-image: url(../../slider/images/thumb-mat.png);
background-repeat: no-repeat;
background-color: transparent;
border: 0;
}
#image_slider.e-control.e-slider .e-handle {
background-image:
url(https://ej2.syncfusion.com/demos/src/slider/images/thumb.png);
background-repeat: no-repeat;
background-color: transparent;
border: 0;
}
#square_slider .e-tab-handle::after,
#circle_slider .e-tab-handle::after {
background-color: #f9920b;
}
#image_slider .e-tab-handle::after {
background-color: transparent;
}
#oval_slider.e-control.e-slider .e-handle {

```

```
height: 25px;
width: 8px;
top: 3px;
border-radius: 15px;
background-color: #f9920b;
}
</style>
```

Square



Circle



Oval



Custom image



Customize the tick label in Blazor Range Slider Component

Slider can be customized via CSS. By overriding the slider CSS classes, you can customize the ticks. The ticks in slider allows to easily identify the current value/values of the slider. It contains `smallStep` and `largeStep`. By default, slider has class `e-tick` for slider ticks. You can override the class as per your requirement. Refer to the following code snippet to render ticks.

CSS

```
.e-scale .e-tick.e-custom::before {
```

```
content: '\e967';
position: absolute;
}
```

Here, the color for rendered ticks has been applied through `nth-child(childnumber)`. *The color is applied to the value of the childnumber* in the slider.

CSS

```
#ticks_slider .e-scale :nth-child(1)::before {
color: red;
}
```

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<div id="app">
<div id="container">
<div class="col-lg-12 control-section">
<div class="slider-content-wrapper">
<div class="slider_container" id="slider_wrapper">
<div class="slider_labelText userselect">Dynamic ticks color</div>
<!-- Ticks slider element -->
<SfSlider ID="ticks_slider" Enabled="true" Min="10" Max="60" Step="10"
Type=SliderType.Default @bind-Value="@value">
<SliderTicks Placement="Placement.Before"
ShowSmallTicks="true"
LargeStep="10"
SmallStep="10"></SliderTicks>
<SliderEvents TicksRendering="@TicksRendering" TValue="int"></SliderEvents>
</SfSlider>
</div>
</div>
</div>
</div>
</div>
@code {
int value = 50;
public void TicksRendering(SliderTickEventArgs args)
{
args.HtmlAttributes["class"] = args.HtmlAttributes["class"] + " e-custom";
}
}
<style>
#app {
height: 40px;
position: absolute;
width: 98%;
}
li {
float: left;
}
.slider-content-wrapper {
width: 40%;
margin: 0 auto;
```



```

min-width: 185px;
}
.userselect {
-webkit-user-select: none;
/* Safari 3.1+ */
-moz-user-select: none;
/* Firefox 2+ */
-ms-user-select: none;
/* IE 10+ */
user-select: none;
/* Standard syntax */
}
.slider_labelText {
text-align: left;
font-weight: 500;
font-size: 13px;
padding-bottom: 40px;
}
.slider_container {
margin-top: 40px;
}
.e-bigger .slider-content-wrapper {
width: 80%;
}
#ticks_slider .e-range {
z-index: unset;
}
#ticks_slider .e-scale .e-tick {
background-image: none;
visibility: visible;
font-family: "e-customized-icons";
}
@@font-face {
font-family: "e-customized-icons";
src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj8iS4cAAAEoAAAAVmNtYXDS5tJrAAABjAAAAEBnbHlmdMA
KbQAAAdQAAAOwaGVhZBNseyYAAADQAAAAANmhoZWEHogNjAAAArAAAACRobXR4C9AAAAAAAYAAAAA
MbG9jYQCaAdgAAAHMAAAACG1heHABEAeUAAABCAAAACBuYW1lc0cOBgAABYQAAAIlcG9zdNSlKbQ
AAAEsAAAArWABAAADUv9qAFoEAAAA//UD8wABAAAAAAAAAAAAAAAAAAAAAwABAAAAAQAAtxzLE18
PPPUACwPoAAAAANgtmycAAAAA2C2bJwAAAAAD8wPzAAAACAACAAAAAAAAAAAAEAAAADASIAAwAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPwAZAABQAAAanoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA6QLpZwNS/2oAWgPzAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAAAAAAAgAAAAAAMAAUAMAAQAAABQABAAsAAAAABgAEAAEAauK6Wf//wAA6QLpZ//AAAAAA
BAAYABgAAAAEAAgAAAAAamgHYAAIAAAAAA+oD6gAzAIcAAAEzHxghNT8WEx8THQEPEisBLxI9AT8
SAGAQECQmKCgPKScTEhIREA8ODQwKCgQHBQQBAfwqAQMFbGcKCgwNDg8QERISEycpKSgoJiQgDQw
MDAwXFhUUEhEPDQsJCAIDAQEBAQMCCakLDQ8REhQVFhcMDAwMDQ0MDAwMFxYVFBIRDw0LCQgCAwE
BAQEDAggJCw0PERIUFRYXDAwMDAGFAQMEBwkKDQ4ICakKCgoLCwwMDAcNDg8Og3sPDw4NDgwMDAs
LCgoKCQgIDg0KCQcEAWJnAQEBAGMHCgsNDxESExUWFfwMDQwNDA0MDAwXFhUTEExAPDQwJBwMCAgE
BAGIDBwkMDQ8QEXMVFhcMDAwNDA0MDQwMFxYVExIRDw0LCgcDAgEBAAAAAwAAAAAD8wPzAF8AwAE
hAAABDxmFfz8XLxcPAjcfFA8XLxc/Fx8CJw8UHxc/Fy8XDwIBqRQUFBISERAQDg0NCwoJBwCFBAI
BAQIEBQcHCQoLDQ00EBAREhIUFBQVFhYWFhYWFRUTFBISERAQDg0NCwoJBwCFBAIBAQIEBQcHCQo
LDQ00EBAREhIUExUVFhYWFhYWtg4NGxkZGBYWFWMSEA8OCwsIBwUDAQEDBQcICwsODxASExUWFhg
ZGRsbHB0dHh4dHRwbGxkZGBYWFWMSEA8NDAsIBwUDAQEDBQcICwsODxASExUVFhgZGRsbHB0dHh4
dHd0QDx4eHBSaGRcWFRIREA0MCQgGAwEBAwYICQwNEBESFRYXGRobHB4eHyEgIiIiIiAhHx4eHBS
aGRcWFRIREA0MCQgGAwEBAwYICQwNEBESFRYXGRobHB4eHyEgIiIiIiEDPAYICQoLDQ00EBAREhI
TFBUVFRYXfYhYWFRQUFBISERAQDg0MDA0JBwCFBAIBAQIEBQcHCQoMDA00EBAREhIUFBQVFhYWFxY
VFRUUEXISERAQDg0NCwoJCAYFBAIBAQIEZAQECgwODxASExUVFhgYGHsbHB0dHh4dHRwbGxkZGBY

```

```

WFBQSEA8NDAoJBWUDAQEDBQcICwsODxASExUWFhgZGRsbHB0dHh4dHRwbGxoYGBcVFRMSEA8OCws
IBWUDAQEDBTYFBQwNEBESFRYXGRobHB0fHyEgIiIiIiEgHx4eHBsaGRcWFBMRDw4MCQgGAwEBAwY
ICQwODxETFBYXGRobHB4eHyEgIiIiIiAhHx4eHBsaGRcWFRIRDw4MCQgGAwEBAwYAAAAAAAAASAN4
AAQAAAAAAAAABAAAAAQAQAAAAAAQAHAAEAQAQAAAAAAAgAHAAgAAQAAAAAAAwAHAA8AAQAAAAAABAA
HABYAAQAAAAAABQALAB0AAQAAAAAABgAHACgAAQAAAAAACgAsAC8AAQAAAAAACwASAFsAAwABBAk
AAAACAG0AAwABBAkAAQA0AG8AAwABBAkAAgA0AH0AAwABBAkAAwA0AIsAAwABBAkABAA0AJkAAwA
BBakABQAWAKcAAwABBAkABgA0AL0AAwABBAkACgBYAMsAAwABBAkACwAkASMgZS1pY29uc1JlZ3V
sYXJlLWljb25zZS1pY29uc1ZlcnNpb24gMS4wZS1pY29uc0ZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN
5bmNmdXNpb24gTWV0cm8gU3RlZGlvd3d3LnN5bmNmdXNpb24uY29tACAAZQAtAGkAYwBvAG4AcwB
SAGUAZwB1AGwAYQBByAGUALQBpAGMABwBuAHMAZQAtAGkAYwBvAG4AcwBWAGUAacgBzAGkAbwBuACA
AMQAuADAAZQAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwB1AG4AZQByAGEAdABlAGQAIAB1AHMAaQB
uAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQB1AHQAcgBvACAAUwB0AHUAZABpAG8AdwB3AHc
ALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAgAAAAAAAAAKAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAADAQIBAwEEAAh0ZW1wLWN1cxJGQl9DaGVja2JveF9zZWxlY3QAAAA=)
format("truetype");
font-weight: normal;
font-style: normal;
}
#ticks_slider .e-scale {
z-index: 0;
}
#ticks_slider .e-scale .e-custom::before {
content: "\e967";
position: absolute;
}
#ticks_slider .e-scale :nth-child(1)::before {
color: red;
}
#ticks_slider .e-scale :nth-child(2)::before {
color: blue;
}
#ticks_slider .e-scale :nth-child(3)::before {
color: green;
}
#ticks_slider .e-scale :nth-child(4)::before {
color: blueviolet;
}
#ticks_slider .e-scale :nth-child(5)::before {
color: orange;
}
#ticks_slider .e-scale :nth-child(6)::before {
color: pink;
}
#ticks_slider .e-scale .e-custom::before {
font-size: 10px;
}
#ticks_slider .e-scale .e-custom::before {
top: calc(50% + 1px);
left: calc(50% - 5px);
}
#slider_wrapper #ticks_slider .e-scale :nth-child(1)::before {
top: calc(50% + 1px);
left: calc(0% - 5px);
}
#slider_wrapper #ticks_slider .e-scale :nth-child(6)::before {
top: calc(50% + 1px);
left: calc(100% - 6px);
}

```

```
}
</style>
```



Numeric Range Slider in Blazor Range Slider Component

The numeric values can be formatted into different decimal digits or fixed number of whole numbers or to represent the units. The Numeric processing is demonstrated below.

There are numeric range sliders, which can be formatted in any way of your choice. In the examples found below, the first one demonstrates the visualization of ticks in km.

ASPX-CS

```
<SliderTicks ShowSmallTicks="true" Placement="Placement.After"
LargeStep="20" SmallStep="10" Format="##.##km"> </SliderTicks>
<SliderTooltip IsVisible="true" Placement="TooltipPlacement.Before"
Format="##.##km"></SliderTooltip>
```

The second example showcases the use of decimal point in the Slider's ticks and tooltip placement.

ASPX-CS

```
<SliderTicks ShowSmallTicks="true" Placement="Placement.After"
LargeStep="20" SmallStep="10" Format="##.##00"> </SliderTicks>
<SliderTooltip IsVisible="true" Placement="TooltipPlacement.Before"
Format="##.##00"></SliderTooltip>
```

And in the third example, the formatting involves the placement of zeros before the required values.

ASPX-CS

```
<SliderTicks ShowSmallTicks="true" Placement="Placement.After"
LargeStep="20" SmallStep="10" Format="0000#"> </SliderTicks>
<SliderTooltip IsVisible="true" Placement="TooltipPlacement.Before"
Format="0000#"></SliderTooltip>
```

The complete code for the above Numeric Range Slider can be found below.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs;
<div class="control-section">
<div class="content-wrapper">
<div class="sliderwrap">
<label class="labeltext userselect">Default Slider</label>
<SfSlider @bind-Value="@Value1">
```

```

<SliderTicks ShowSmallTicks="true" Placement="Placement.After"
LargeStep="20" SmallStep="10" Format="##.##km"> </SliderTicks>
<SliderTooltip IsVisible="true" Placement="TooltipPlacement.Before"
Format="##.##km"></SliderTooltip>
</SfSlider>
</div>
<div class="sliderwrap">
<label class="labeltext userselect">MinRange Slider</label>
<SfSlider @bind-Value="@Value2" Type=SliderType.MinRange>
<SliderTicks ShowSmallTicks="true" Placement="Placement.After"
LargeStep="20" SmallStep="10" Format="##.##00"> </SliderTicks>
<SliderTooltip IsVisible="true" Placement="TooltipPlacement.Before"
Format="##.##00"></SliderTooltip>
</SfSlider>
</div>
<div class="sliderwrap">
<label class="labeltext userselect">Range slider</label>
<SfSlider @bind-Value="@Value" Type=SliderType.Range>
<SliderTicks ShowSmallTicks="true" Placement="Placement.After"
LargeStep="20" SmallStep="10" Format="0000#"> </SliderTicks>
<SliderTooltip IsVisible="true" Placement="TooltipPlacement.Before"
Format="0000#"></SliderTooltip>
</SfSlider>
</div>
</div>
</div>
@code{
int Value1 = 30;
int Value2 = 30;
public int[] Value = { 20, 50 };
}
<style>
.content-wrapper {
width: 40%;
margin: 0 auto;
min-width: 185px;
}
.sliderwrap {
margin-top: 40px;
}
.e-bigger .content-wrapper {
width: 80%;
}
.sliderwrap label {
padding-bottom: 26px;
font-size: 13px;
font-weight: 500;
margin-top: 15px;
}
.userselect {
-webkit-user-select: none; /* Safari 3.1+ */
-moz-user-select: none; /* Firefox 2+ */
-ms-user-select: none; /* IE 10+ */
user-select: none; /* Standard syntax */
}
</style>

```



Date Range Slider in Blazor Range Slider Component

The date formatting can be achieved using `TicksRendering` and `TooltipChange` events. The process of date formatting is explained in the below sample.

ASPX-CS

```
@using System.Globalization;
@using Syncfusion.Blazor.Inputs
<SfSlider TValue="int" Min="MinValue()" Max="@MaxValue()" @bind-
Value="@value">
  <SliderEvents TicksRendering="@TicksRendering" TValue="int"
OnTooltipChange="@TooltipChange"></SliderEvents>
  <SliderTicks LargeStep="1" ShowSmallTicks="true"
Placement="Placement.Before"> </SliderTicks>
  <SliderTooltip Placement="TooltipPlacement.After"
IsVisible="true"></SliderTooltip>
</SfSlider>
@code{
int value = 15;
string MonthName = new DateTime(DateTime.Now.Year, DateTime.Now.Month,
13).ToString("MMM", CultureInfo.InvariantCulture);
```

```

public double MinValue()
{
    DateTime datetime = new DateTime(DateTime.Now.Year, DateTime.Now.Month, 13);
    return datetime.Day;
}
public double MaxValue()
{
    DateTime datetime = new DateTime(DateTime.Now.Year, DateTime.Now.Month, 21);
    return datetime.Day;
}
public void TicksRendering(SliderTickEventArgs args)
{
    args.Text = MonthName + " " + args.Value + ", " + DateTime.Now.Year;
}
public void TooltipChange(SliderTooltipEventArgs<int> args)
{
    args.Text = MonthName + " " + args.Value + ", " + DateTime.Now.Year;
}
}

```



Time Range Slider in Blazor Range Slider Component

The time formatting can be achieved same as the date formatting using **TicksRendering** and **TooltipChange** events.

ASPX-CS

```

@using Syncfusion.Blazor.Inputs
<SfSlider TValue="int[]" Min="MinValue()" Max="@MaxValue()"
Type="SliderType.Range" @bind-Value="@SliderValues">
<SliderEvents TValue="int[]" OnTooltipChange="@TooltipChange"
TicksRendering="@TicksRendering"></SliderEvents>
<SliderTicks Placement="Placement.Before" LargeStep="7200000"
SmallStep="3600000" ShowSmallTicks="true"></SliderTicks>
<SliderTooltip Placement="TooltipPlacement.After"
IsVisible="true"></SliderTooltip>
</SfSlider>
@code{
int[] SliderValues = new int[] { 43200000, 54000000 };
public double MinValue()
{
    DateTime datetime = new DateTime(2013, 6, 13, 11, 0, 0);
    return datetime.TimeOfDay.TotalMilliseconds;
}
public double MaxValue()
{
    DateTime datetime = new DateTime(2013, 6, 13, 23, 0, 0);
    return datetime.TimeOfDay.TotalMilliseconds;
}
}

```

```

public void TicksRendering(SliderTickEventArgs args)
{
    double time = args.Value / 3600000;
    args.Text = time > 11 ? time + ".00 PM" : time + ".00 AM";
}
public void TooltipChange(SliderTooltipEventArgs<int[]> args)
{
    double FirstValue = args.Value[0] / 3600000;
    double SecondValue = args.Value[1] / 3600000;
    if (FirstValue <= 11 && SecondValue < 11)
    {
        args.Text = FirstValue + ".00 AM -" + SecondValue + ".00 AM";
    }
    else if (FirstValue <= 11 && SecondValue > 11)
    {
        args.Text = FirstValue + ".00 AM -" + SecondValue + ".00 PM";
    }
    else if (FirstValue > 11 && SecondValue > 11)
    {
        args.Text = FirstValue + ".00 PM -" + SecondValue + ".00 PM";
    }
}

```



Validation of Slider in Blazor Range Slider Component

The Slider component can be validated using our FormValidator. The following steps walk-through slider validation. Render slider component inside a form, by giving the required styles.

In Blazor Slider component, the DataAnnotation Validator is used here.

C#

```

public class Annotation
{
    [Required, Range(0, 40, ErrorMessage = "You must select a value less than or equal to forty.")]
    public int Value { get; set; }
}

```

ASPX-CS

```

@using System.ComponentModel.DataAnnotations;
@using Syncfusion.Blazor.Inputs;
<div class="form-title">
    <span>Range</span>
</div>
<EditForm Model="@annotation">
    <DataAnnotationsValidator />

```

```

<div class="form-group">
<div class="e-float-input">
<SfSlider @bind-Value ="annotation.Value"></SfSlider>
<ValidationMessage For="@(() => annotation.Value)" />
</div>
</div>
</EditForm>
<div class="form-title">
<span>Value</span>
</div>
<EditForm Model="@annotation">
<DataAnnotationsValidator />
<div class="form-group">
<div class="e-float-input">
<SfSlider @bind-Value="annotation.rangeval"></SfSlider>
<ValidationMessage For="@(() => annotation.rangeval)" />
</div>
</div>
</EditForm>
@code {
private Annotation annotation = new Annotation();
public class Annotation
{
[Required, Range(0, 40, ErrorMessage = "You must select a value less than or
equal to forty.")]
public int Value { get; set; }
[Required, RegularExpression("40", ErrorMessage = "You must select a value
equal to forty.")]
public int rangeval { get; set; }
}
}
<style>
.e-error,
.e-float-text {
font-weight: 500;
}
table,
td,
th {
padding: 5px;
}
.form-horizontal {
margin-left: 0;
margin-right: 0;
}
form {
border: 1px solid #ccc;
box-shadow: 0 1px 3px 0 rgba(0, 0, 0, 0.36);
border-radius: 5px;
background: #f9f9f9;
padding: 23px;
padding-bottom: 20px;
margin: auto;
max-width: 650px;
}
.form-title {
width: 100%;

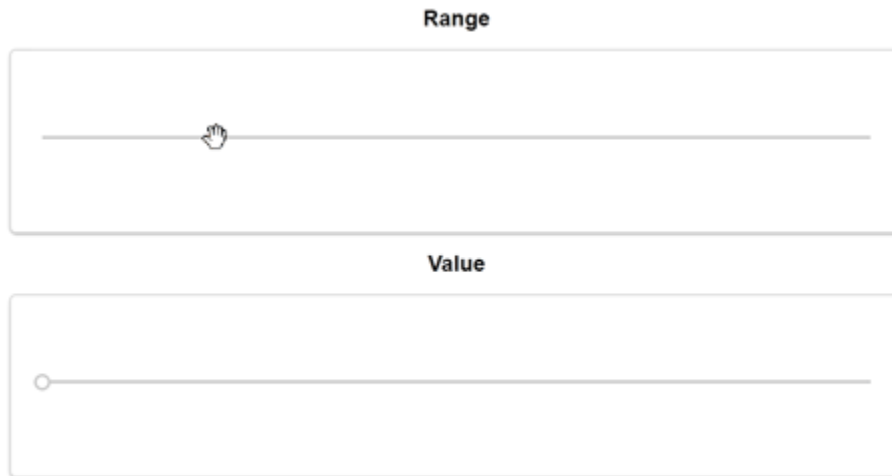
```



```

text-align: center;
padding: 10px;
font-size: 16px;
font-weight: 600;
color: black;
}
</style>

```



RichTextEditor

<!-- markdownlint-disable MD024 -->

Getting Started with Blazor RichTextEditor Component

This section briefly explains how to include a Rich Text Editor component in your Blazor Server-side application. You can refer to our Getting Started with [Syncfusion Blazor for Server-Side in Visual Studio page](#) for the introduction and configuring the common specifications.

To get start quickly with Blazor Rich Text Editor components, you can check on this video:

{% youtube

"youtube:https://www.youtube.com/watch?v=NvauE3z7W0k"%}

You can also explore our [Blazor Rich Text Editor](#) example to knows how to render and configure the rich text editor tools.

Importing Syncfusion Blazor component in the application

- Install **Syncfusion.Blazor.RichTextEditor** NuGet package to the application by using the **NuGet Package Manager**.
- You can add the client-side resources through [CDN](#) or from [NuGet](#) package in the **HEAD** element of the `~/Pages/_Host.cshtml` page.

ASPX-CS

```

<head>
<environment include="Development">

```

```

....
....
<link href="_content/Syncfusion.Blazor/styles/fabric.css" rel="stylesheet"
/>
<!--CDN-->
@*<link href="https://cdn.syncfusion.com/blazor/18.4.42/styles/fabric.css"
rel="stylesheet" />*@
</environment>
</head>

```

For Internet Explorer 11 kindly refer the polyfills. Refer the [documentation](#) for more information.

ASPX-CS

```

<head>
<environment include="Development">
<link href="_content/Syncfusion.Blazor/styles/fabric.css" rel="stylesheet"
/>
<script
src="https://github.com/Daddoon/Blazor.Polyfill/releases/download/3.0.1/blaz
or.polyfill.min.js"></script>
</environment>
</head>

```

Adding component package to the application

Open ~/_Imports.razor file and import the **Syncfusion.Blazor.RichTextEditor** package.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
```

Add SyncfusionBlazor service in Startup.cs

Open the **Startup.cs** file and add services required by Syncfusion components using **services.AddSyncfusionBlazor()** method. Add this method in the **ConfigureServices** function as follows.

CSHARP

```

using Syncfusion.Blazor;
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void ConfigureServices(IServiceCollection services)
        {
            ....
            ....
            services.AddSyncfusionBlazor();
        }
    }
}

```

Add Rich Text Editor component

To initialize the Rich Text Editor component, add the below code to your **Index.razor** view page which is present under **~/Pages** folder.

The following code explains how to initialize a simple Rich Text Editor in Razor page.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<p>Rich Text Editor allows to insert images from online source as well as local computer where you want to insert the image in your content.</p>
<p><b>Get started Quick Toolbar to click on the image</b></p>
<p>It is possible to add custom style on the selected image inside the Rich Text Editor through quick toolbar.</p>
</SfRichTextEditor>
```

Run the application

After successful compilation of your application, run the application.



Configure the Toolbar

Configure the toolbar with the tools using **Items** field of the **RichTextEditorToolbarSettings** property as your application requires.

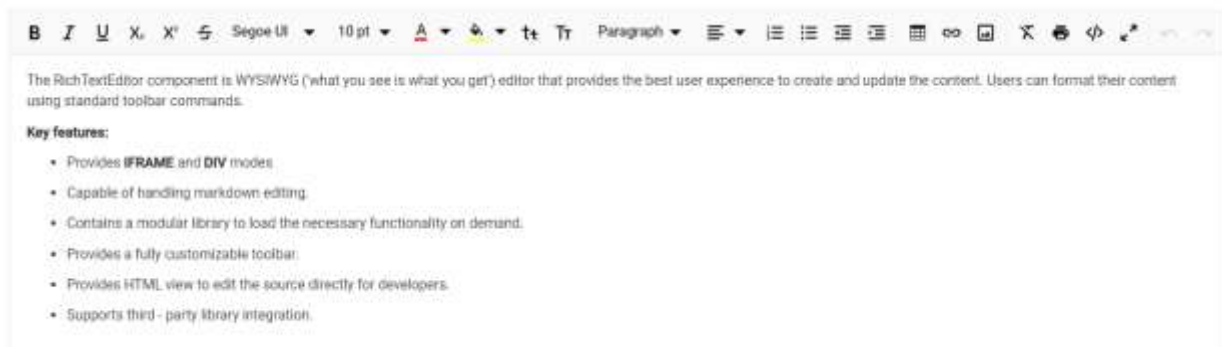
ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorToolbarSettings Items="@Tools" />
<p>The Rich Text Editor component is WYSIWYG ('what you see is what you get') editor that provides the best user experience to create and update the content. Users can format their content using standard toolbar commands.</p>
<p><b>Key features:</b></p>
<ul>
<li><p>Provides <b>IFRAME</b> and <b>DIV</b> modes </p></li>
<li><p>Capable of handling markdown editing.</p></li>
<li><p>Contains a modular library to load the necessary functionality on demand.</p></li>
<li><p>Provides a fully customizable toolbar.</p></li>
<li><p>Provides HTML view to edit the source directly for developers.</p></li>
<li><p>Supports third - party library integration.</p></li>
<li><p>Allows preview of modified content before saving it.</p></li>
</ul>
</SfRichTextEditor>
@code {
private List<ToolbarItemModel> Tools = new List<ToolbarItemModel>()
{
```

```

new ToolbarItemModel() { Command = ToolbarCommand.Bold },
new ToolbarItemModel() { Command = ToolbarCommand.Italic },
new ToolbarItemModel() { Command = ToolbarCommand.Underline },
new ToolbarItemModel() { Command = ToolbarCommand.StrikeThrough },
new ToolbarItemModel() { Command = ToolbarCommand.FontName },
new ToolbarItemModel() { Command = ToolbarCommand.FontSize },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.FontColor },
new ToolbarItemModel() { Command = ToolbarCommand.BackgroundColor },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.Formats },
new ToolbarItemModel() { Command = ToolbarCommand.Alignments },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.LowerCase },
new ToolbarItemModel() { Command = ToolbarCommand.UpperCase },
new ToolbarItemModel() { Command = ToolbarCommand.SuperScript },
new ToolbarItemModel() { Command = ToolbarCommand.SubScript },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.OrderedList },
new ToolbarItemModel() { Command = ToolbarCommand.UnorderedList },
new ToolbarItemModel() { Command = ToolbarCommand.Outdent },
new ToolbarItemModel() { Command = ToolbarCommand.Indent },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.CreateLink },
new ToolbarItemModel() { Command = ToolbarCommand.Image },
new ToolbarItemModel() { Command = ToolbarCommand.CreateTable },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.ClearFormat },
new ToolbarItemModel() { Command = ToolbarCommand.Print },
new ToolbarItemModel() { Command = ToolbarCommand.SourceCode },
new ToolbarItemModel() { Command = ToolbarCommand.FullScreen },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.Undo },
new ToolbarItemModel() { Command = ToolbarCommand.Redo }
};
}

```



Insert images and links

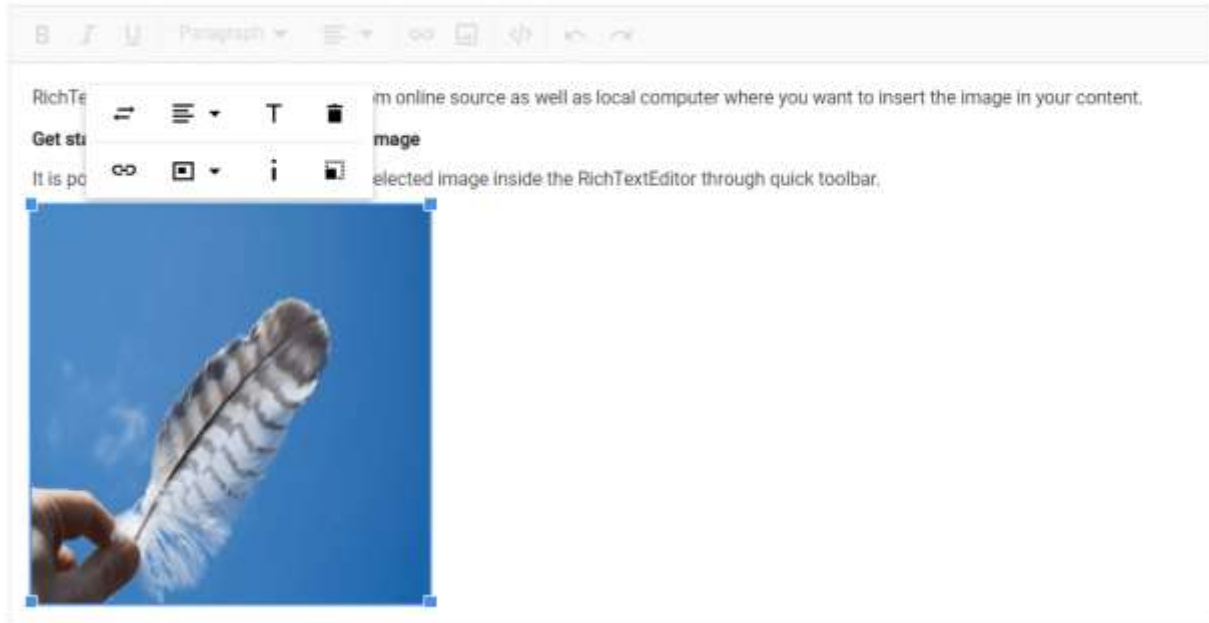
The **Image** module inserts an image into Rich Text Editor's content area, and the **Link** module links external resources such as website URLs to the selected text in the Rich Text Editor's content respectively.

Specifies the items to be rendered in quick toolbar based on the target elements such as image, link and table element. The quick toolbar opens to customize the element by clicking the target element.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorToolbarSettings Items="@Tools" />
<RichTextEditorQuickToolbarSettings Image="@Image" Link="@Link" />
<p>Rich Text Editor allows to insert images from online source as well as
local computer where you want to insert the image in your content.</p>
<p><b>Get started Quick Toolbar to click on the image</b></p>
<p>It is possible to add custom style on the selected image inside the Rich
Text Editor through quick toolbar.</p>
<img alt='Logo' style='width: 300px; height: 300px; transform:
rotate(0deg);'
src='https://blazor.syncfusion.com/demos/images/RichTextEditor/RTEImage-
Feather.png' />
</SfRichTextEditor>
@code {
private List<ToolbarItemModel> Tools = new List<ToolbarItemModel>()
{
new ToolbarItemModel() { Command = ToolbarCommand.Bold },
new ToolbarItemModel() { Command = ToolbarCommand.Italic },
new ToolbarItemModel() { Command = ToolbarCommand.Underline },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.Formats },
new ToolbarItemModel() { Command = ToolbarCommand.Alignments },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.CreateLink },
new ToolbarItemModel() { Command = ToolbarCommand.Image },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.SourceCode },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.Undo },
new ToolbarItemModel() { Command = ToolbarCommand.Redo }
};
private List<ImageToolbarItemModel> Image = new
List<ImageToolbarItemModel>()
{
new ImageToolbarItemModel() { Command = ImageToolbarCommand.Replace },
new ImageToolbarItemModel() { Command = ImageToolbarCommand.Align },
new ImageToolbarItemModel() { Command = ImageToolbarCommand.Caption },
new ImageToolbarItemModel() { Command = ImageToolbarCommand.Remove },
new ImageToolbarItemModel() { Command = ImageToolbarCommand.OpenImageLink },
new ImageToolbarItemModel() { Command =
ImageToolbarCommand.HorizontalSeparator },
new ImageToolbarItemModel() { Command = ImageToolbarCommand.EditImageLink },
new ImageToolbarItemModel() { Command = ImageToolbarCommand.RemoveImageLink
},
new ImageToolbarItemModel() { Command = ImageToolbarCommand.Display },
new ImageToolbarItemModel() { Command = ImageToolbarCommand.AltText },
new ImageToolbarItemModel() { Command = ImageToolbarCommand.Dimension }
};
private List<LinkToolbarItemModel> Link = new List<LinkToolbarItemModel>()
{
new LinkToolbarItemModel() { Command = LinkToolbarCommand.Open },
```

```
new LinkToolbarItemModel() { Command = LinkToolbarCommand.Edit },
new LinkToolbarItemModel() { Command = LinkToolbarCommand.UnLink }
};
}
```



Retrieve the formatted content

To retrieve the editor contents, use the `Value` property of Rich Text Editor. To fetch the Rich Text Editor's text content, use `GetText` method of Rich Text Editor.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.RichTextEditor
@using Syncfusion.Blazor.Popups
<SfButton @onclick="@GetValue">Get Value</SfButton>
<SfButton @onclick="@GetText">Get Text</SfButton>
<br />
<SfDialog @ref="DialogObj" @bind-Visible="@Visibility" Content="@Content"
Header="@Header" Target="#target" Height="200px"
Width="400px" ShowCloseIcon="true">
<DialogButtons>
<DialogButton Content="Ok" IsPrimary="true" OnClick="@DlgButtonClick" />
</DialogButtons>
</SfDialog>
<SfRichTextEditor @ref="RteObj" Value="@RteValue" />
@code {
SfRichTextEditor RteObj;
SfDialog DialogObj;
private string Content;
private bool Visibility = false;
private string Header = "RichTextEditor's Value";
private string RteValue = @"<p>Rich Text Editor allows to insert images from
online source as well as local computer where you want to insert the image
```

```

in your content.</p><p><b>Get started Quick Toolbar to click on the
image</b></p><p>It is possible to add custom style on the selected image
inside the Rich Text Editor through quick toolbar.</p><img alt='Logo'
style='width: 300px; height: 300px; transform: rotate(0deg);'
src='https://blazor.syncfusion.com/demos/images/RichTextEditor/RTEImage-
Feather.png' />";
private async Task GetValue()
{
    this.Content = this.RteValue;
    await this.DialogObj.ShowAsync();
}
private async Task GetText()
{
    this.Content = await this.RteObj.GetTextAsync();
    this.DialogObj.ShowAsync();
}
private async Task DlgButtonClick(object arg)
{
    await this.DialogObj.HideAsync();
}
}

```

See Also

- [Getting Started with Syncfusion Blazor for client-side in .NET Core CLI](#)
- [Getting Started with Syncfusion Blazor for server-side in Visual Studio](#)
- [Getting Started with Syncfusion Blazor for server-side in .NET Core CLI](#)

Data Binding in Blazor RichTextEditor Component

This section explains how to bind the **Value** to the Rich Text Editor component that can be achieved in the following ways:

- One-way data binding
- Two-way data binding
- Dynamic value binding

One-way data binding

You can bind the value to the Rich Text Editor by using the **Value** property directly as string or from code-behind as the following example.

ASPX-CS

```

@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor Value="@Value" />
@code {
    private string Value { get; set; } = "<p>Syncfusion RichTextEditor</p>";
}

```

Two-way data binding

The two-way data binding can be achieved by using the **@bind-Value** attribute from code-behind in Rich Text Editor.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor @bind-Value="@Value" />
<br />
<textarea rows="5" cols="60" @bind="@Value" />
@code {
private string Value { get; set; } = "<p>Syncfusion RichTextEditor</p>";
}
```

Dynamic value binding

You can update the value dynamically by using the `Value` property.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.RichTextEditor
<SfButton @onclick="@Update">Update Value</SfButton>
<SfRichTextEditor Value="@Value" />
@code {
private string Value { get; set; } = "<p>Syncfusion RichTextEditor</p>";
private void Update()
{
this.Value = "<p>Dynamic Value</p>";
}
}
```

You can refer to our [Blazor Rich Text Editor](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor Rich Text Editor](#) example to know how to render and configure the rich text editor tools.

Editor Modes in Blazor RichTextEditor Component

The [Blazor Rich Text Editor](#) component is used to create and edit the content and return valid HTML markup or markdown (MD) of the content. It supports the following two editing formations:

- HTML Editor
- Markdown Editor

HTML editor

Rich Text Editor is a [WYSIWYG Editor](#) component for formatting the word content as HTML. The HTML editing mode is the default mode in Rich Text Editor to format the content through the available toolbar items to return the valid HTML markup. Set the `EditorMode` property to `HTML`.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor EditorMode="EditorMode.HTML" >
<p>The Rich Text Editor component is WYSIWYG ('what you see is what you get') editor that provides the best user experience to create and update the content. Users can format their content using standard toolbar commands.</p>
<p><b>Key features:</b></p>
<ul>
<li><p> Provides <b>IFRAME</b> and <b>DIV</b> modes </p></li>
```



```

<li><p> Capable of handling markdown editing.</p></li>
<li><p> Contains a modular library to load the necessary functionality on
demand.</p></li>
<li><p> Provides a fully customizable toolbar.</p></li>
<li><p> Provides HTML view to edit the source directly for
developers.</p></li>
<li><p> Supports third - party library integration.</p></li>
</ul>
</SfRichTextEditor>

```



Markdown editor

Set the `EditorMode` property to `Markdown` to create or edit the content and apply formatting to view markdown formatted content. The third-party library such as `Marked` or any other library is used to convert markdown into HTML content.

- The Supported Tags are `h6,h5,h4,h3,h2,h1,blockquote,pre,p,OL,UL`.
- The Supported Selection Tags are `Bold, Italic, StrikeThrough, InlineCode, SubScript, SuperScript, UpperCase` and `LowerCase`.

ASPX-CS

```

@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor EditorMode="EditorMode.Markdown" >
***Overview***The Rich Text Editor component is WYSIWYG ('what you see is
what you get') editor used to create and edit the content and return valid
HTML markup or markdown (MD) of the content. The editor provides a standard
toolbar to format content using its commands. Modular library features to
load the necessary functionality on demand. The toolbar contains commands to
align the text, insert link, insert image, insert list, undo/redo operation,
HTML view, and more.
***Key features***
*Mode*: Provides IFRAME and DIV mode.
*Module*: Modular library to load the necessary functionality on demand.
*Toolbar*: Provide a fully customizable toolbar.
*Editing*: HTML view to edit the source directly for developers.
*Third-party Integration*: Supports to integrate third-party library.
*Preview*: Preview the modified content before saving it.
*Tools*: Handling images, hyperlinks, video, uploads and more.
*Undo and Redo*: Undo/redo manager.
*Lists*:Creates bulleted and numbered list
</SfRichTextEditor>

```



For further details on Markdown editing, refer to the [Markdown](#) section.

You can also explore our [Blazor Rich Text Editor](#) example to know how to render and configure the rich text editor tools.

See Also

- [How to render the iframe](#)

Toolbar in Blazor RichTextEditor Component

The Rich Text Editor toolbar contains a collection of tools such as bold, Italic, and text alignment buttons that are used to format the content. However, in most integrations, you can customize the toolbar configurations easily to suit your needs. The Rich Text Editor allows to configure different types of toolbar using `RichTextEditorToolbarSettings - Type` property. The types of toolbar are:

1. Expand
2. MultiRow

Expand Toolbar

The default mode of `RichTextEditorToolbarSettings - Type` as `Expand` to hide the overflowing items in the next row. By clicking the expand arrow, view the overflowing toolbar items.

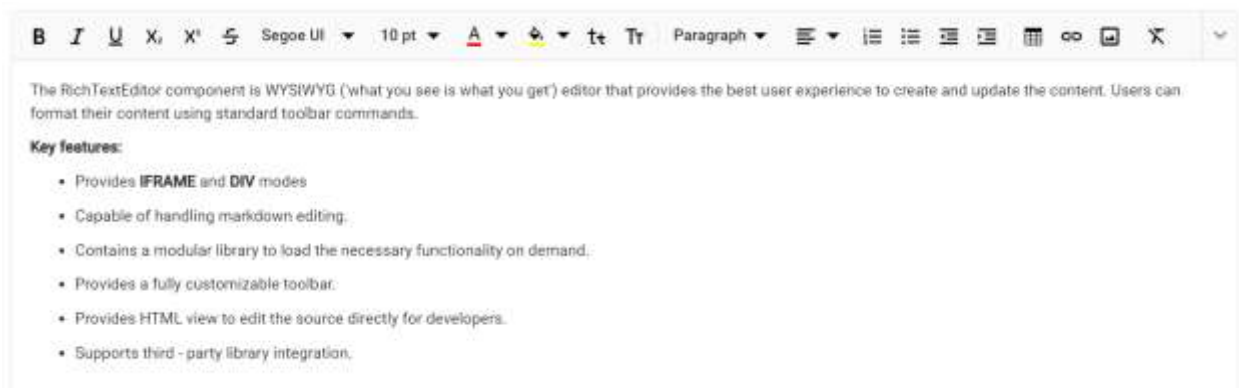
ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorToolbarSettings Items="@Tools" Type="ToolbarType.Expand" />
<p>The Rich Text Editor component is WYSIWYG ('what you see is what you get') editor that provides the best user experience to create and update the content. Users can format their content using standard toolbar commands.</p>
<p><b>Key features:</b></p>
<ul>
<li><p>Provides <b>IFRAME</b> and <b>DIV</b> modes </p></li>
<li><p>Capable of handling markdown editing.</p></li>
<li><p>Contains a modular library to load the necessary functionality on demand.</p></li>
<li><p>Provides a fully customizable toolbar.</p></li>
<li><p>Provides HTML view to edit the source directly for developers.</p></li>
<li><p>Supports third - party library integration.</p></li>
</ul>
</SfRichTextEditor>
@code{
```

```

private List<ToolbarItemModel> Tools = new List<ToolbarItemModel>()
{
    new ToolbarItemModel() { Command = ToolbarCommand.Bold },
    new ToolbarItemModel() { Command = ToolbarCommand.Italic },
    new ToolbarItemModel() { Command = ToolbarCommand.Underline },
    new ToolbarItemModel() { Command = ToolbarCommand.StrikeThrough },
    new ToolbarItemModel() { Command = ToolbarCommand.Separator },
    new ToolbarItemModel() { Command = ToolbarCommand.FontName },
    new ToolbarItemModel() { Command = ToolbarCommand.FontSize },
    new ToolbarItemModel() { Command = ToolbarCommand.Separator },
    new ToolbarItemModel() { Command = ToolbarCommand.FontColor },
    new ToolbarItemModel() { Command = ToolbarCommand.BackgroundColor },
    new ToolbarItemModel() { Command = ToolbarCommand.Separator },
    new ToolbarItemModel() { Command = ToolbarCommand.LowerCase },
    new ToolbarItemModel() { Command = ToolbarCommand.UpperCase },
    new ToolbarItemModel() { Command = ToolbarCommand.Separator },
    new ToolbarItemModel() { Command = ToolbarCommand.SuperScript },
    new ToolbarItemModel() { Command = ToolbarCommand.SubScript },
    new ToolbarItemModel() { Command = ToolbarCommand.Separator },
    new ToolbarItemModel() { Command = ToolbarCommand.Formats },
    new ToolbarItemModel() { Command = ToolbarCommand.Alignments },
    new ToolbarItemModel() { Command = ToolbarCommand.Separator },
    new ToolbarItemModel() { Command = ToolbarCommand.OrderedList },
    new ToolbarItemModel() { Command = ToolbarCommand.UnorderedList },
    new ToolbarItemModel() { Command = ToolbarCommand.Separator },
    new ToolbarItemModel() { Command = ToolbarCommand.Outdent },
    new ToolbarItemModel() { Command = ToolbarCommand.Indent },
    new ToolbarItemModel() { Command = ToolbarCommand.Separator },
    new ToolbarItemModel() { Command = ToolbarCommand.CreateLink },
    new ToolbarItemModel() { Command = ToolbarCommand.Image },
    new ToolbarItemModel() { Command = ToolbarCommand.CreateTable },
    new ToolbarItemModel() { Command = ToolbarCommand.Separator },
    new ToolbarItemModel() { Command = ToolbarCommand.ClearFormat },
    new ToolbarItemModel() { Command = ToolbarCommand.Separator },
    new ToolbarItemModel() { Command = ToolbarCommand.Print },
    new ToolbarItemModel() { Command = ToolbarCommand.SourceCode },
    new ToolbarItemModel() { Command = ToolbarCommand.FullScreen },
    new ToolbarItemModel() { Command = ToolbarCommand.Separator },
    new ToolbarItemModel() { Command = ToolbarCommand.Undo },
    new ToolbarItemModel() { Command = ToolbarCommand.Redo }
};
}

```



Multi-row Toolbar

Set the `RichTextEditorToolbarSettings` - `Type` as `MultiRow` to display the toolbar items in a row-wise format. All toolbar items are visible always.

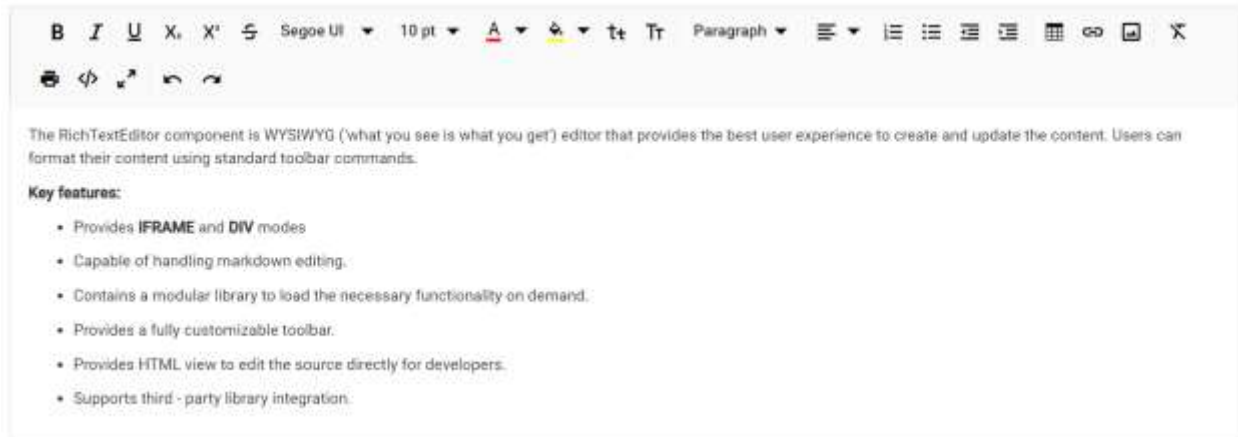
ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorToolbarSettings Items="@Tools" Type="ToolbarType.MultiRow" />
<p>The Rich Text Editor component is WYSIWYG ('what you see is what you
get') editor that provides the best user experience to create and update the
content. Users can format their content using standard toolbar commands.</p>
<p><b>Key features:</b></p>
<ul>
<li><p>Provides <b>IFRAME</b> and <b>DIV</b> modes </p></li>
<li><p>Capable of handling markdown editing.</p></li>
<li><p>Contains a modular library to load the necessary functionality on
demand.</p></li>
<li><p>Provides a fully customizable toolbar.</p></li>
<li><p>Provides HTML view to edit the source directly for
developers.</p></li>
<li><p>Supports third - party library integration.</p></li>
</ul>
</SfRichTextEditor>
@code {
private List<ToolbarItemModel> Tools = new List<ToolbarItemModel>()
{
new ToolbarItemModel() { Command = ToolbarCommand.Bold },
new ToolbarItemModel() { Command = ToolbarCommand.Italic },
new ToolbarItemModel() { Command = ToolbarCommand.Underline },
new ToolbarItemModel() { Command = ToolbarCommand.StrikeThrough },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.FontName },
new ToolbarItemModel() { Command = ToolbarCommand.FontSize },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.FontColor },
new ToolbarItemModel() { Command = ToolbarCommand.BackgroundColor },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.LowerCase },
new ToolbarItemModel() { Command = ToolbarCommand.UpperCase },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.SuperScript },
new ToolbarItemModel() { Command = ToolbarCommand.SubScript },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.Formats },
new ToolbarItemModel() { Command = ToolbarCommand.Alignments },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.OrderedList },
new ToolbarItemModel() { Command = ToolbarCommand.UnorderedList },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.Outdent },
new ToolbarItemModel() { Command = ToolbarCommand.Indent },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.CreateLink },
new ToolbarItemModel() { Command = ToolbarCommand.Image },
new ToolbarItemModel() { Command = ToolbarCommand.CreateTable },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
}
```

```

new ToolbarItemModel() { Command = ToolbarCommand.ClearFormat },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.Print },
new ToolbarItemModel() { Command = ToolbarCommand.SourceCode },
new ToolbarItemModel() { Command = ToolbarCommand.FullScreen },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.Undo },
new ToolbarItemModel() { Command = ToolbarCommand.Redo }
};
}

```



Floating Toolbar

By default, toolbar is float at the top of the Rich Text Editor on scrolling. It can be customized by specifying the offset of the floating toolbar from documents top position using `FloatingToolbarOffset`.

Enable or disable the floating toolbar using `EnableFloating` of the `RichTextEditorToolbarSettings` property.

ASPX-CS

```

@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor Height="800px">
<RichTextEditorToolbarSettings EnableFloating="false" Items="@Tools" />
<p>The Rich Text Editor component is WYSIWYG ('what you see is what you get') editor that provides the best user experience to create and update the content. Users can format their content using standard toolbar commands.</p>
<p><b>Key features:</b></p>
<ul>
<li><p>Provides <b>IFRAME</b> and <b>DIV</b> modes </p></li>
<li><p>Capable of handling markdown editing.</p></li>
<li><p>Contains a modular library to load the necessary functionality on demand.</p></li>
<li><p>Provides a fully customizable toolbar.</p></li>
<li><p>Provides HTML view to edit the source directly for developers.</p></li>
<li><p>Supports third - party library integration.</p></li>
</ul>
</SfRichTextEditor>
@code{
private List<ToolbarItemModel> Tools = new List<ToolbarItemModel>()

```

```

{
new ToolbarItemModel() { Command = ToolbarCommand.Bold },
new ToolbarItemModel() { Command = ToolbarCommand.Italic },
new ToolbarItemModel() { Command = ToolbarCommand.Underline },
new ToolbarItemModel() { Command = ToolbarCommand.StrikeThrough },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.FontName },
new ToolbarItemModel() { Command = ToolbarCommand.FontSize },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.FontColor },
new ToolbarItemModel() { Command = ToolbarCommand.BackgroundColor },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.LowerCase },
new ToolbarItemModel() { Command = ToolbarCommand.UpperCase },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.SuperScript },
new ToolbarItemModel() { Command = ToolbarCommand.SubScript },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.Formats },
new ToolbarItemModel() { Command = ToolbarCommand.Alignments },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.OrderedList },
new ToolbarItemModel() { Command = ToolbarCommand.UnorderedList },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.Outdent },
new ToolbarItemModel() { Command = ToolbarCommand.Indent },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.CreateLink },
new ToolbarItemModel() { Command = ToolbarCommand.Image },
new ToolbarItemModel() { Command = ToolbarCommand.CreateTable },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.ClearFormat },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.Print },
new ToolbarItemModel() { Command = ToolbarCommand.SourceCode },
new ToolbarItemModel() { Command = ToolbarCommand.FullScreen },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.Undo },
new ToolbarItemModel() { Command = ToolbarCommand.Redo }
};
}

```

Toolbar items

The following table lists the tools available in the toolbar.

Name	Summary	Initialization
Undo	Allows to undo the actions.	<pre> public List<ToolbarItemModel> Items = new List<ToolbarItemModel>() {
 new ToolbarItemModel() { Command = ToolbarCommand.Undo }
 }; </pre>
Redo	Allows to redo the actions.	<pre> public List<ToolbarItemModel> Items = new List<ToolbarItemModel>() {
 new ToolbarItemModel() { Command = ToolbarCommand.Redo }
 }; </pre>

| Alignment | Aligns the content with left, center, and right margin. | public List<ToolbarItemModel> Items = new List<ToolbarItemModel>() {
#160;#160;#160;#160;new ToolbarItemModel() { Command = ToolbarCommand.Alignments }
 }; |

| OrderedList | Creates a new list item(numbered). | public List<ToolbarItemModel> Items = new List<ToolbarItemModel>() {
#160;#160;#160;#160;new ToolbarItemModel() { Command = ToolbarCommand.OrderedList }
 }; |

| UnorderedList | Creates a new list item(bulleted). | public List<ToolbarItemModel> Items = new List<ToolbarItemModel>() {
#160;#160;#160;#160;new ToolbarItemModel() { Command = ToolbarCommand.UnorderedList }
 }; |

| Indent | Allows to increase the indent level of the content. | public List<ToolbarItemModel> Items = new List<ToolbarItemModel>() {
#160;#160;#160;#160;new ToolbarItemModel() { Command = ToolbarCommand.Indent }
 }; |

| Outdent | Allows to decrease the indent level of the content. | public List<ToolbarItemModel> Items = new List<ToolbarItemModel>() {
#160;#160;#160;#160;new ToolbarItemModel() { Command = ToolbarCommand.Outdent }
 }; |

| Hyperlink | Creates a hyperlink to a text or image to a specific location in the content. | public List<ToolbarItemModel> Items = new List<ToolbarItemModel>() {
#160;#160;#160;#160;new ToolbarItemModel() { Command = ToolbarCommand.CreateLink }
 }; |

| Images | Inserts an image from an online source or local computer. | public List<ToolbarItemModel> Items = new List<ToolbarItemModel>() {
#160;#160;#160;#160;new ToolbarItemModel() { Command = ToolbarCommand.Image }
 }; |

| LowerCase | Changes the case of selected text to lower in the content. | public List<ToolbarItemModel> Items = new List<ToolbarItemModel>() {
#160;#160;#160;#160;new ToolbarItemModel() { Command = ToolbarCommand.LowerCase }
 }; |

| UpperCase | Changes the case of selected text to upper in the content. | public List<ToolbarItemModel> Items = new List<ToolbarItemModel>() {
#160;#160;#160;#160;new ToolbarItemModel() { Command = ToolbarCommand.UpperCase }
 }; |

| SubScript | Makes the selected text as subscript (lower). | public List<ToolbarItemModel> Items = new List<ToolbarItemModel>() {
#160;#160;#160;#160;new ToolbarItemModel() { Command = ToolbarCommand.SubScript }
 }; |

| SuperScript | Makes the selected text as superscript (higher). | public List<ToolbarItemModel> Items = new List<ToolbarItemModel>() {
#160;#160;#160;#160;new ToolbarItemModel() { Command = ToolbarCommand.SuperScript }
 }; |

| Print | Allows to print the editor content. | public List<ToolbarItemModel> Items = new List<ToolbarItemModel>() {
#160;#160;#160;#160;new ToolbarItemModel() { Command = ToolbarCommand.Print }
 }; |

| FontName | Defines the fonts that appear under the Font Family DropDownList from the Rich Text Editor's toolbar. | public List<ToolbarItemModel> Items = new List<ToolbarItemModel>() {
#160;#160;#160;#160;new ToolbarItemModel() { Command = ToolbarCommand.FontName }
 }; |

| FontSize | Defines the font sizes that appear under the Font Size DropDownList from the Rich Text Editor's toolbar. | public List<ToolbarItemModel> Items = new List<ToolbarItemModel>()

```
{<br>&#160;&#160;&#160;&#160;new ToolbarItemModel() { Command = ToolbarCommand.FontSize }
<br>};|
```

| FontColor | Specifies an array of colors can be used in the colors popup for font color. | public List<ToolbarItemModel> Items = new List<ToolbarItemModel>() {
 new ToolbarItemModel() { Command = ToolbarCommand.FontColor }
 }; |

| BackgroundColor | Specifies an array of colors can be used in the colors popup for background color. | public List<ToolbarItemModel> Items = new List<ToolbarItemModel>() {
 new ToolbarItemModel() { Command = ToolbarCommand.BackgroundColor }
 }; |

| Format | An Object with the options that will appear in the Paragraph Format dropdown from the toolbar. | public List<ToolbarItemModel> Items = new List<ToolbarItemModel>() {
 new ToolbarItemModel() { Command = ToolbarCommand.Formats }
 }; |

| StrikeThrough | Applies double line strike through formatting for the selected text. | public List<ToolbarItemModel> Items = new List<ToolbarItemModel>() {
 new ToolbarItemModel() { Command = ToolbarCommand.StrikeThrough }
 }; |

| ClearFormat | The clear format tool is useful to remove all formatting styles (such as bold, italic, underline, color, superscript, subscript, and more) from currently selected text. As a result, all the text formatting will be cleared and return to its default formatting styles. | public List<ToolbarItemModel> Items = new List<ToolbarItemModel>() {
 new ToolbarItemModel() { Command = ToolbarCommand.ClearFormat }
 }; |

| FullScreen | Stretches the editor to the maximum width and height of the browser window. | public List<ToolbarItemModel> Items = new List<ToolbarItemModel>() {
 new ToolbarItemModel() { Command = ToolbarCommand.FullScreen }
 }; |

| SourceCode | Rich Text Editor includes the ability for users to directly edit HTML code via "Source View". If you made any modification in Source view directly, synchronize with Design view. | public List<ToolbarItemModel> Items = new List<ToolbarItemModel>() {
 new ToolbarItemModel() { Command = ToolbarCommand.SourceCode }
 }; |

| NumberFormatList | Allows to create list items with various list style types(numbered). | public List<ToolbarItemModel> Items = new List<ToolbarItemModel>() {
 new ToolbarItemModel() { Command = ToolbarCommand.NumberFormatList }
 }; |

| BulletFormatList | Allows to create list items with various list style types(bulleted). | public List<ToolbarItemModel> Items = new List<ToolbarItemModel>() {
 new ToolbarItemModel() { Command = ToolbarCommand.BulletFormatList }
 }; |

By default, tools will be arranged in the following order.

```
new List<ToolbarItemModel>()<br>{<br>&#160;&#160;&#160;&#160;new ToolbarItemModel() {
Command = ToolbarCommand.Bold },<br>&#160;&#160;&#160;&#160;new ToolbarItemModel() {
Command = ToolbarCommand.Italic },<br>&#160;&#160;&#160;&#160;new ToolbarItemModel() {
Command = ToolbarCommand.Underline },<br>&#160;&#160;&#160;&#160;new ToolbarItemModel() {
Command = ToolbarCommand.Separator },<br>&#160;&#160;&#160;&#160;new ToolbarItemModel() {
Command = ToolbarCommand.Formats },<br>&#160;&#160;&#160;&#160;new ToolbarItemModel() {
Command = ToolbarCommand.Alignments },<br>&#160;&#160;&#160;&#160;new ToolbarItemModel()
{ Command = ToolbarCommand.OrderedList },<br>&#160;&#160;&#160;&#160;new
```



```

ToolbarItemModel() { Command = ToolbarCommand.UnorderedList
},<br>#160;#160;#160;#160;new ToolbarItemModel() { Command = ToolbarCommand.Separator
},<br>#160;#160;#160;#160;new ToolbarItemModel() { Command = ToolbarCommand.CreateLink
},<br>#160;#160;#160;#160;new ToolbarItemModel() { Command = ToolbarCommand.Image
},<br>#160;#160;#160;#160;new ToolbarItemModel() { Command = ToolbarCommand.Separator
},<br>#160;#160;#160;#160;new ToolbarItemModel() { Command =
ToolbarCommand.SourceCode },<br>#160;#160;#160;#160;new ToolbarItemModel() { Command
= ToolbarCommand.Undo },<br>#160;#160;#160;#160;new ToolbarItemModel() { Command =
ToolbarCommand.Redo }<br>};

```

The tools order can be customized as your application requirement. If you are not specifying any tools order, the editor will create the toolbar with default items.

Custom Tool

The Rich Text Editor allows to configure your own tools to its toolbar using

`RichTextEditorCustomToolbarItems` tag directive with in a `RichTextEditorToolbarSettings`. The tools can be plain text, icon, HTML template. You can also define the order and group where the tool should be included.

This sample shows how to add your own tools to toolbar of the Rich Text Editor. The `Ω` command is added to insert special characters in the editor.

Refer to the following code snippet for custom tool with tooltip text which will be included in `Items` field of `RichTextEditorToolbarSettings` property.

ASPX-CS

```

@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorToolbarSettings Items="@Tools">
<RichTextEditorCustomToolbarItems>
<RichTextEditorCustomToolbarItem Name="Symbol">
<Template>
<SfButton @onclick="ClickHandler">Ω</SfButton>
</Template>
</RichTextEditorCustomToolbarItem>
</RichTextEditorCustomToolbarItems>
</RichTextEditorToolbarSettings>
<p>The Rich Text Editor component is WYSIWYG ('what you see is what you
get') editor that provides the best user experience to create and update the
content. Users can format their content using standard toolbar commands.</p>
<p><b>Key features:</b></p>
<ul>
<li><p>Provides <b>IFRAME</b> and <b>DIV</b> modes </p></li>
<li><p>Capable of handling markdown editing.</p></li>
<li><p>Contains a modular library to load the necessary functionality on
demand.</p></li>
<li><p>Provides a fully customizable toolbar.</p></li>
<li><p>Provides HTML view to edit the source directly for
developers.</p></li>
<li><p>Supports third - party library integration.</p></li>
</ul>
</SfRichTextEditor>
@code {

```

```
private List<ToolbarItemModel> Tools = new List<ToolbarItemModel>()
{
    new ToolbarItemModel() { Command = ToolbarCommand.Bold },
    new ToolbarItemModel() { Command = ToolbarCommand.Italic },
    new ToolbarItemModel() { Command = ToolbarCommand.Underline },
    new ToolbarItemModel() { Command = ToolbarCommand.Separator },
    new ToolbarItemModel() { Name = "Symbol", TooltipText = "Insert Symbol" },
    new ToolbarItemModel() { Command = ToolbarCommand.SourceCode },
    new ToolbarItemModel() { Command = ToolbarCommand.FullScreen }
};
private void ClickHandler()
{
    //Perform your action here
}
}
```

Quick inline toolbar

Quick commands are opened as context-menu on clicking the corresponding element. The commands must be passed to image, link and table attributes of the `RichTextEditorQuickToolbarSettings` property.

| Target Element | Default Quick Toolbar items |

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

| Image | 'Replace', 'Align', 'Caption', 'Remove', 'InsertLink', 'Display', 'AltText', 'Dimension'. |

| Link | 'Open', 'Edit', 'UnLink'. |

| Table | 'TableHeader', 'TableRows', 'TableColumns', 'BackgroundColor', 'TableRemove', 'Alignments', 'TableCellVerticalAlign', 'Styles'. |

The following sample demonstrates the option to insert the image to the Rich Text Editor content as well as option to rotate the image through the quick toolbar. The image rotation functionalities have been achieved through the `OnToolbarClick` event.

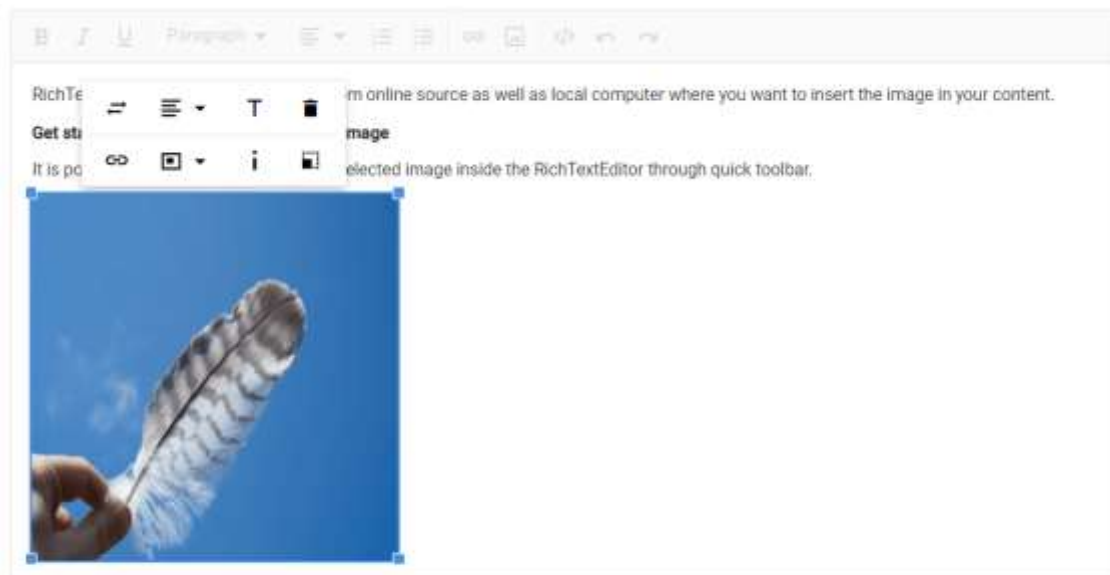
ASPX-CS

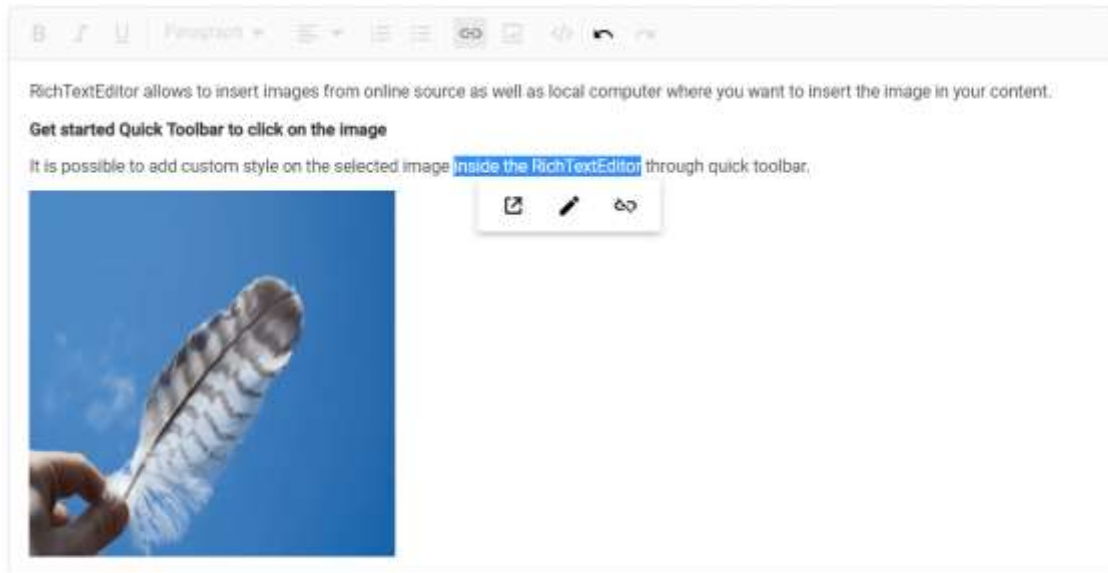
```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorQuickToolbarSettings Image="@Image" Link="@Link" />
<p>Rich Text Editor allows to insert images from online source as well as
local computer where you want to insert the image in your content.</p>
<p><b>Get started Quick Toolbar to click on the image</b></p>
<p>It is possible to add custom style on the selected image inside the Rich
Text Editor through quick toolbar.</p>
<img alt='Logo' style='width: 300px; height: 300px; transform:
rotate(0deg);'
src='https://blazor.syncfusion.com/demos/images/RichTextEditor/RTEImage-
Feather.png' />
</SfRichTextEditor>
@code {
private List<ImageToolbarItemModel> Image = new
List<ImageToolbarItemModel>()
{
    new ImageToolbarItemModel() { Command = ImageToolbarCommand.Replace },
    new ImageToolbarItemModel() { Command = ImageToolbarCommand.Align },
```

```

new ImageToolbarItemModel() { Command = ImageToolbarCommand.Caption },
new ImageToolbarItemModel() { Command = ImageToolbarCommand.Remove },
new ImageToolbarItemModel() { Command = ImageToolbarCommand.OpenImageLink },
new ImageToolbarItemModel() { Command =
ImageToolbarCommand.HorizontalSeparator },
new ImageToolbarItemModel() { Command = ImageToolbarCommand.EditImageLink },
new ImageToolbarItemModel() { Command = ImageToolbarCommand.RemoveImageLink
},
new ImageToolbarItemModel() { Command = ImageToolbarCommand.Display },
new ImageToolbarItemModel() { Command = ImageToolbarCommand.AltText },
new ImageToolbarItemModel() { Command = ImageToolbarCommand.Dimension }
};
private List<LinkToolbarItemModel> Link = new List<LinkToolbarItemModel>()
{
new LinkToolbarItemModel() { Command = LinkToolbarCommand.Open },
new LinkToolbarItemModel() { Command = LinkToolbarCommand.Edit },
new LinkToolbarItemModel() { Command = LinkToolbarCommand.UnLink }
};
}

```





You can refer to our [Blazor Rich Text Editor](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor Rich Text Editor](#) example to know how to render and configure the rich text editor tools.

See Also

- [How to render the toolbar in inline mode](#)

Inline Mode in Blazor RichTextEditor Component

This is the inline example for the Rich Text Editor. For this, you must set the `RichTextEditorInlineMode` property. Inline editing allows to select any editable element or click the element on the page and edit it in-place. Inline editing is a true WYSIWYG formation and on the contrary to Rich Text Editor HTML/MD editing, the styles that are used for edited content comes directly from the document stylesheet. This means that inline editors ignore the default Rich Text Editor content styles.

Show on select/click

Enabling the `ShowOnSelection` option of `RichTextEditorInlineMode` makes the inline Rich Text Editor to appear. You can select the text in the editable area, otherwise the inline Rich Text Editor will be appeared after clicking the editable area.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorInlineMode Enable="true" ShowOnSelection="true" />
<p>The sample is configured with inline mode of editor. Initially, the
editor is rendered without a
<a href='https://blazor.syncfusion.com/home/' target='_blank'>toolbar</a>.
The toolbar becomes visible only when the content is selected.</p>
</SfRichTextEditor>
```



You can refer to our [Blazor Rich Text Editor](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor Rich Text Editor](#) example to know how to render and configure the rich text editor tools.

See Also

- [How to edit the quick toolbar settings](#)
- [How to insert link editing option in the toolbar items](#)
- [How to insert image editing option in the toolbar items](#)

Paste from Microsoft Word in Blazor RichTextEditor Component

The Rich Text Editor allows to reduce the effort while converting the Microsoft Word content to HTML format with format and styles.

Microsoft Word to HTML

By default, the Rich Text Editor consider the following processes on paste content from Microsoft Word.

List conversion: The list elements copied from the Microsoft Word document contains paragraph tags with styles and classes. The list elements are converted to standard HTML list elements by referring the styles and class names in the paragraph tags.

Converting style: The styles of the elements copied from the Microsoft Word document are converted to standard CSS styles and added as inline styles for each respective element.

Tags and comments: The Microsoft Word specific XML tags and comments are removed when cleanup on paste.

Paste cleanup

You can control the formatting and styles on pasting the content to the editor using the pasteCleanup settings property. The following settings are available to clean up the content:

API	Description	Default Value	Type
Prompt	To invoke prompt dialog with paste options on pasting the content in editor.	false	boolean
PlainText	To paste the content as plain text.	false	boolean
KeepFormat	To keep the same format with copied content.	true	boolean
DeniedTags	To ignore the tags when pasting HTML content.	null	string[]
DeniedAttributes	To paste the content by filtering out these attributes from the content.	null	string[]

| [AllowedStyleProperties](#) | To paste the content by accepting these style attributes and removing other style attributes. | ['background', 'background-color', 'border', 'border-bottom', 'border-left', 'border-radius', 'border-right', 'border-style', 'border-top', 'border-width', 'clear', 'color', 'cursor', 'direction', 'display', 'float', 'font', 'font-family', 'font-size', 'font-weight', 'font-style', 'height', 'left', 'line-height', 'margin', 'margin-top', 'margin-left', 'margin-right', 'margin-bottom', 'max-height', 'max-width', 'min-height', 'min-width', 'overflow', 'overflow-x', 'overflow-y', 'padding', 'padding-bottom', 'padding-left', 'padding-right', 'padding-top', 'position', 'right', 'table-layout', 'text-align', 'text-decoration', 'text-indent', 'top', 'vertical-align', 'visibility', 'white-space', 'width'] | string[] |

To use paste cleanup, configure paste cleanup using the [RichTextEditorPasteCleanupSettings](#).

Prompt dialog

When [Prompt](#) is set to true, pasting the content in the editor will open a dialog box that contains three options [Keep](#), [Clean](#), and [Plain Text](#) as radio buttons:

1. [Keep](#): Radio button to keep the same format with copied content.
2. [Clean](#): Radio button to clear all the style formats with copied content.
3. [Plain Text](#): Radio button to paste the copied content as plain text without any formatting or style (including the removal of all tags).

When [Prompt](#) value is set true, the API properties [PlainText](#) and [KeepFormat](#) will not be considered for processing when pasting the content.

Paste as plain text

When [PlainText](#) is set to true, the copied content will be converted as plain text by removing all the HTML tags and styles applied to it and only the plain text is pasted in the editor.

When [PlainText](#) value is set true, the API property [Prompt](#) should be set to false, and [KeepFormat](#) will not be considered for processing when pasting the content.

Keep format

When [KeepFormat](#) is set to true, the copied content will maintain all the style formatting allowed in the [AllowedStyleProperties](#) on pasting the content in the editor.

When [KeepFormat](#) is set to false, the style in the copied content will be removed without considering the allowed styles in the [AllowedStyleProperties](#) when pasting the content in the editor.

When [KeepFormat](#) value is set true, the API property [Prompt](#) and [PlainText](#) should be set to false.

Denied tags

When [DeniedTags](#) values are set, the tags that matches the 'denied tags' list will be removed on pasting the copied content in the editor. For Example,

1. ['a'](#): Paste the content by filtering out anchor tags.
2. ['a\[!href\]'](#): Paste the content by filtering out anchor tags that do not have the 'href' attribute.
3. ['a\[href, target\]'](#): Paste the content by filtering out anchor tags that have the 'href' and 'target' attributes.

Denied attributes

When the `DeniedAttributes` values are set, the attributes that matches the 'denied attributes' list will be removed on pasting the copied content in the editor. For Example,

`'id', 'title'`: This will remove the attributes 'id' and 'title' from all tags.

Allowed style properties

By default, the following basic styles are allowed on pasting the content to the editor.

`['background', 'background-color', 'border', 'border-bottom', 'border-left', 'border-radius', 'border-right', 'border-style', 'border-top', 'border-width', 'clear', 'color', 'cursor', 'direction', 'display', 'float', 'font', 'font-family', 'font-size', 'font-weight', 'font-style', 'height', 'left', 'line-height', 'margin', 'margin-top', 'margin-left', 'margin-right', 'margin-bottom', 'max-height', 'max-width', 'min-height', 'min-width', 'overflow', 'overflow-x', 'overflow-y', 'padding', 'padding-bottom', 'padding-left', 'padding-right', 'padding-top', 'position', 'right', 'table-layout', 'text-align', 'text-decoration', 'text-indent', 'top', 'vertical-align', 'visibility', 'white-space', 'width']`

When you configure `AllowedStyleProperties`, the styles which matches the 'allowed style properties' list are allowed, all other style properties will be removed on pasting the content in the editor.

For Example,

`public string[] AllowedStyles = new string[] { "color", "margin" };` This will allow only the style properties 'color' and 'margin' in each pasted element.

In the following example, the paste cleanup related settings are explained with configuration.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorPasteCleanupSettings Prompt="true" PlainText="false"
KeepFormat="false" DeniedTags="@DeniedTag"
DeniedAttributes="@DeniedAttributes" AllowedStyleProperties="@AllowedStyles"
/>
<p>Rich Text Editor is a WYSIWYG editing control which will reduce the
effort for users while trying to express their formatting word content as
HTML or Markdown format.</p>
<p><b>Paste Cleanup properties:</b></p>
<ul>
<li><p>prompt - specifies whether to enable the prompt when pasting in Rich
Text Editor.</p></li>
<li><p>plainText - specifies whether to paste as plain text or not in Rich
Text Editor.</p></li>
<li><p>keepFormat- specifies whether to keep or remove the format when
pasting in Rich Text Editor.</p></li>
<li><p>deniedTags - specifies the tags to restrict when pasting in Rich Text
Editor.</p></li>
<li><p>deniedAttributes - specifies the attributes to restrict when pasting
in Rich Text Editor.</p></li>
<li><p>allowedStyleProperties - specifies the allowed style properties when
pasting in Rich Text Editor.</p></li>
</ul>
</SfRichTextEditor>
@code {
private string[] DeniedTag = new string[] { "a" };
```

```
private string[] DeniedAttributes = new string[] { "class", "title", "id" };
private string[] AllowedStyles = new string[] { "color", "margin", "font-size" };
}
```



You can refer to our [Blazor Rich Text Editor](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor Rich Text Editor](#) example to know how to render and configure the rich text editor tools.

Styling in Blazor RichTextEditor Component

Font name and size

By default, the editor is initialized with default items of font name and font size. To change it, select a different font name and font size from the drop-down in the editor's toolbar.

To apply different font style for section of the content, select the text that you would like to change, and select a required font style from the drop-down to apply the changes to the selected text.

Font name

The following table, lists the default font name and width of the **FontName** dropdown and available list of font names.

Default Key	Default Value
-----	-----
Default	null
Width	65px

```
Items | new List<DropDownItemModel>() {
    new DropDownItemModel() { Text = "Segoe UI", Value = "Segoe UI" },
    new DropDownItemModel() { Text = "Arial", Value = "Arial,Helvetica,sans-serif" },
    new DropDownItemModel() { Text = "Georgia", Value = "Georgia,serif" },
    new DropDownItemModel() { Text = "Impact", Value = "Impact,Charcoal,sans-serif" },
    new DropDownItemModel() { Text = "Tahoma", Value = "Tahoma,Geneva,sans-serif" },
    new DropDownItemModel() { Text = "Times New Roman", Value = "Times New Roman,Times,serif" }
}
```



```
new DropDownItemModel() { Text = "Verdana", Value = "Verdana,Geneva,sans-serif"}<br>|
```

Font size

The following table list the default font size and width of the **FontSize** dropdown and available list of font size.

Default Key	Default Value
----	-----
Default	null
Width	35px.

```
| Items | new List<DropDownItemModel>()<br><br>#160;#160;#160;#160;new  
DropDownItemModel() { Text = "8 pt", Value = "8pt" },<br>#160;#160;#160;#160;new  
DropDownItemModel() { Text = "10 pt", Value = "10pt" },<br>#160;#160;#160;#160;new  
DropDownItemModel() { Text = "12 pt", Value = "12pt" },<br>#160;#160;#160;#160;new  
DropDownItemModel() { Text = "14 pt", Value = "14pt" },<br>#160;#160;#160;#160;new  
DropDownItemModel() { Text = "18 pt", Value = "18pt" },<br>#160;#160;#160;#160;new  
DropDownItemModel() { Text = "24 pt", Value = "24pt" },<br>#160;#160;#160;#160;new  
DropDownItemModel() { Text = "36 pt", Value = "36pt" }<br>; |
```

The following sample demonstrates the option to add the font name and font size tools to the toolbar as well as modify the default Width of the tools.

CSHARP

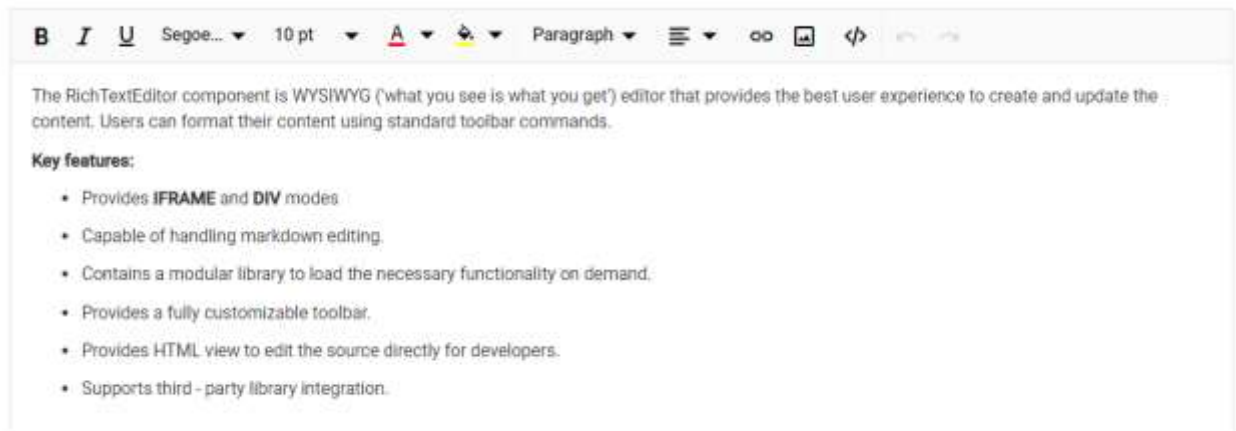
```
@using Microsoft.Bazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorToolBarSettings Items="@Tools" />
<RichTextEditorFontFamily Width="50px" />
<RichTextEditorFontSize Width="50px" />
<p>The Rich Text Editor component is WYSIWYG ('what you see is what you
get') editor that provides the best user experience to create and update the
content. Users can format their content using standard toolbar commands.</p>
<p><b>Key features:</b></p>
<ul>
<li><p>Provides <b>IFRAME</b> and <b>DIV</b> modes </p></li>
<li><p>Capable of handling markdown editing.</p></li>
<li><p>Contains a modular library to load the necessary functionality on
demand.</p></li>
<li><p>Provides a fully customizable toolbar.</p></li>
<li><p>Provides HTML view to edit the source directly for
developers.</p></li>
<li><p>Supports third - party library integration.</p></li>
</ul>
</SfRichTextEditor>
@code{
private List<ToolBarItemModel> Tools = new List<ToolBarItemModel>()
{
new ToolBarItemModel() { Command = ToolbarCommand.Bold },
new ToolBarItemModel() { Command = ToolbarCommand.Italic },
new ToolBarItemModel() { Command = ToolbarCommand.Underline },
new ToolBarItemModel() { Command = ToolbarCommand.FontName },
new ToolBarItemModel() { Command = ToolbarCommand.FontSize },

```

```

new ToolbarItemModel () { Command = ToolbarCommand.FontColor },
new ToolbarItemModel () { Command = ToolbarCommand.BackgroundColor },
new ToolbarItemModel () { Command = ToolbarCommand.Separator },
new ToolbarItemModel () { Command = ToolbarCommand.Formats },
new ToolbarItemModel () { Command = ToolbarCommand.Alignments },
new ToolbarItemModel () { Command = ToolbarCommand.Separator },
new ToolbarItemModel () { Command = ToolbarCommand.CreateLink },
new ToolbarItemModel () { Command = ToolbarCommand.Image },
new ToolbarItemModel () { Command = ToolbarCommand.Separator },
new ToolbarItemModel () { Command = ToolbarCommand.SourceCode },
new ToolbarItemModel () { Command = ToolbarCommand.Separator },
new ToolbarItemModel () { Command = ToolbarCommand.Undo },
new ToolbarItemModel () { Command = ToolbarCommand.Redo }
};
}

```



Custom font and size

Rich Text Editor provides support to custom fonts and size with existing list.

If you want to add additional font names and font sizes to font drop-down, pass the font information as `List<DropDownItemModel>` data to the `Items` field of `RichTextEditorFontSize` and `RichTextEditorFontFamily` tag.

CSHARP

```

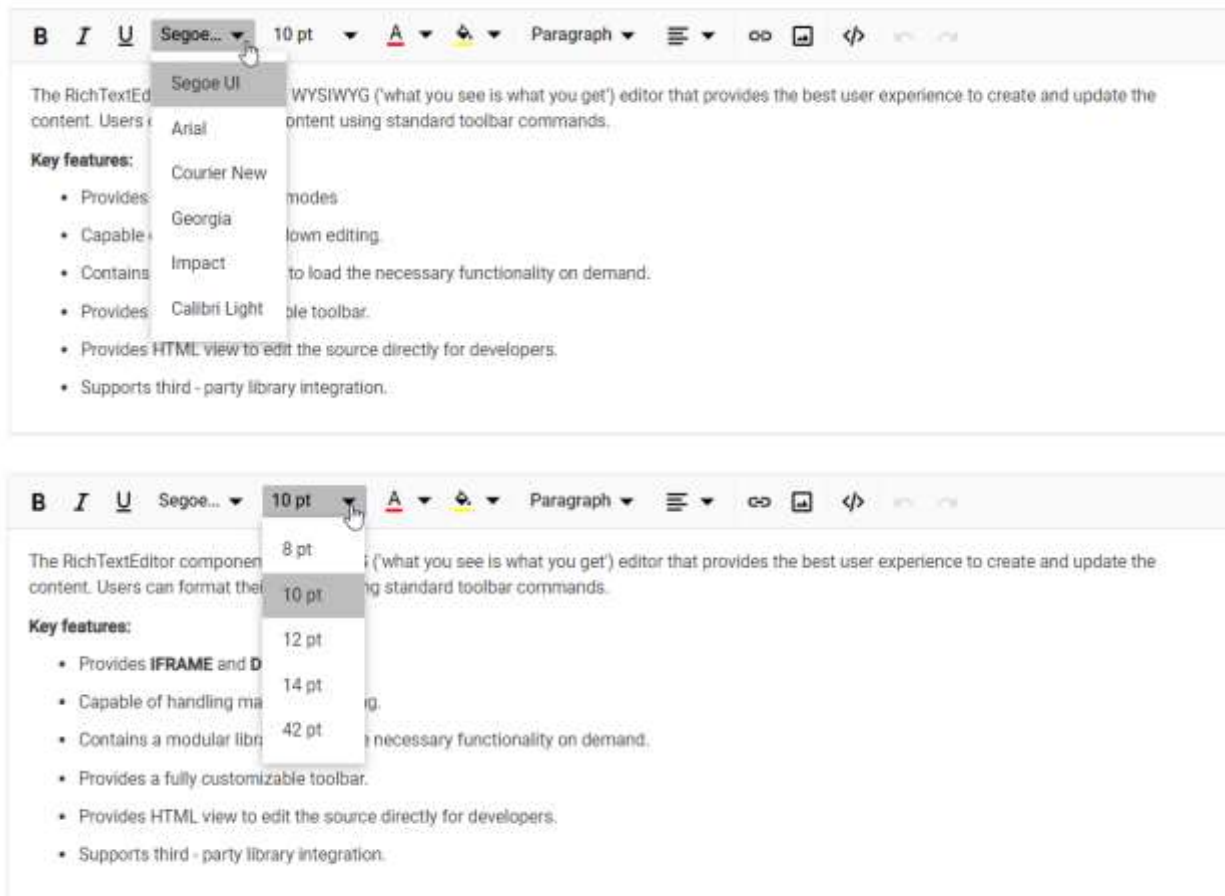
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorToolbarSettings Items="@Tools" />
<RichTextEditorFontFamily Width="50px" Items="@FontFamilyItems" />
<RichTextEditorFontSize Width="50px" Items="@FontSizeItems" />
<p>The Rich Text Editor component is WYSIWYG ('what you see is what you get') editor that provides the best user experience to create and update the content. Users can format their content using standard toolbar commands.</p>
<p><b>Key features:</b></p>
<ul>
<li><p>Provides <b>IFRAME</b> and <b>DIV</b> modes </p></li>
<li><p>Capable of handling markdown editing.</p></li>
<li><p>Contains a modular library to load the necessary functionality on demand.</p></li>
<li><p>Provides a fully customizable toolbar.</p></li>

```

```

<li><p> Provides HTML view to edit the source directly for
developers.</p></li>
<li><p> Supports third - party library integration.</p></li>
</ul>
</SfRichTextEditor>
@code{
private List<ToolbarItemModel> Tools = new List<ToolbarItemModel>()
{
new ToolbarItemModel() { Command = ToolbarCommand.Bold },
new ToolbarItemModel() { Command = ToolbarCommand.Italic },
new ToolbarItemModel() { Command = ToolbarCommand.Underline },
new ToolbarItemModel() { Command = ToolbarCommand.FontName },
new ToolbarItemModel() { Command = ToolbarCommand.FontSize },
new ToolbarItemModel() { Command = ToolbarCommand.FontColor },
new ToolbarItemModel() { Command = ToolbarCommand.BackgroundColor },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.Formats },
new ToolbarItemModel() { Command = ToolbarCommand.Alignments },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.CreateLink },
new ToolbarItemModel() { Command = ToolbarCommand.Image },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.SourceCode },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.Undo },
new ToolbarItemModel() { Command = ToolbarCommand.Redo }
};
private List<DropDownItemModel> FontFamilyItems = new
List<DropDownItemModel>()
{
new DropDownItemModel() { Text = "Segoe UI", Value = "Arial,Helvetica,sans-
serif" },
new DropDownItemModel() { Text = "Arial", Value = "Roboto" },
new DropDownItemModel() { Text = "Georgia", Value = "Georgia,serif" },
new DropDownItemModel() { Text = "Impact", Value = "Impact,Charcoal,sans-
serif" },
new DropDownItemModel() { Text = "Tahoma", Value = "Tahoma,Geneva,sans-
serif" }
};
private List<DropDownItemModel> FontSizeItems = new
List<DropDownItemModel>()
{
new DropDownItemModel() { Text = "8 pt", Value = "8pt" },
new DropDownItemModel() { Text = "10 pt", Value = "10pt" },
new DropDownItemModel() { Text = "12 pt", Value = "12pt" },
new DropDownItemModel() { Text = "14 pt", Value = "14pt" },
new DropDownItemModel() { Text = "42 pt", Value = "42pt" }
};
}

```



Formats

By default, the editor is initialized with default items of formats. To change it, select a different format from the drop-down in the editor's toolbar.

To apply different format style for section of the content, select a required format from the drop-down to apply the changes to the selection.

The following table, lists the default format name and width of the **Format** dropdown and available list of format names.

Default Key	Default Value
-----	-----
Default	null
Width	65px

```
| Items | new List<DropDownItemModel>(){  
    new DropDownItemModel() { Text = "Paragraph", Value = "P" },  
    new DropDownItemModel() { Text = "Code", Value = "Pre" },  
    new DropDownItemModel() { Text = "Quotation", Value = "BlockQuote" },  
    new DropDownItemModel() { Text = "Heading 1", Value = "H1" },  
    new DropDownItemModel() { Text = "Heading 2", Value = "H2" },  
    new DropDownItemModel() { Text = "Heading 3", Value = "H3" }
```

```
},<br>&#160;&#160;&#160;&#160;new DropDownListModel() { Text = "Heading 4", Value = "H4"
}<br>}; |
```

Custom formats

Rich Text Editor provides support to custom formats with existing list.

If you want to add additional formats to format drop-down, pass the format information as `List<DropDownItemModel>` data to the `Items` field of `RichTextEditorFormat` tag.

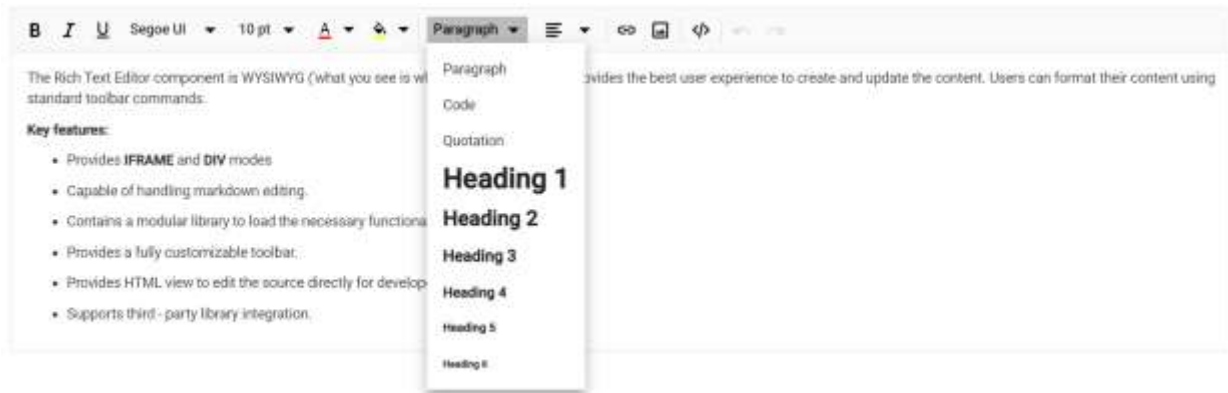
CSHARP

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorToolbarSettings Items="@Tools" />
<RichTextEditorFormat Items="@FormatItems" />
<p>The Rich Text Editor component is WYSIWYG ('what you see is what you
get') editor that provides the best user experience to create and update the
content. Users can format their content using standard toolbar commands.</p>
<p><b>Key features:</b></p>
<ul>
<li><p>Provides <b>IFRAME</b> and <b>DIV</b> modes </p></li>
<li><p>Capable of handling markdown editing.</p></li>
<li><p>Contains a modular library to load the necessary functionality on
demand.</p></li>
<li><p>Provides a fully customizable toolbar.</p></li>
<li><p>Provides HTML view to edit the source directly for
developers.</p></li>
<li><p>Supports third - party library integration.</p></li>
</ul>
</SfRichTextEditor>
@code{
private List<ToolbarItemModel> Tools = new List<ToolbarItemModel>()
{
new ToolbarItemModel() { Command = ToolbarCommand.Bold },
new ToolbarItemModel() { Command = ToolbarCommand.Italic },
new ToolbarItemModel() { Command = ToolbarCommand.Underline },
new ToolbarItemModel() { Command = ToolbarCommand.FontName },
new ToolbarItemModel() { Command = ToolbarCommand.FontSize },
new ToolbarItemModel() { Command = ToolbarCommand.FontColor },
new ToolbarItemModel() { Command = ToolbarCommand.BackgroundColor },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.Formats },
new ToolbarItemModel() { Command = ToolbarCommand.Alignments },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.CreateLink },
new ToolbarItemModel() { Command = ToolbarCommand.Image },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.SourceCode },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.Undo },
new ToolbarItemModel() { Command = ToolbarCommand.Redo }
};
private List<DropDownItemModel> FormatItems = new List<DropDownItemModel>()
{
new DropDownItemModel() { Text = "Paragraph", Value = "P" },
new DropDownItemModel() { Text = "Code", Value = "Pre" },
new DropDownItemModel() { Text = "Quotation", Value = "BlockQuote" },
```

```

new DropDownItemModel () { Text = "Heading 1", Value = "H1" },
new DropDownItemModel () { Text = "Heading 2", Value = "H2" },
new DropDownItemModel () { Text = "Heading 3", Value = "H3" },
new DropDownItemModel () { Text = "Heading 4", Value = "H4" },
new DropDownItemModel () { Text = "Heading 5", Value = "H5" },
new DropDownItemModel () { Text = "Heading 6", Value = "H6" }
};
}

```



Font and Background color

To apply font color or background color for a selected content of RTE, use the font color and background color tools.

Rich Text Editor support to provide custom font color and background color with existing list through the `ColorCode` field of `RichTextEditorFontColor` and `RichTextEditorBackgroundColor`.

The `RichTextEditorFontColor` and `RichTextEditorBackgroundColor` tag has two **Mode** of **Picker** and **Palette**. Palette mode has predefined set of `ColorCode` and in the picker mode, more colors have been provided. Through `ModeSwitcher`, you can able to switch between these two options.

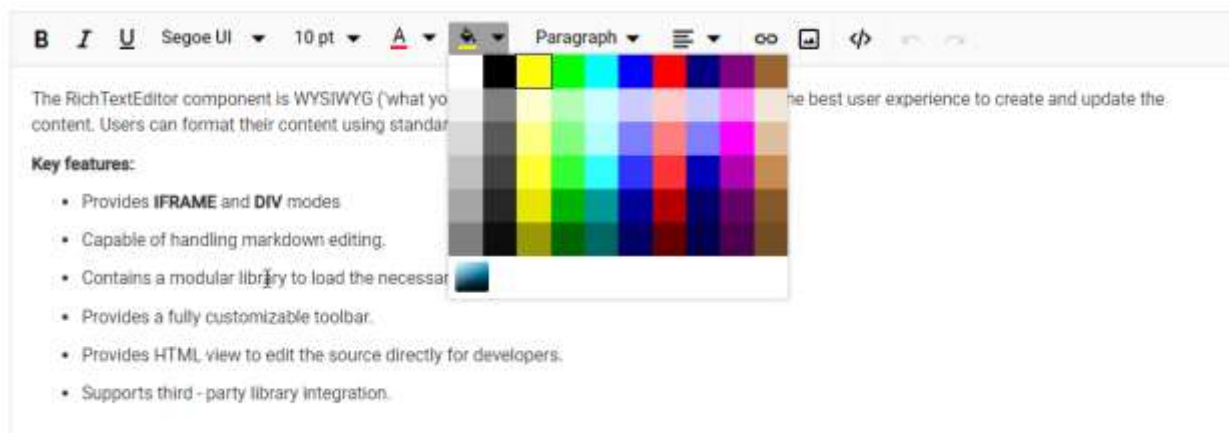
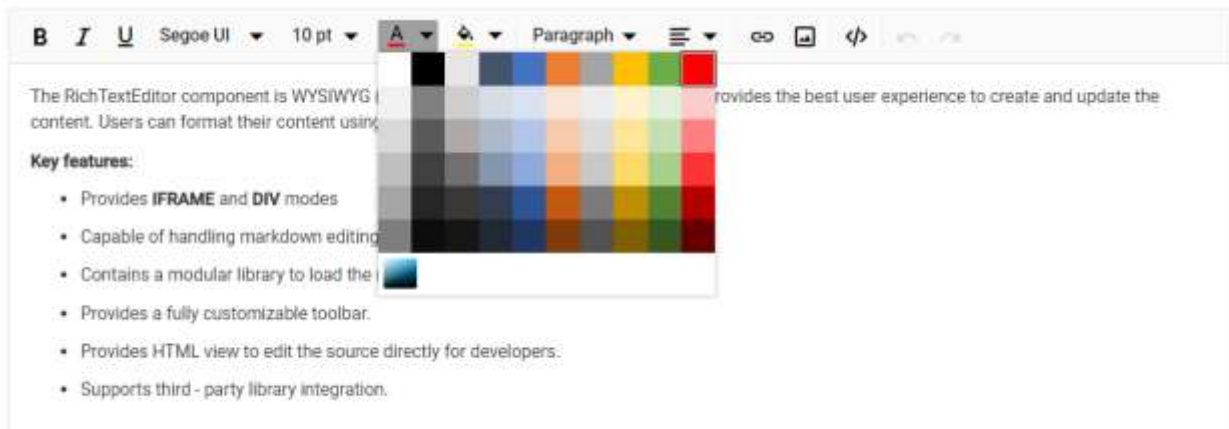
C# SHARP

```

@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorToolbarSettings Items="@Tools" />
<RichTextEditorBackgroundColor ModeSwitcher="true" />
<RichTextEditorFontColor ModeSwitcher="true" />
<p>The Rich Text Editor component is WYSIWYG ('what you see is what you
get') editor that provides the best user experience to create and update the
content. Users can format their content using standard toolbar commands.</p>
<p><b>Key features:</b></p>
<ul>
<li><p> Provides <b>IFRAME</b> and <b>DIV</b> modes </p></li>
<li><p> Capable of handling markdown editing.</p></li>
<li><p> Contains a modular library to load the necessary functionality on
demand.</p></li>
<li><p> Provides a fully customizable toolbar.</p></li>
<li><p> Provides HTML view to edit the source directly for
developers.</p></li>
<li><p> Supports third - party library integration.</p></li>
</ul>
</SfRichTextEditor>

```

```
@code{
private List<ToolbarItemModel> Tools = new List<ToolbarItemModel>()
{
new ToolbarItemModel() { Command = ToolbarCommand.Bold },
new ToolbarItemModel() { Command = ToolbarCommand.Italic },
new ToolbarItemModel() { Command = ToolbarCommand.Underline },
new ToolbarItemModel() { Command = ToolbarCommand.FontName },
new ToolbarItemModel() { Command = ToolbarCommand.FontSize },
new ToolbarItemModel() { Command = ToolbarCommand.FontColor },
new ToolbarItemModel() { Command = ToolbarCommand.BackgroundColor },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.Formats },
new ToolbarItemModel() { Command = ToolbarCommand.Alignments },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.CreateLink },
new ToolbarItemModel() { Command = ToolbarCommand.Image },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.SourceCode },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.Undo },
new ToolbarItemModel() { Command = ToolbarCommand.Redo }
};
}
```



Editor content styles

By default, The content styles of Rich Text Editor are not returned while retrieving HTML value from the editor. So, the styles are not applied when using the HTML value outside of the editor. To get the styles to Rich Text Editor's content for your application, You can copy and use the below styles directly in your application. The styles listed below which used in the UI elements of the Rich Text Editor.

Make sure to add a CSS class 'e-rte-content' to the content container.

CSS

```
.e-rte-content p {
margin: 0 0 10px;
margin-bottom: 10px;
}
.e-rte-content li {
margin-bottom: 10px;
}
.e-rte-content h1 {
font-size: 2.17em;
font-weight: 400;
line-height: 1;
margin: 10px 0;
}
.e-rte-content h2 {
font-size: 1.74em;
font-weight: 400;
margin: 10px 0;
}
.e-rte-content h3 {
font-size: 1.31em;
font-weight: 400;
margin: 10px 0;
}
.e-rte-content h4 {
font-size: 1em;
font-weight: 400;
margin: 0;
}
.e-rte-content h5 {
font-size: 0.8em;
font-weight: 400;
margin: 0;
}
.e-rte-content h6 {
font-size: 0.65em;
font-weight: 400;
margin: 0;
}
.e-rte-content blockquote {
margin: 10px 0;
margin-left: 0;
padding-left: 5px;
}
.e-rte-content pre {
background-color: inherit;
border: 0;
border-radius: 0;
```



```
color: #333;
font-size: inherit;
line-height: inherit;
margin: 0 0 10px;
overflow: visible;
padding: 0;
white-space: pre-wrap;
word-break: inherit;
word-wrap: break-word;
}
.e-rte-content strong, .e-rte-content b {
font-weight: 700;
}
.e-rte-content a {
text-decoration: none;
-webkit-user-select: auto;
-ms-user-select: auto;
user-select: auto;
}
.e-rte-content a:hover {
text-decoration: underline;
}
.e-rte-content h3 + h4,
.e-rte-content h4 + h5,
.e-rte-content h5 + h6 {
margin-top: 0.6em;
}
.e-rte-content .e-rte-image.e-imgbreak {
border: 0;
cursor: pointer;
display: block;
float: none;
margin: 5px auto;
max-width: 100%;
position: relative;
}
.e-rte-content .e-rte-image {
border: 0;
cursor: pointer;
display: block;
float: none;
margin: auto;
max-width: 100%;
position: relative;
}
.e-rte-content .e-rte-image.e-imginline {
display: inline-block;
float: none;
margin-left: 5px;
margin-right: 5px;
max-width: calc(100% - (2 * 5px));
vertical-align: bottom;
}
.e-rte-content .e-rte-image.e-imgcenter {
cursor: pointer;
display: block;
float: none;
```

```
margin: 5px auto;
max-width: 100%;
position: relative;
}
.e-rte-content .e-rte-image.e-imgleft {
float: left;
margin: 0 5px 0 0;
text-align: left;
}
.e-rte-content .e-rte-image.e-imgright {
float: right;
margin: 0 0 0 5px;
text-align: right;
}
.e-rte-content .e-rte-img-caption {
display: inline-block;
margin: 5px auto;
max-width: 100%;
position: relative;
}
.e-rte-content .e-rte-img-caption.e-caption-inline {
display: inline-block;
margin: 5px auto;
margin-left: 5px;
margin-right: 5px;
max-width: calc(100% - (2 * 5px));
position: relative;
text-align: center;
vertical-align: bottom;
}
.e-rte-content .e-rte-img-caption.e-imgcenter {
display: block;
}
.e-rte-content .e-rte-img-caption .e-rte-image.e-imgright,
.e-rte-content .e-rte-img-caption .e-rte-image.e-imgleft {
float: none;
margin: 0;
}
.e-rte-content .e-rte-table {
border-collapse: collapse;
empty-cells: show;
}
.e-rte-content .e-rte-table td,
.e-rte-content .e-rte-table th {
border: 1px solid #bdbdbd;
height: 20px;
min-width: 20px;
padding: 2px 5px;
vertical-align: middle;
}
.e-rte-content .e-rte-table.e-dashed-border td,
.e-rte-content .e-rte-table.e-dashed-border th {
border-style: dashed;
}
.e-rte-content .e-rte-img-caption .e-img-inner {
box-sizing: border-box;
display: block;
```

```
font-size: 16px;
font-weight: initial;
margin: auto;
opacity: .9;
position: relative;
text-align: center;
width: 100%;
}
.e-rte-content .e-rte-img-caption .e-img-wrap {
display: inline-block;
margin: auto;
padding: 0;
width: 100%;
}
.e-rte-content blockquote {
border-left: solid 2px #333;
}
.e-rte-content a {
color: #2e2ef1;
}
.e-rte-content .e-rte-table th {
background-color: #e0e0e0;
}
```

You can refer to our [Blazor Rich Text Editor](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor Rich Text Editor](#) example to know how to render and configure the rich text editor tools.

See Also

- [How to add google fonts to the font family](#)
- [How to customize the placeholder](#)

Image in Blazor RichTextEditor Component

Rich Text Editor allows inserting images from online sources and local computer where you want to insert the image in your content. For inserting the image to the Rich Text Editor, the following list of options have been provided in the `RichTextEditorImageSettings`.

Options	Description
----- -----	
AllowedTypes	Specifies the extensions of the image types allowed to insert on bowering and passing the extensions with comma separators. For example, pass AllowedTypes as .jpg and .png.
Display	Sets the default display for an image when it is inserted into the Rich Text Editor. Possible options are: <code>Inline</code> and <code>Break</code> .
Width	Sets the default width of the image when it is inserted in the Rich Text Editor.
Height	Sets the default height of the image when it is inserted in the Rich Text Editor.
SaveUrl	Provides URL to map the action result method to save the image.

| Path | Specifies the location to store the image. It is of string type and it appends to the name of the file or folder where you want to store the image. For example, "api/Images/" |

| EnableResize | Enables resizing for image element. |

| MinWidth | Defines the minimum Width of the image. |

| MaxWidth | Defines the maximum Width of the image. |

| MinHeight | Defines the minimum Height of the image. |

| MaxHeight | Defines the maximum Height of the image. |

| ResizeByPercent | Image resizing should be done by percentage calculation. |

Upload options

Through the **browse** option in the Image dialog, select the image from the local machine and insert into the Rich Text Editor content.

If the path field is not specified in the **RichTextEditorImageSettings**, the image will be converted into base64 and blob url for the image will be created then generated url will be set as **src** property of **** tag as below.

The image has been loaded from the local machine and it will be saved in the given location.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorImageSettings SaveUrl="api/Image/Save" Path="./Images/" />
</SfRichTextEditor>
```

ImageController.cs

CSHARP

```
using System;
using System.IO;
using System.Net.Http.Headers;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Http;
using System.Collections.Generic;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http.Features;
namespace ImageUpload.Controllers
{
    [ApiController]
    public class ImageController : ControllerBase
    {
        private readonly IWebHostEnvironment hostingEnv;
        public ImageController(IWebHostEnvironment env)
        {
            this.hostingEnv = env;
        }
        [HttpPost("[action]")]
        [Route("api/Image/Save")]
        public void Save(IList<IFormFile> UploadFiles)
        {

```

```
try
{
    foreach (var file in UploadFiles)
    {
        if (UploadFiles != null)
        {
            string targetPath = hostingEnv.ContentRootPath + "\\wwwroot\\Images";
            string filename =
                ContentDispositionHeaderValue.Parse(file.ContentDisposition).FileName.Trim('
                ');
            // Create a new directory, if it does not exists
            if (!Directory.Exists(targetPath))
            {
                Directory.CreateDirectory(targetPath);
            }
            // Name which is used to save the image
            filename = targetPath + $"\"{filename}\"";
            if (!System.IO.File.Exists(filename))
            {
                // Upload a image, if the same file name does not exist in the directory
                using (FileStream fs = System.IO.File.Create(filename))
                {
                    file.CopyTo(fs);
                    fs.Flush();
                }
                Response.StatusCode = 200;
            }
            else
            {
                Response.StatusCode = 204;
            }
        }
    }
    catch (Exception e)
    {
        Response.Clear();
        Response.ContentType = "application/json; charset=utf-8";
        Response.HttpContext.Features.Get<IHttpResponseFeature>().ReasonPhrase =
            e.Message;
    }
}
```

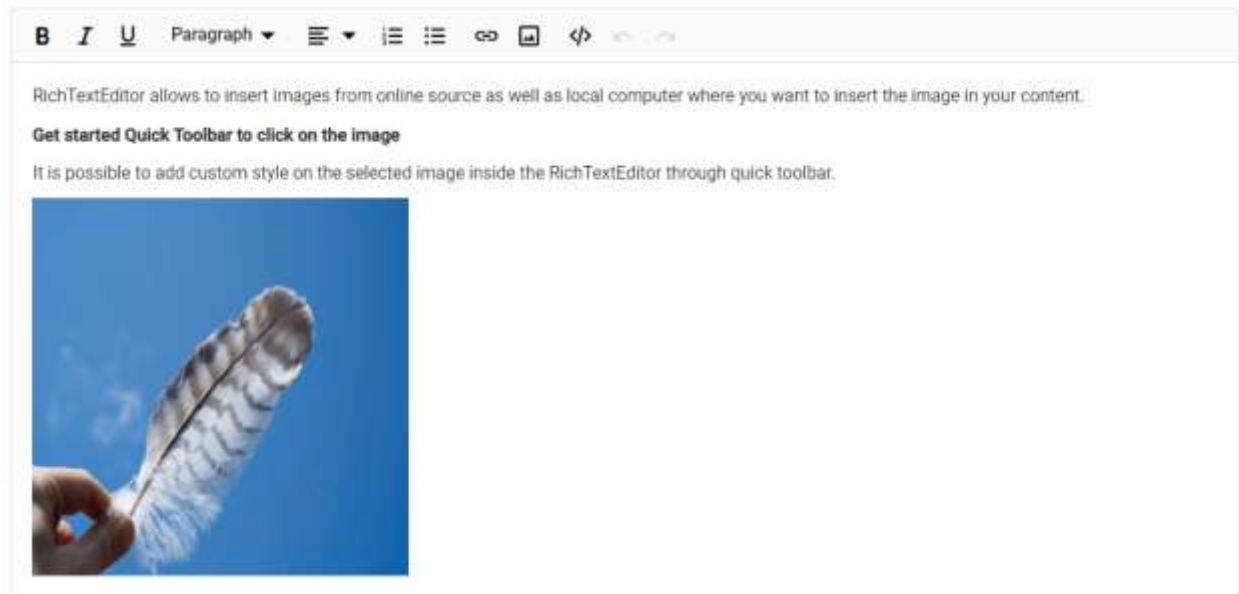
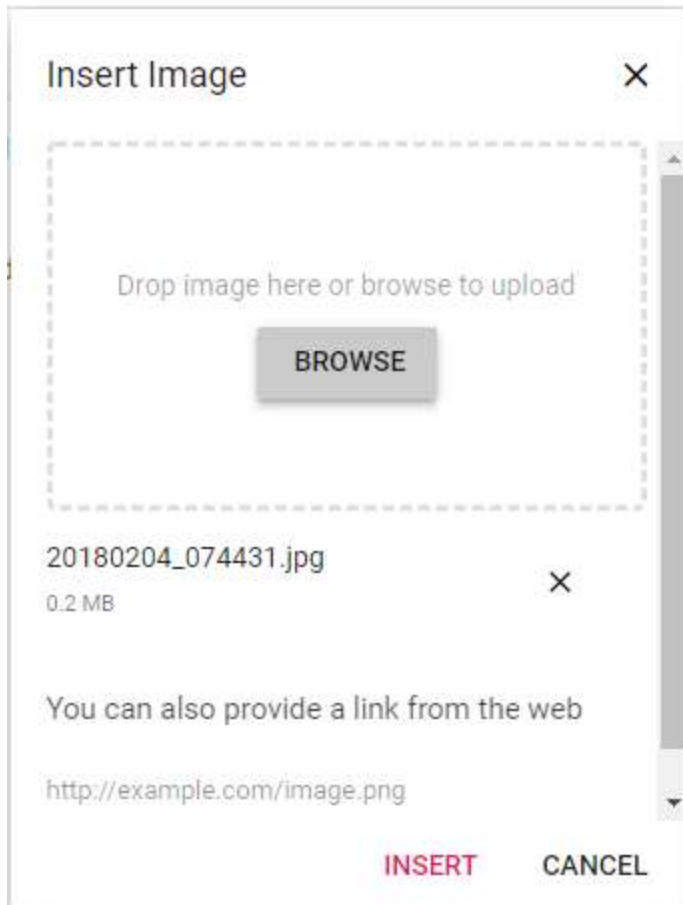


Image delete

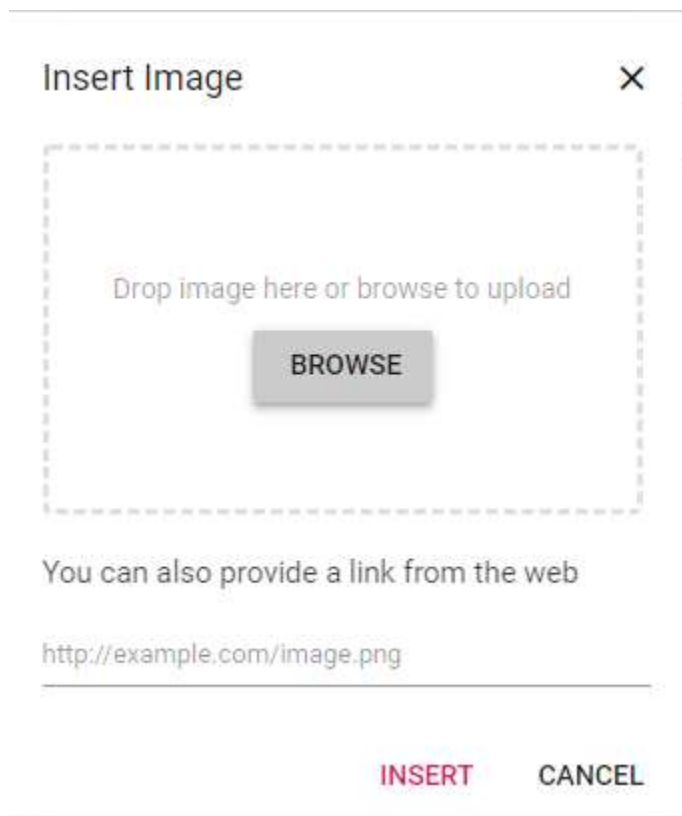
To remove an image from the Rich Text Editor content, select the image and click "Remove" tool from quick toolbar. It will delete the image from the Rich Text Editor content.

After selecting the image from the local machine, the URL for the image will be generated, from there also you can remove the image from the service location by clicking the cross icon as in the following image.



Insert from web

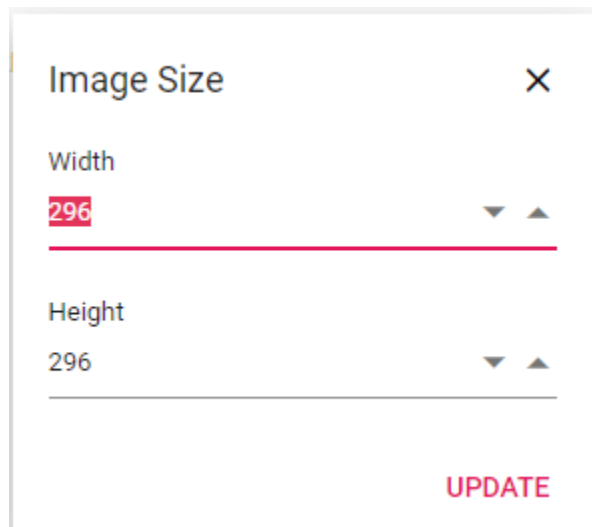
To insert an image from online source like Google, Bing, and more, enable images tool on the editor's toolbar. By default, the images tool opens an image dialog that allows to insert an image from online source.



Dimension

Sets the default width and height of the image when it is inserted in the Rich Text Editor using **Width** and **Height** of **RichTextEditorImageSettings**.

Through the **QuickToolbar** also you can change the width and height using **Change Size** option. After clicking the option, the image size will open as below. In that specify the width and height of the image in pixel.



Caption and Alt Text

Image caption and alternative text can be specified for the inserted image in the Rich Text Editor using the `RichTextEditorQuickToolbarSettings` options such as Image Caption and Alternative Text.

Through the Alternative Text option, set the alternative text for the image, when the image is not uploaded successfully into the Rich Text Editor.

By clicking the Image Caption, the image will get wrapped in an image element with a caption. Then, you can type caption content inside the Rich Text Editor.

Display position

Sets the default display for an image when it is inserted in the Rich Text Editor using `Display` field in `RichTextEditorImageSettings`.

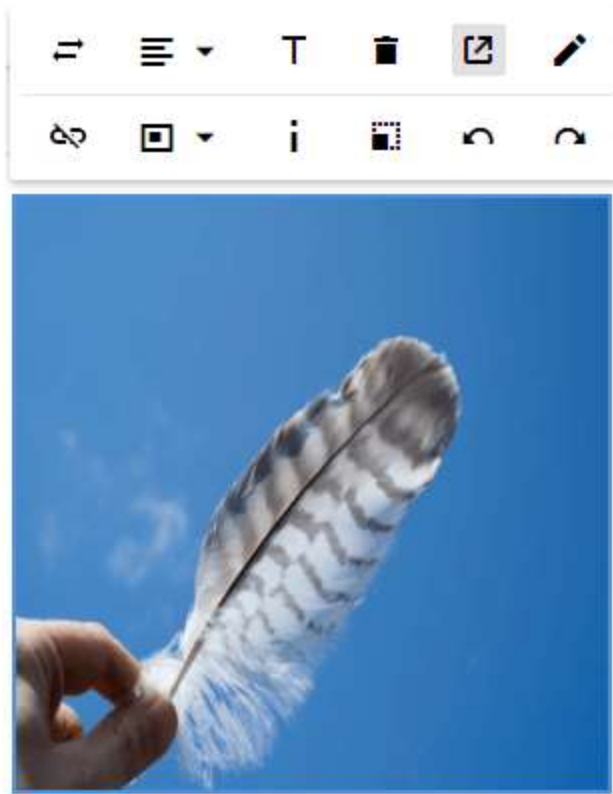
It has two possible options: `Inline` and `Break`.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorImageSettings Display="ImageDisplay.Inline" />
<p>Rich Text Editor allows to insert images from online source as well as
local computer where you want to insert the image in your content.</p>
<p><b>Get started Quick Toolbar to click on the image</b></p>
<p>It is possible to add custom style on the selected image inside the Rich
Text Editor through quick toolbar.</p>
<img alt='Logo' style='width: 300px; height: 300px; transform:
rotate(0deg); '
src='https://blazor.syncfusion.com/demos/images/RichTextEditor/RTEImage-
Feather.png' />
</SfRichTextEditor>
```

Image with link

The hyperlink itself can be an image in Rich Text Editor. If the image given as hyperlink, the remove, edit, and open links will be added to the quick toolbar of image as below. For further details about link, refer to the [link documentation](#).



Resize

Rich Text Editor has a built-in image inserting support. The resize points will be appearing on each corner of image when focus. So, users can resize the image using mouse points or thumb through the resize points easily. Also, the resize calculation will be done based on aspect ratio.



You can refer to our [Blazor Rich Text Editor](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor Rich Text Editor](#) example to know how to render and configure the rich text editor tools.

See Also

- [How to edit the quick toolbar settings](#)
- [How to use link editing option in the toolbar items](#)

Link in Blazor RichTextEditor Component

A hyperlink can be inserted into the editor for quick access to the related information. The hyperlink itself can be a text or an image.

Insert link

Point the cursor anywhere within the editor where you would like to insert the link. It is also possible to select a text or an image within the editor and can be converted to the hyperlink. Click the Insert HyperLink tool on the toolbar. The Insert Link Dialog will open. The dialog has the following options.

Options	Description
Web Address	Types or pastes the destination for the link you are creating
Display Text	Types or edits the required text that you want to display text for the link
Tooltip	Displays additional helpful information when you place the pointer on the hyperlink, type the required text in the "Tooltip" field.
Open Link in New Window	Specifies whether the given link will open in new window or not

The Rich Text Editor link tool validates the URLs as you type them in Web Address. URLs considered invalid will be highlighted with red color by clicking the insert button in the **Insert Link** dialog.

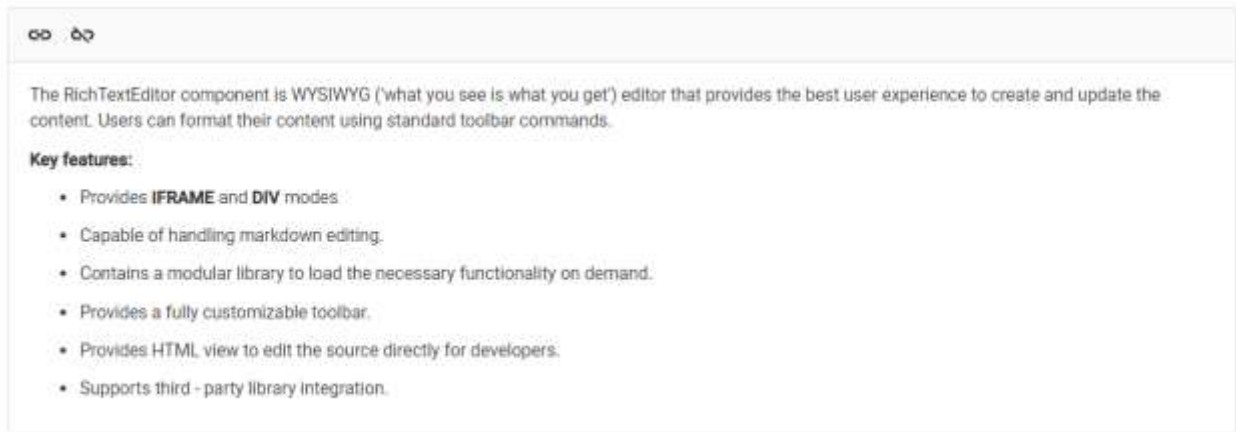
ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorToolbarSettings Items="@Tools" />
<p>The Rich Text Editor component is WYSIWYG ('what you see is what you
get') editor that provides the best user experience to create and update the
content. Users can format their content using standard toolbar commands.</p>
```

```

<p><b> Key features:</b></p>
<ul>
<li><p> Provides <b>IFRAME</b> and <b>DIV</b> modes </p></li>
<li><p> Capable of handling markdown editing.</p></li>
<li><p> Contains a modular library to load the necessary functionality on demand.</p></li>
<li><p> Provides a fully customizable toolbar.</p></li>
<li><p> Provides HTML view to edit the source directly for developers.</p></li>
<li><p> Supports third - party library integration.</p></li>
</ul>
</SfRichTextEditor>
@code {
private List<ToolbarItemModel> Tools = new List<ToolbarItemModel>()
{
new ToolbarItemModel() { Command = ToolbarCommand.CreateLink },
new ToolbarItemModel() { Command = ToolbarCommand.RemoveLink }
};
}

```



Remove Link

To remove a hyperlink from a text or image, select the text or image with the hyperlink and click **Remove Hyperlink** tool from the toolbar. It will keep the text or image.

Auto-link

When you type URL and enter key to the Rich Text Editor, the typed URL will be automatically changed into the hyperlink.

Manipulation

Add the custom tools on the selected link inside the Rich Text Editor through the quick toolbar.



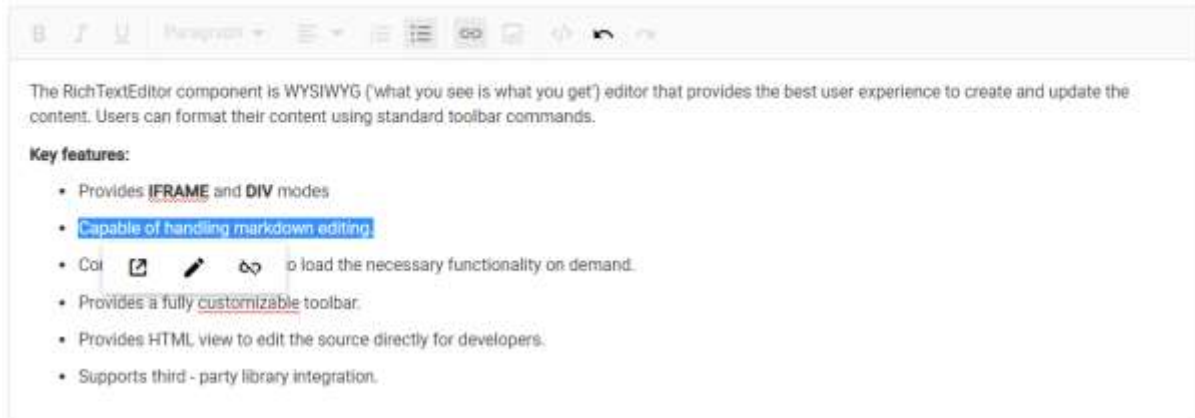
The quick toolbar for the link has the following options.

Tools	Description
-----	-----

- | Open | The given link page will open in new window. |
- | Edit Link | Edits the link in the Rich Text Editor content. |
- | Remove Link | Removes link from the content of Rich Text Editor. |

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorQuickToolbarSettings Link="@Link" />
<p>The Rich Text Editor component is WYSIWYG ('what you see is what you
get') editor that provides the best user experience to create and update the
content. Users can format their content using standard toolbar commands.</p>
<p><b>Key features:</b></p>
<ul>
<li><p>Provides <b>IFRAME</b> and <b>DIV</b> modes </p></li>
<li><p>Capable of handling markdown editing.</p></li>
<li><p>Contains a modular library to load the necessary functionality on
demand.</p></li>
<li><p>Provides a fully customizable toolbar.</p></li>
<li><p>Provides HTML view to edit the source directly for
developers.</p></li>
<li><p>Supports third - party library integration.</p></li>
</ul>
</SfRichTextEditor>
@code {
private List<LinkToolbarItemModel> Link = new List<LinkToolbarItemModel>()
{
new LinkToolbarItemModel() { Command = LinkToolbarCommand.Open },
new LinkToolbarItemModel() { Command = LinkToolbarCommand.Edit },
new LinkToolbarItemModel() { Command = LinkToolbarCommand.UnLink }
};
}
```



You can refer to our [Blazor Rich Text Editor](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor Rich Text Editor](#) example to know how to render and configure the rich text editor tools.

See Also

- [How to edit the quick toolbar settings](#)
- [How to insert link editing option in the toolbar items](#)
- [How to insert image editing option in the toolbar items](#)

Table in Blazor RichTextEditor Component

Rich Text Editor allows to insert table of content in edit panel and provide options to add, edit, and remove the table as well as perform other table related action. For inserting the table to the Rich Text Editor, the following list of options have been provided in the `RichTextEditorTableSettings`

Options	Description	Default Value
MinWidth	Sets the default minWidth of the table.	0
MaxWidth	Sets the default maxWidth of the table.	null
EnableResize	Enables resize feature in table.	true
Styles	This is an array of key value pair, on each pair, key should be name of styling and value is class name. This list will be shown on quick toolbar options to change the styles of table on designing like dashed, double bordered.	<code>List<DropDownItemModel></code>
Width	Sets the default width of the table.	100%

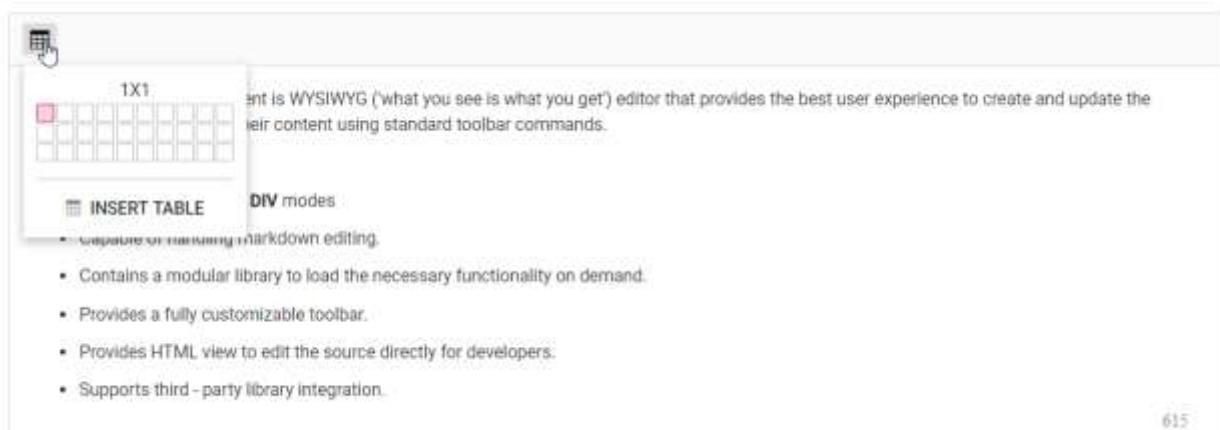
Insert table

Using the **CreateTable** toolbar option, select a number of rows and columns to be inserted over the table grid and insert table into Rich Text Editor content using the mouse. Tables can also be inserted through the **Insert Table** option in the pop-up where the number of rows and columns can be provided manually and this is the default way in devices.

In the following sample, the table has been inserted using **CreateTable** toolbar item.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor ShowCharCount="true">
  <RichTextEditorToolbarSettings Items="@Tools" />
  <p>The Rich Text Editor component is WYSIWYG ('what you see is what you get') editor that provides the best user experience to create and update the content. Users can format their content using standard toolbar commands.</p>
  <p><b>Key features:</b></p>
  <ul>
    <li><p>Provides <b>IFRAME</b> and <b>DIV</b> modes </p></li>
    <li><p>Capable of handling markdown editing.</p></li>
    <li><p>Contains a modular library to load the necessary functionality on demand.</p></li>
    <li><p>Provides a fully customizable toolbar.</p></li>
    <li><p>Provides HTML view to edit the source directly for developers.</p></li>
    <li><p>Supports third - party library integration.</p></li>
  </ul>
</SfRichTextEditor>
@code{
private List<ToolbarItemModel> Tools = new List<ToolbarItemModel>()
{
  new ToolbarItemModel() { Command = ToolbarCommand.CreateTable }
};
}
```



Quick Toolbar

Quick toolbar is opened by clicking the table. It has different sets of commands to be performed on the table which increases the feasibility to edit the table easily.

Table Header

Table Header command is available with quick toolbar option through which the header row can be added or removed from the inserted table. The following image illustrates the table header.

Insert Rows

Rows can be inserted above or below the required table cell through the quick toolbar. Also, focused row can be deleted. The following screenshot shows the available options of the row item.



Insert Columns

Columns can be inserted to the left or right side of the required table cell through the quick toolbar. Also, the focused column can be deleted. The following screenshot shows the available options of the column item.



Set Color

The Background Color can be set for each table cell through the **BackgroundColor** command available with quick toolbar.



Delete Table

Using the delete item in the quick toolbar, users can delete the entire table.

Vertical Align

Text inside the table can be aligned to top, middle, or bottom using the `TableCellVerticalAlign` command of the quick toolbar.



Horizontal Align

Text inside the table can be aligned left, right, or center using the `TableCellHorizontalAlign` command of the quick toolbar.

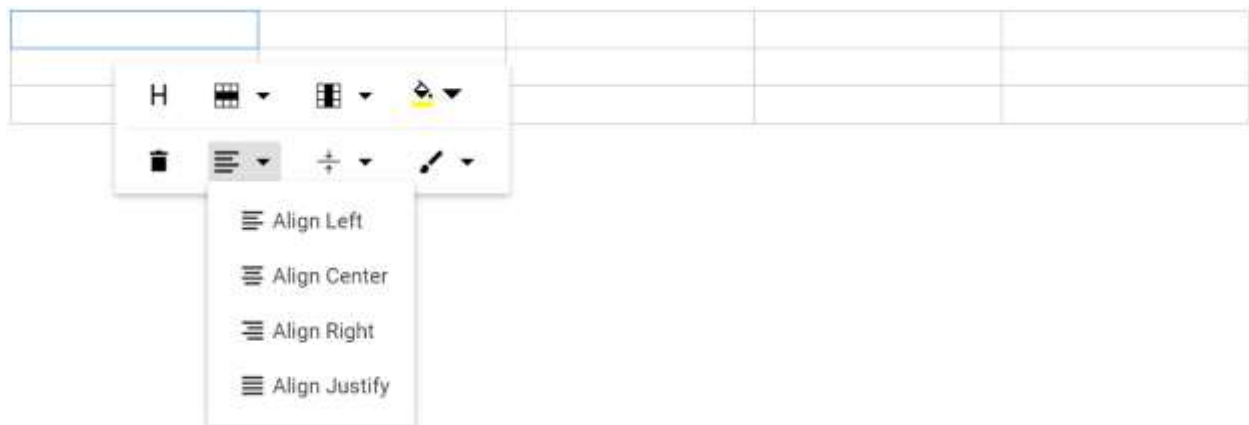


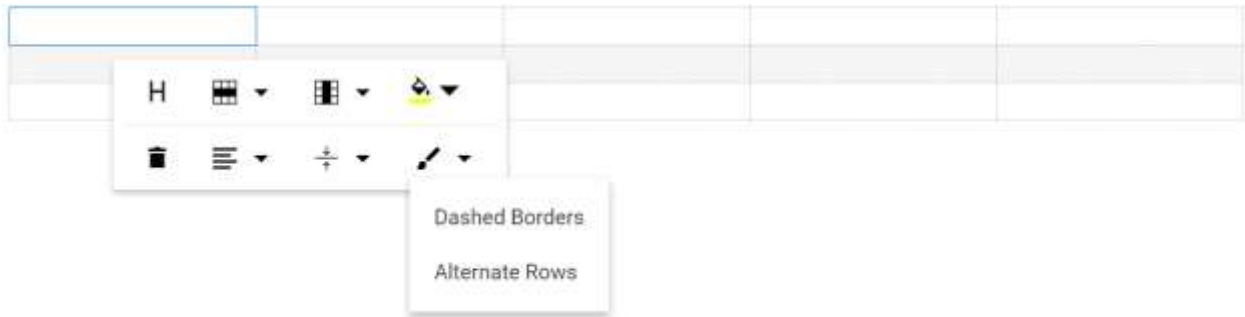
Table Styles

Table styles provided for class name should be appended to a table element. It helps to design the table in specific CSS styles when inserting in the editor.

By default, provides `Dashed border` and `Alternate rows`.

Dashed border: Applies the dashed border to the table.

Alternate border: Applies the alternative background to the table.



Custom Styles

Rich Text Editor provides support to custom styles for tables. If you want to add additional styles, pass the styles information as `List<DropDownItemModel>` data to the `Styles` field of `RichTextEditorTableSettings` tag.

ASPX-CS

```
<SfRichTextEditor>
  <RichTextEditorTableSettings Styles="@StyleItems" />
  <RichTextEditorToolbarSettings Items="@Tools" />
</SfRichTextEditor>
@code{
private List<ToolbarItemModel> Tools = new List<ToolbarItemModel>()
{
    new ToolbarItemModel() { Command = ToolbarCommand.CreateTable }
};
private List<DropDownItemModel> StyleItems = new List<DropDownItemModel>()
{
    new DropDownItemModel() { Text = "Alternate Rows" }
};
}
```



Table Properties

Sets the default width of the table when it is inserted in the Rich Text Editor using the width of `RichTextEditorTableSettings`.

Using the quick toolbar, users can change the width, cell padding, and cell spacing in the selected table using the `TableEditProperties` command dialog action.

Edit Table

×

Width

1,189

▼ ▲

Cell Padding

0

▼ ▲

Cell Spacing

0

▼ ▲

UPDATE

CANCEL

You can refer to our [Blazor Rich Text Editor](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor Rich Text Editor](#) example to know how to render and configure the rich text editor tools.

Table cell merge and split

The Rich Text Editor allows users to change the appearance of the tables by splitting or merging the table cells.

TableCell item should be configured in the Table [quickToolbarSettings](#) property to show the merge/split icons while selecting the table cells.

Table cell merge

The table cell merge feature allows to merge two or more row and column cells into a single cell with its contents.



Table cell split

The table cell split feature allows to a selected cell can be split both horizontally and vertically.



Markdown in Blazor RichTextEditor Component

In Rich Text Editor, click the toolbar buttons to format the words and the changes are visible immediately.

Markdown is not like that. When you format the word in Markdown format, you need to add Markdown syntax to the word to indicate which words and phrases should look different from each other. Rich Text Editor supports markdown editing when the `EditorMode` set as `Markdown` and using both *keyboard interaction* and *toolbar action*, you can apply the formatting to text.

Supported Commands

The [Blazor Markdown editor](#) supports the following commands to format the markdown content:

Commands	Syntax	Description

Bold	Sample content for bold text .	For bold, add or to front and back of the text. For order list, precede each line with a number.
Italic	Sample content for <i>Italic text</i> .	For Italic, add <i>* or _</i> to front and back of the text.
Bold and Italics	Sample content for <i>bold and Italic text</i> .	For bold and Italics, add * to the front and back of the text.
Heading 1	# Heading 1 content	For heading 1, add # to start of the line.
Heading 2	## Heading 2 content	For heading 2, add ## to start of the line.
Heading 3	### Heading 3 content	For heading 3, add ### to start of the line.
Heading 4	#### Heading 4 content	For heading 4, add #### to start of the line.
Heading 5	##### Heading 5 content	For heading 5, add ##### to start of the line.
Heading 6	##### Heading 6 content	For heading 6, add ##### to start of the line.
Line Break	First line
 Second line	For line break, press enter two times (or) add
 in between the first and the second line.
Blockquotes	> Blockquotes text	For blockquotes, add > to start of the line.
Strike Through	Sample content for strike through text .	For strike through, add ~~ to front and back of the text.
Code (Single line)	\Single line code\	For single line code, add ` to front and back of the text.

| Code block (Multi Line) | `\\
Multi line code text
Multi line code text
\\` | For multiple line code, add `\\` in the new line before and after the content. |

| Subscript | `_{Subscript text}` | For subscript, add `_{` to the front and `}` to the back of the text. |

| Superscript | `^{Superscript text}` | For superscript, add `^{` to the front and `}` to the back of the text. |

| Ordered List | `1. First
1. Second` | For ordered list, preceding one or more lines of text with `1.` |

| Unordered List | `First
second` | For unordered list, preceding one or more lines of text with `*`. |

| Links | **Link text without title text**`
` `Link text ` `
Link text with title text
[Link text](URL , "title text")` | Create an inline link by wrapping link text in brackets [], and then wrapping the URL as first parameter and title as second parameter in the parentheses ().
Note: The title text is optional, if needed it can be given manually. |

Table		Heading 1	Heading 2	` 	-----	-----
Col A1	Col A2					
Col B1	Col B2	`	Create a table using the pipes and underscores as given in the syntax to create 2 x 2 table.			

| Horizontal Line | ***(three asterisk in new line)
(or)
(three underscores in new line)*** | **For horizontal line, add or** to the start of the new line. |

| Image | `![alt text](URL path)` | Create an image by wrapping the image source in parentheses (). |

| Image with alternate text | `![alternate text](URL path)` | Create an image with alternate text by wrapping an alternative text in brackets [], and then link of the image source in parentheses ().
Note: When inserting the image using toolbar, the alternate text cannot be provided that needs to be given manually. |

| Escape tick marks supported | Sample text content with **bold and** not bold **text can be in the same line.** | In the syntax, the whole content is made as bold where the content not bold can be made as normal text by adding the bold syntax to the start and end of the respective text. Likewise you can do the same for various inline commands. |

| Escape Character | `\(any syntax)` | Escape any markdown syntax by prefix `\` to the syntax.
Example:`
\Bold text` |

| HTML Entities | Copyright: `©` - `©`; `
`Trade mark: `™` - `™`; `
`Registered: `®` - `®`; `
`Ampersand: `&` - `&`; `
`Less than: `<` - `<
`Greater than: `>` - `>>` | For HTML entities, add `&` and `;` to the front and back of the respective entities. |

The above listed commands alone are supported in Syncfusion Markdown editor. For other unsupported commands, you can achieve using the HTML tags in Markdown editor. The foot notes, definitions, math, and check list markdown syntax are also not supported.

Table

Rich Text Editor allows to insert Markdown table in edit panel with 2 X 2 rows and columns along with the heading. To use table tool, add the `CreateTable` item in toolbar items.

Insert table

To insert the table in Rich Text Editor, click the



Table toolbar option to insert the table into Rich Text Editor content and this is the default way in all the devices. Refer to the following sample and code snippets to add the table in Markdown editor.

ASPX-CS

```

@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor EditorMode="EditorMode.Markdown" Height="340px"
Placeholder="Type Something">
<RichTextEditorToolbarSettings Items="@Tools" />
In Rich Text Editor , you click the toolbar buttons to format the words and
the changes are visible immediately.
Markdown is not like that. When you format the word in Markdown format, you
need to add Markdown syntax to the word to indicate which words
and phrases should look different from each other.
Rich Text Editor supports markdown editing when the editorMode set as
**markdown** and using both *keyboard interaction* and *toolbar action*,
you can apply the formatting to text.
We can add our own custom formation syntax for the Markdown formation,
[sample link] (https://blazor.syncfusion.com/home/).
The third-party library <b>Marked</b> is used in this sample to convert
markdown into HTML content.
Markdown Table Format
|Heading 1|Heading 2|
|-----|-----|
|Col A1|Col A2|
|Col B1|Col B2|
</SfRichTextEditor>
@code {
private List<ToolbarItemModel> Tools = new List<ToolbarItemModel>()
{
new ToolbarItemModel() { Command = ToolbarCommand.CreateTable }
};
}

```

![Blazor RichTextEditor with Markdown Table](./images/blazor-richtexteditor-markdown-table.png)

Changing table constants

The Markdown table constants can be changed for the table heading and the column names.

You can also explore our [Blazor Markdown editor](#) example to know how to render and configure the rich text editor tools.

See Also

- [How to change the editor mode](#)

Form validation in Blazor RichTextEditor Component

This following sample demonstrates how to get the Rich Text Editor validation error message in button click.

Render the editor in a form

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
@using System.ComponentModel.DataAnnotations;
<div class="control-section">
<div class="content-container">
<div id="content" class="box-form" style="margin: 0 auto; max-width:750px;
padding:25px">
<EditForm Model="@Model">
<DataAnnotationsValidator />
<SfRichTextEditor ShowCharCount="true" MaxLength="100" Placeholder="Type
something..." @bind-Value="@Model.Description" />
<ValidationMessage For="@(() => Model.Description)" />
<div class="btn-grp">
<button class="samplebtn e-control e-btn" type="reset" data-
ripple="true">Reset</button>
<button class="samplebtn e-control e-btn" type="submit" data-
ripple="true">Submit</button>
</div>
</EditForm>
</div>
</div>
</div>
<style>
.box-form {
webkit-box-shadow: 0 2px 2px 0 rgba(0, 0, 0, 0.14), 0 3px 1px -2px rgba(0,
0, 0, 0.12), 0 1px 5px 0 rgba(0, 0, 0, 0.2);
box-shadow: 0 2px 2px 0 rgba(0, 0, 0, 0.14), 0 3px 1px -2px rgba(0, 0, 0,
0.12), 0 1px 5px 0 rgba(0, 0, 0, 0.2);
}
.btn-grp {
text-align: center;
margin-top: 15px;
}
.validation-message {
color: #f44336;
font-family: "Roboto", "Segoe UI", "GeezaPro", "DejaVu Serif", "sans-serif",
"-apple-system", "BlinkMacSystemFont";
font-size: 12px;
font-weight: normal;
}
```

```

}
</style>
@code{
private class FormModel
{
[Required]
[MinLength(20, ErrorMessage = "Please enter at least 20 characters.")]
public string Description { get; set; }
}
private FormModel Model = new FormModel();
}

```

Enter Text

The Description field is required.

SUBMIT

Validation Rules

The Rich Text Editor is a textarea control. The Rich Text Editor also provides the functionality of character count and its validation. So, you can validate the Rich Text Editor's value on form submission by applying Validation Rules and Validation Message to the Rich Text Editor.

| Rules | Description |

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

| Required | Requires value for the Rich Text Editor control. |

| MinLength | Requires the value to be of given minimum characters count. |

| MaxLength | Requires the value to be of given maximum characters count. |

This sample is demonstrated form validation using the `DataAnnotationsValidator`.

ASPX-CS

```

@using Syncfusion.Blazor.RichTextEditor
@using System.ComponentModel.DataAnnotations;
<EditForm Model="@MyForm">
<DataAnnotationsValidator />
<p>
<label for="description">Enter Text</label>
<SfRichTextEditor ShowCharCount="true" MaxLength="200" Placeholder="Type
something" @bind-Value="@MyForm.Description">
<RichTextEditorToolbarSettings Items="@Tools" />
</SfRichTextEditor>
<ValidationMessage For="@(( ) => MyForm.Description)"></ValidationMessage>
</p>
<button class="e-btn" type="submit">Submit</button>
</EditForm>

```



```
@code{
public class Form
{
    [Required]
    [MinLength(20, ErrorMessage = "Please enter at least 20 characters.")]
    public string Description { get; set; } = "<div>Rich Text Editor allows to
insert images from online source as well as local computer where you want to
insert the image in your content.</div>";
}
private Form MyForm = new Form();
private List<ToolbarItemModel> Tools = new List<ToolbarItemModel>()
{
    new ToolbarItemModel() { Command = ToolbarCommand.Bold },
    new ToolbarItemModel() { Command = ToolbarCommand.Italic },
    new ToolbarItemModel() { Command = ToolbarCommand.Underline },
    new ToolbarItemModel() { Command = ToolbarCommand.Separator },
    new ToolbarItemModel() { Command = ToolbarCommand.Formats },
    new ToolbarItemModel() { Command = ToolbarCommand.Alignments },
    new ToolbarItemModel() { Command = ToolbarCommand.Separator },
    new ToolbarItemModel() { Command = ToolbarCommand.CreateLink },
    new ToolbarItemModel() { Command = ToolbarCommand.Image },
    new ToolbarItemModel() { Command = ToolbarCommand.Separator },
    new ToolbarItemModel() { Command = ToolbarCommand.SourceCode },
    new ToolbarItemModel() { Command = ToolbarCommand.Separator },
    new ToolbarItemModel() { Command = ToolbarCommand.Undo },
    new ToolbarItemModel() { Command = ToolbarCommand.Redo }
};
}
```

Enter Text

Validation Message

The default error message for a rule can be customizable by defining it along with concern rule object as follows.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
@using System.ComponentModel.DataAnnotations;
<EditForm Model="@MyForm">
    <DataAnnotationsValidator />
    <p>
        <label for="description">Enter Text</label>
        <SfRichTextEditor @bind-Value="@MyForm.Description" ShowCharCount="true"
        MaxLength="200" Placeholder="Type something">
```

```

<RichTextEditorToolbarSettings items="@Tools" />
</SfRichTextEditor>
<ValidationMessage For="@(() => MyForm.Description)"></ValidationMessage>
</p>
<button class="e-btn" type="submit">Submit</button>
</EditForm>
@code{
public class Form
{
    [Required]
    [MinLength(10, ErrorMessage = "Please enter at least 10 characters.")]
    [MaxLength(100, ErrorMessage = "Maximum 200 characters only")]
    public string Description { get; set; } = "<div>Rich Text Editor allows to
insert images from online source as well as local computer where you want to
insert the image in your content.</div>";
}
private Form MyForm = new Form();
private List<ToolbarItemModel> Tools = new List<ToolbarItemModel>()
{
    new ToolbarItemModel() { Command = ToolbarCommand.Bold },
    new ToolbarItemModel() { Command = ToolbarCommand.Italic },
    new ToolbarItemModel() { Command = ToolbarCommand.Underline },
    new ToolbarItemModel() { Command = ToolbarCommand.Separator },
    new ToolbarItemModel() { Command = ToolbarCommand.Formats },
    new ToolbarItemModel() { Command = ToolbarCommand.Alignments },
    new ToolbarItemModel() { Command = ToolbarCommand.Separator },
    new ToolbarItemModel() { Command = ToolbarCommand.CreateLink },
    new ToolbarItemModel() { Command = ToolbarCommand.Image },
    new ToolbarItemModel() { Command = ToolbarCommand.Separator },
    new ToolbarItemModel() { Command = ToolbarCommand.SourceCode },
    new ToolbarItemModel() { Command = ToolbarCommand.Separator },
    new ToolbarItemModel() { Command = ToolbarCommand.Undo },
    new ToolbarItemModel() { Command = ToolbarCommand.Redo }
};
}

```

Custom Placement of Validation Message

The Form Validation error message can be placed from default position to desired custom location.

ASPX-CS

```

@using Syncfusion.Blazor.RichTextEditor
@using System.ComponentModel.DataAnnotations;
<EditForm Model="@MyForm">
<p>
<ValidationMessage For="@(() => MyForm.Description)"></ValidationMessage>
</p>
<DataAnnotationsValidator />
<p>
<label for="description">Enter Text</label>
<SfRichTextEditor ShowCharCount="true" MaxLength="200" Placeholder="Type
something" @bind-Value="@MyForm.Description">
<RichTextEditorToolbarSettings Items="@Tools" />
</SfRichTextEditor>
</p>
<button class="e-btn" type="submit">Submit</button>

```

```

</EditForm>
@code{
public class Form
{
    [Required(ErrorMessage = "RTE: value is required")]
    [MinLength(15, ErrorMessage = "RTE: Need atleast 15 character length")]
    [MaxLength(100, ErrorMessage = "RTE: Maximum 200 characters only")]
    public string Description { get; set; } = "<div>Rich Text Editor allows to
    insert images from online source as well as local computer where you want to
    insert the image in your content.</div>";
}
private Form MyForm = new Form();
private List<ToolbarItemModel> Tools = new List<ToolbarItemModel>()
{
    new ToolbarItemModel() { Command = ToolbarCommand.Bold },
    new ToolbarItemModel() { Command = ToolbarCommand.Italic },
    new ToolbarItemModel() { Command = ToolbarCommand.Underline },
    new ToolbarItemModel() { Command = ToolbarCommand.Separator },
    new ToolbarItemModel() { Command = ToolbarCommand.Formats },
    new ToolbarItemModel() { Command = ToolbarCommand.Alignments },
    new ToolbarItemModel() { Command = ToolbarCommand.Separator },
    new ToolbarItemModel() { Command = ToolbarCommand.CreateLink },
    new ToolbarItemModel() { Command = ToolbarCommand.Image },
    new ToolbarItemModel() { Command = ToolbarCommand.Separator },
    new ToolbarItemModel() { Command = ToolbarCommand.SourceCode },
    new ToolbarItemModel() { Command = ToolbarCommand.Separator },
    new ToolbarItemModel() { Command = ToolbarCommand.Undo },
    new ToolbarItemModel() { Command = ToolbarCommand.Redo }
};
}

```

RTE: Maximum 100 character only

Enter Text

You can refer to our [Blazor Rich Text Editor](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor Rich Text Editor](#) example to know how to render and configure the rich text editor tools.

Globalization in Blazor RichTextEditor Component

Localization

The Rich Text Editor provides an option to localize its strings; it is used for adapting the editor to a particular local language. By default, the editor will use the **US English (en-US)** as its language. Find the table with a list of keys and their corresponding values for the default language (en-US).

Use **Resource** file to translate the static text of the Rich Text Editor. The Resource file is an XML file which contains the strings(key and value pairs) that you want to translate into different language. You can also refer [Localization](#) link to know more about how to configure and use localization in the Blazor Server and WebAssembly project for Syncfusion Blazor components.

Locale key	en-US (default)
-----	-----
RichTextEditor_Alignments	Alignments
RichTextEditor_JustifyLeft	Align Left
RichTextEditor_JustifyCenter	Align Center
RichTextEditor_JustifyRight	Align Right
RichTextEditor_JustifyFull	Align Justify
RichTextEditor_FontName	Font Name
RichTextEditor_FontColor	Font Color
RichTextEditor_BackgroundColor	Background Color
RichTextEditor_Bold	Bold
RichTextEditor_Italic	Italic
RichTextEditor_Underline	Underline
RichTextEditor_Strikethrough	Strikethrough
RichTextEditor_ClearFormat	Clear Format
RichTextEditor_ClearAll	Clear All
RichTextEditor_Cut	Cut
RichTextEditor_Copy	Copy
RichTextEditor_Paste	Paste
RichTextEditor_UnorderedList	Bulleted List
RichTextEditor_OrderedList	Numbered List
RichTextEditor_Indent	Increase Indent
RichTextEditor_Outdent	Decrease Indent
RichTextEditor_Undo	Undo
RichTextEditor_Redo	Redo
RichTextEditor_Superscript	Superscript
RichTextEditor_Subscript	Subscript
RichTextEditor_CreateLink	Insert Link
RichTextEditor_OpenLink	Open Link
RichTextEditor_EditLink	Edit Link

| RichTextEditor_RemoveLink | Remove Link |

| RichTextEditor_Image | Insert Image |

| RichTextEditor_Replace | Replace |

| RichTextEditor_Align | Align |

| RichTextEditor_Caption | Image Caption |

| RichTextEditor_Remove | Remove |

| RichTextEditor_InsertLink| Insert Link |

| RichTextEditor_Display |Display |

| RichTextEditor_AltText |Alternative Text |

| RichTextEditor_Dimension | Change Size |

| RichTextEditor_Fullscreen | Maximize |

| RichTextEditor*Maximize* / *RichTextEditor*Maximize |

| RichTextEditor_Minimize | Minimize |

| RichTextEditor_LowerCase | Lower Case |

| RichTextEditor_UpperCase | Upper Case |

| RichTextEditor_Print| Print |

| RichTextEditor_Formats | Formats |

| RichTextEditor_Sourcecode | Code View |

| RichTextEditor_Preview | Preview |

| RichTextEditor_Viewside | ViewSide |

| RichTextEditor_InsertCode | Insert Code |

| RichTextEditor_LinkText | Display Text |

| RichTextEditor_LinkTooltipLabel | Title |

| RichTextEditor_LinkWebUrl | Web Address |

| RichTextEditor_LinkTitle| Enter a title |

| RichTextEditor_LinkUrl | http://example.com |

| RichTextEditor_LinkOpenInNewWindow |Open Link in New Window |

| RichTextEditor_LinkHeader | Insert Link |

| RichTextEditor_DialogInsert |Insert |

| RichTextEditor_DialogCancel | Cancel |

| RichTextEditor_DialogUpdate | Update |

| RichTextEditor_ImageHeader |Insert Image |

| RichTextEditor_ImageLinkHeader |You can also provide a link from the web |

| RichTextEditor_MdImageLink | Please provide a URL for your image |

| RichTextEditor_ImageUploadMessage | Drop image here or browse to upload |

| RichTextEditor_ImageDeviceUploadMessage | Click here to upload |

| RichTextEditor_ImageAlternateText | Alternate Text |

| RichTextEditor_Browse | Browse |

| RichTextEditor_ImageUrl | http://example.com/image.png |

| RichTextEditor_ImageCaption | Caption |

| RichTextEditor_ImageSizeHeader | Image Size |

| RichTextEditor_ImageHeight | Height |

| RichTextEditor_ImageWidth | Width |

| RichTextEditor_TextPlaceholder | Enter Text |

| RichTextEditor_InsertTableBtn | Insert Table |

| RichTextEditor_TableDialogHeader | Insert Table |

| RichTextEditor_TableWidth | Width |

| RichTextEditor_Cellpadding | Cell Padding |

| RichTextEditor_Cellspaceing | Cell Spacing |

| RichTextEditor_Columns | Number of columns |

| RichTextEditor_Rows | Number of rows |

| RichTextEditor_TableRows | Table Rows |

| RichTextEditor_TableColumns | Table Columns |

| RichTextEditor_TableCellHorizontalAlign | Table Cell Horizontal Align |

| RichTextEditor_TableCellVerticalAlign | Table Cell Vertical Align |

| RichTextEditor_CreateTable | Create Table |

| RichTextEditor_RemoveTable | Remove Table |

| RichTextEditor_TableHeader | Table Header |

| RichTextEditor_TableRemove | Table Remove |

| RichTextEditor_TableCellBackground | Table Cell Background |

| RichTextEditor_TableEditProperties | Table Edit Properties |

| RichTextEditor_Styles | Styles |

| RichTextEditor_InsertColumnLeft | Insert Column Left |

| RichTextEditor_InsertColumnRight | Insert Column Right |

| RichTextEditor_DeleteColumn | Delete Column |

| RichTextEditor_InsertRowBefore | Insert Row Before |

| RichTextEditor_InsertRowAfter | Insert Row After |

| RichTextEditor_DeleteRow | Delete Row |

| RichTextEditor_TableEditHeader | Edit Table |

| RichTextEditor_TableHeadingText |Heading |

| RichTextEditor_TableColText |Col |

| RichTextEditor_ImageInsertLinkHeader | Insert Link |

| RichTextEditor_EditImageHeader | Edit Image |

| RichTextEditor_AlignmentsDropDownLeft | Align Left |

| RichTextEditor_AlignmentsDropDownCenter |Align Center |

| RichTextEditor_AlignmentsDropDownRight | Align Right |

| RichTextEditor_AlignmentsDropDownJustify |Align Justify |

| RichTextEditor_ImageDisplayDropDownInline | Inline |

| RichTextEditor_ImageDisplayDropDownBreak | Break |

| RichTextEditor_TableInsertRowDropDownBefore | Insert row before |

| RichTextEditor_TableInsertRowDropDownAfter |Insert row after |

| RichTextEditor_TableInsertRowDropDownDelete | Delete row |

| RichTextEditor_TableInsertColumnDropDownLeft |Insert column left |

| RichTextEditor_TableInsertColumnDropDownRight | Insert column right |

| RichTextEditor_TableInsertColumnDropDownDelete | Delete column |

| RichTextEditor_TableVerticalAlignDropDownTop | Align Top |

| RichTextEditor_TableVerticalAlignDropDownMiddle | Align Middle |

| RichTextEditor_TableVerticalAlignDropDownBottom | Align Bottom |

| RichTextEditor_TableStylesDropDownDashedBorder | Dashed Borders |

| RichTextEditor_TableStylesDropDownAlternateRows | Alternate Rows |

| RichTextEditor_PasteFormat |Paste Format |

| RichTextEditor_PasteFormatContent | Choose the formatting action |

| RichTextEditor_PlainText | Plain Text |

| RichTextEditor_CleanFormat |Clean |

| RichTextEditor_KeepFormat | Keep |

| RichTextEditor_PasteDialogOk | OK |

| RichTextEditor_PasteDialogCancel | Cancel |

| RichTextEditor_FormatsDropDownParagraph | Paragraph |

| RichTextEditor_FormatsDropDownCode | Code |

| RichTextEditor_FormatsDropDownQuotation | Quotation |

| RichTextEditor_FormatsDropDownHeading1 | Heading 1 |

| RichTextEditor_FormatsDropDownHeading2 | Heading 2 |

| RichTextEditor_FormatsDropDownHeading3 | Heading 3 |

| RichTextEditor_FormatsDropDownHeading4 | Heading 4 |

| RichTextEditor_FormatsDropDownHeading5 | Heading 5 |

| RichTextEditor_FormatsDropDownHeading6 | Heading 6 |

| RichTextEditor_FontNameSegoeUI | Segoe UI |

| RichTextEditor_FontNameArial | Arial |

| RichTextEditor_FontNameGeorgia | Georgia |

| RichTextEditor_FontNameImpact | Impact |

| RichTextEditor_FontNameTahoma | Tahoma |

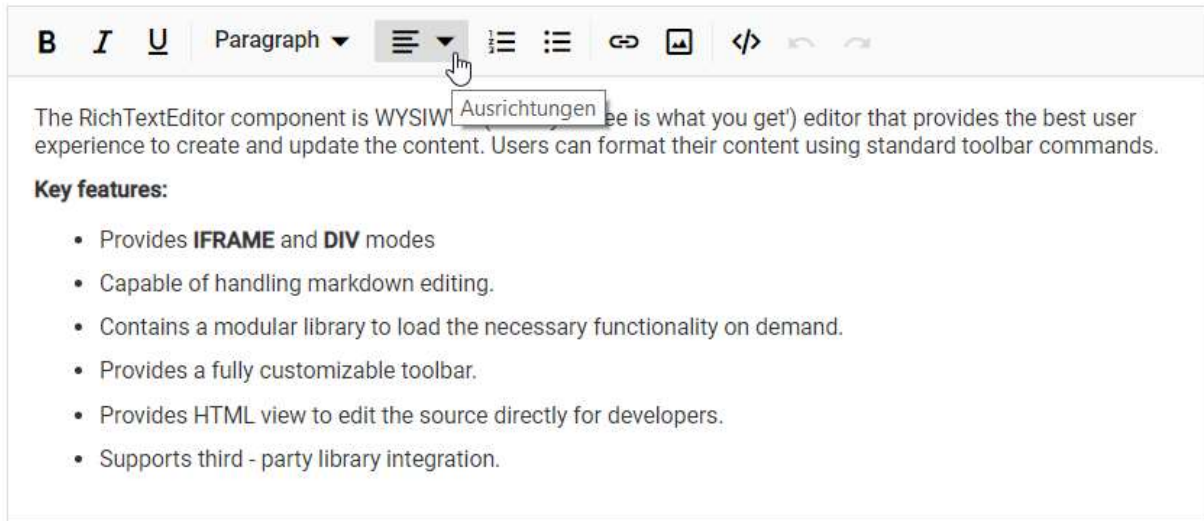
| RichTextEditor_FontNameTimesNewRoman | Times New Roman |

| RichTextEditor_FontNameVerdana | Verdana |

The following sample code block demonstrates that the Rich Text Editor control rendered with **de-DE** German language.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<p>The Rich Text Editor component is WYSIWYG ('what you see is what you get') editor that provides the best user experience to create and update the content. Users can format their content using standard toolbar commands.</p>
<p><b>Key features:</b></p>
<ul>
<li><p>Provides <b>IFRAME</b> and <b>DIV</b> modes </p></li>
<li><p>Capable of handling markdown editing.</p></li>
<li><p>Contains a modular library to load the necessary functionality on demand.</p></li>
<li><p>Provides a fully customizable toolbar.</p></li>
<li><p>Provides HTML view to edit the source directly for developers.</p></li>
<li><p>Supports third - party library integration.</p></li>
</ul>
</SfRichTextEditor>
```

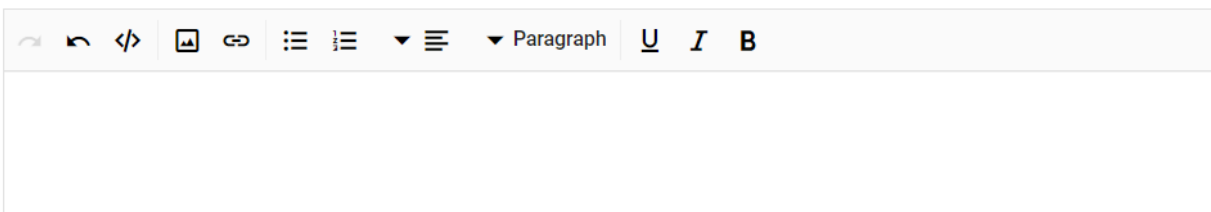
RTL

Specifies the direction of the Rich Text Editor component using the `EnableRtl` property. For writing systems will require Arabic, Hebrew, and more. The direction can be switched to right-to-left.

`EnableRtl` property will not change, based on current culture.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor EnableRtl="true" />
```



You can refer to our [Blazor Rich Text Editor](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor Rich Text Editor](#) example to know how to render and configure the rich text editor tools.

ExecCommand in Rich Text Editor in Blazor RichTextEditor Component

In Rich Text Editor, the `ExecuteCommand` is used to perform commands for the modification of content in editable area.

The `ExecuteCommand` will perform the following commands.

Commands	Description	Code snippets
-----	-----	-----
Bold	Bold the selected content in the Rich Text Editor.	<code>await this.RteObj.ExecuteCommandAsync(CommandName.Bold);</code>

| Italic | The selected text will be italics. | await
this.RteObj.ExecuteCommandAsync(CommandName.Italic);|

| Underline | Underline the selected text in the Rich Text Editor. | await
this.RteObj.ExecuteCommandAsync(CommandName.Underline);|

| StrikeThrough | Apply single line strike through formatting for the selected text. | await
this.RteObj.ExecuteCommandAsync(CommandName.StrikeThrough);|

| Superscript | Makes the selected text as superscript (higher). | await
this.RteObj.ExecuteCommandAsync(CommandName.Superscript);|

| Subscript | Makes the selected text as subscript (lower). | await
this.RteObj.ExecuteCommandAsync(CommandName.Subscript);|

| Uppercase | Change the case of selected text to upper in the content. | await
this.RteObj.ExecuteCommandAsync(CommandName.Uppercase);|

| Lowercase | Change the case of selected text to lower in the content. | await
this.RteObj.ExecuteCommandAsync(CommandName.Lowercase);|

| FontColor | Apply the specified font color for the selected text. | await
this.RteObj.ExecuteCommandAsync(CommandName.FontColor, "Red");|

| FontName | Apply the specified font name for the selected text. | await
this.RteObj.ExecuteCommandAsync(CommandName.FontName, "Impact");|

| FontSize | Apply the specified font size for the selected text. | await
this.RteObj.ExecuteCommandAsync(CommandName.FontSize, "10pt");|

| BackgroundColor | Apply the specified background color the selected text. | await
this.RteObj.ExecuteCommandAsync(CommandName.BackgroundColor, "red");|

| JustifyCenter | Align the content with center margin. | await
this.RteObj.ExecuteCommandAsync(CommandName.JustifyCenter);|

| JustifyFull | Align the content with justify margin. | await
this.RteObj.ExecuteCommandAsync(CommandName.JustifyFull);|

| JustifyLeft | Align the content with left margin. | await
this.RteObj.ExecuteCommandAsync(CommandName.JustifyLeft);|

| JustifyRight | Align the content with right margin. | await
this.RteObj.ExecuteCommandAsync(CommandName.JustifyRight);|

| CreateLink | Creates a hyperlink to a text or image to a specific location in the content. | await
this.RteObj.ExecuteCommandAsync(CommandName.CreateLink, new LinkCommandsArgs() {
Text = "Links", Url= "http://", Title = "Link"});|

| Indent | Allows to increase the indent level of the content. | await
this.RteObj.ExecuteCommandAsync(CommandName.Indent);|

| InsertHTML | Insert the html content to the current cursor position. | await
 this.RteObj.ExecuteCommandAsync(CommandName.InsertHTML,"<div>Syncfusion Rich Text Editor|

| InsertOrderedList | Create a new list item(numbered). | await
 this.RteObj.ExecuteCommandAsync(CommandName.InsertOrderedList);|

| InsertUnorderedList | Create a new list item(bulleted). | await
 this.RteObj.ExecuteCommandAsync(CommandName.InsertUnorderedList);|

| Outdent | Allows to decrease the indent level of the content. | await
 this.RteObj.ExecuteCommandAsync(CommandName.Outdent);|

| Redo | Allows to redo the actions | await
 this.RteObj.ExecuteCommandAsync(CommandName.Redo);|

| RemoveFormat | Remove all formatting styles (such as bold, italic, underline, color, superscript, subscript, and more) from currently selected text. | await
 this.RteObj.ExecuteCommandAsync(CommandName.RemoveFormat);|

| InsertText | Insert text to the current cursor position. | await
 this.RteObj.ExecuteCommandAsync(CommandName.InsertText, "Inserted text");|

| InsertImage | Insert an image to the current cursor position. | await
 this.RteObj.ExecuteCommandAsync(CommandName.InsertImage, new ImageCommandsArgs()
 { Url = "https://ej2.syncfusion.com/javascript/demos/src/rich-text-editor/images/RTEImage-
 Feather.png", CssClass = "rte-img" });|

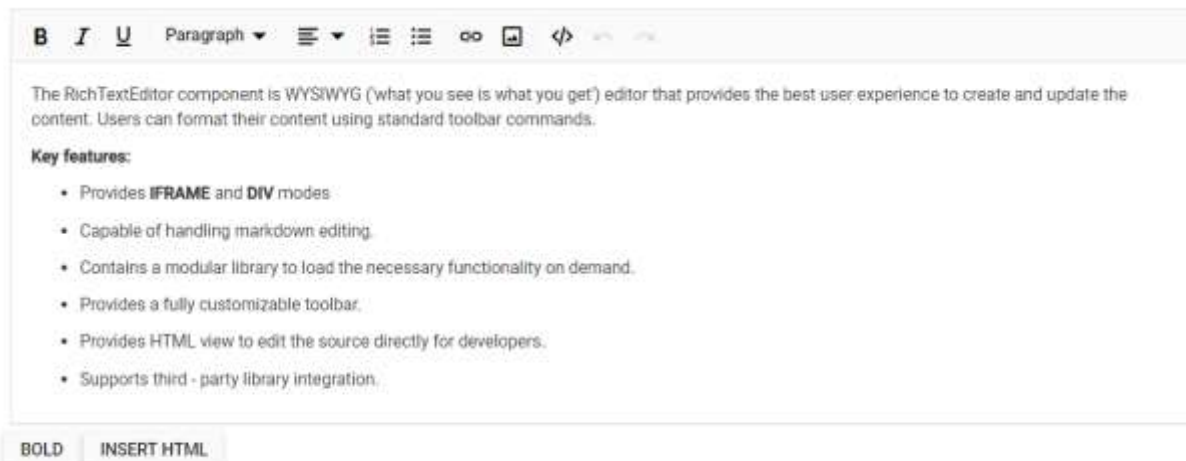
Provided support to apply execute commands which do not require direct DOM access.

The following code block demonstrates the usage of the ExecuteCommand in Rich Text Editor.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor @ref="@RteObj">
<p>The Rich Text Editor component is WYSIWYG ('what you see is what you
get') editor that provides the best user experience to create and update the
content. Users can format their content using standard toolbar commands.</p>
<p><b>Key features:</b></p>
<ul>
<li><p>Provides <b>IFRAME</b> and <b>DIV</b> modes </p></li>
<li><p>Capable of handling markdown editing.</p></li>
<li><p>Contains a modular library to load the necessary functionality on
demand.</p></li>
<li><p>Provides a fully customizable toolbar.</p></li>
<li><p>Provides HTML view to edit the source directly for
developers.</p></li>
<li><p>Supports third - party library integration.</p></li>
</ul>
</SfRichTextEditor>
<SfButton Content="Bold" @onclick="@OnBoldCommand" />
<SfButton Content="InsertHTML" @onclick="@OnInsertHtmlCommand" />
@code {
```

```
SfRichTextEditor RteObj;
private async Task OnBoldCommand()
{
    await this.RteObj.ExecuteCommandAsync(CommandName.Bold);
}
private async Task OnInsertHtmlCommand()
{
    await this.RteObj.ExecuteCommandAsync(CommandName.InsertHTML,
    "<div>Syncfusion Rich Text Editor component</div>");
}
}
```



You can refer to our [Blazor Rich Text Editor](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor Rich Text Editor](#) example to know how to render and configure the rich text editor tools.

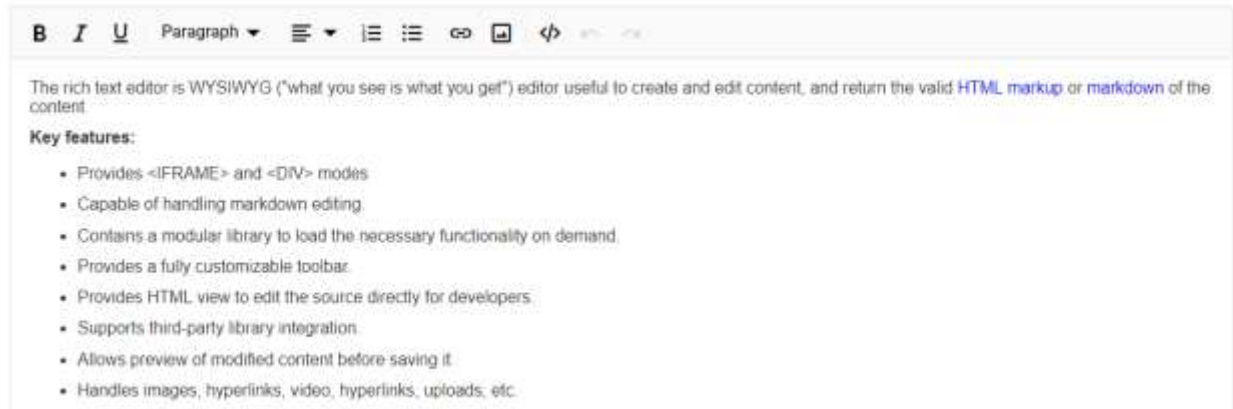
IFrame Rendering in Blazor RichTextEditor Component

When the `RichTextEditorIFrameSettings` option is enabled, the Rich Text Editor creates the `iframe` element as the content area on component initialization, it is used to display and edit the content. In content area, the editor displays only the body tag of a `<iframe>` document.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorIFrameSettings Enable="true" />
<p>The Rich Text Editor component is WYSIWYG ('what you see is what you get') editor that provides the best user experience to create and update the content. Users can format their content using standard toolbar commands.</p>
<p><b>Key features:</b></p>
<ul>
<li><p>Provides <b>IFRAME</b> and <b>DIV</b> modes </p></li>
<li><p>Capable of handling markdown editing.</p></li>
<li><p>Contains a modular library to load the necessary functionality on demand.</p></li>
<li><p>Provides a fully customizable toolbar.</p></li>
<li><p>Provides HTML view to edit the source directly for developers.</p></li>
</ul>
```

```
<li><p> Supports third - party library integration.</p></li>
</ul>
</SfRichTextEditor>
```



IFrame attributes

The editor allows to pass an additional attribute to body tag of a <iframe> element using **Attributes** fields of **RichTextEditorIframeSettings** property. This property contains name or value pairs in string format. It is used to override the default appearance of the content area.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorIframeSettings Enable="true" Attributes="@IframeAttributes"
/>
<p>The Rich Text Editor component is WYSIWYG ('what you see is what you
get') editor that provides the best user experience to create and update the
content. Users can format their content using standard toolbar commands.</p>
<p><b>Key features:</b></p>
<ul>
<li><p> Provides <b>IFRAME</b> and <b>DIV</b> modes </p></li>
<li><p> Capable of handling markdown editing.</p></li>
<li><p> Contains a modular library to load the necessary functionality on
demand.</p></li>
<li><p> Provides a fully customizable toolbar.</p></li>
<li><p> Provides HTML view to edit the source directly for
developers.</p></li>
<li><p> Supports third - party library integration.</p></li>
</ul>
</SfRichTextEditor>
@code {
private Dictionary<string, object> IframeAttributes = new Dictionary<string,
object>() {
{ "style", "background: lightgray;" }
};
}
```

Adding external CSS/Script File

The editor offers to add external CSS file to style the `<iframe>` element. Easily change the appearance of editor's content using an external CSS file using `Resources - Styles` field in the `RichTextEditorIframeSettings` property.

Likewise, add the external script file to the `<iframe>` element using `Resources - Scripts` field of `RichTextEditorIframeSettings` to provide the additional functionalities to the Rich Text Editor.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorIframeSettings Enable="true" Resources="@Resources" />
<p>Rich Text Editor allows to insert images from online source as well as
local computer where you want to insert the image in your content.</p>
<p><b>Get started Quick Toolbar to click on the image</b></p>
<p>It is possible to add custom style on the selected image inside the Rich
Text Editor through quick toolbar.</p>
<img alt='Logo' style='width: 300px; height: 300px; transform:
rotate(0deg);'
src='https://blazor.syncfusion.com/demos/images/RichTextEditor/RTEImage-
Feather.png' />
</SfRichTextEditor>
@code {
private ResourcesModel Resources { get; set; } = new ResourcesModel()
{
Styles = new string[] { "/styles.css" },
Scripts = new string[] { "/script.js" }
};
}
```

You can refer to our [Blazor Rich Text Editor](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor Rich Text Editor](#) example to know how to render and configure the rich text editor tools.

See Also

- [How to change the editor mode](#)

Style and appearance in Blazor RichTextEditor 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 Rich Text Editor's content

Use the following CSS to customize the default Rich Text Editor's content properties like font-family, font-size and color.

CSS

```
/* To change font family and font size */
.e-richtexteditor .e-rte-content .e-content,
.e-richtexteditor .e-source-content .e-content {
font-size: 20px;
font-family: Segoe ui;
```

```
}
/* To change font color and content background */
.e-richtexteditor .e-rte-content,
.e-richtexteditor .e-source-content {
background: seashell;
color: blue;
}
```

Customizing the Rich Text Editor's toolbar

Use the following CSS to customize the default color in the Rich Text Editor's toolbar icon.

CSS

```
/* To change font color for toolbar icon */
.e-richtexteditor .e-rte-toolbar .e-toolbar-item .e-icons,
.e-richtexteditor .e-rte-toolbar .e-toolbar-item .e-icons:active {
color: red;
}
/* To change font color for toolbar button */
.e-toolbar .e-tbar-btn,
.e-toolbar .e-tbar-btn:active,
.e-toolbar .e-tbar-btn:hover {
color: red;
}
/* To change font color for toolbar button in active state*/
.e-richtexteditor .e-rte-toolbar .e-toolbar-item .e-dropdown-btn.e-active
.e-icons,
.e-richtexteditor .e-rte-toolbar .e-toolbar-item .e-dropdown-btn.e-active
.e-rte-dropdown-btn-text {
color: red;
}
/* To change font color for expanded toolbar items */
.e-richtexteditor .e-rte-toolbar .e-toolbar-extended .e-toolbar-item .e-
tbar-btn .e-icons,
.e-toolbar.e-extended-toolbar .e-toolbar-extended .e-toolbar-item .e-tbar-
btn {
color: red;
}
```

Customizing the Rich Text Editor's character count

Use the following CSS to customize the default color in the Rich Text Editor's character count.

CSS

```
/* To change font color, font family, font size and opacity */
.e-richtexteditor .e-rte-character-count {
color: red;
font-family: segoe ui;
font-size: 18px;
opacity: 0.54;
padding-bottom: 2px;
padding-right: 14px;
}
```

You can refer to our [Blazor Rich Text Editor](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor Rich Text Editor](#) example to know how to render and configure the rich text editor tools.

Keyboard support in Blazor RichTextEditor Component

The editor has full keyboard accessibility that includes shortcuts to open and other actions with toolbar items, drop-down lists, and dialogs.

HTML formation shortcut key

You can use the following key shortcuts when the Rich Text Editor renders with **HTML** editMode.

| Actions | Keyboard shortcuts |

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

| Toolbar focus | ALT + F10 |

| Insert link | CTRL + K |

| Insert image | CTRL + SHIFT + I |

| Insert table | CTRL + SHIFT + E |

| Undo | CTRL + Z |

| Redo | CTRL + Y |

| Copy | CTRL + C |

| Cut | CTRL + X |

| Paste | CTRL + V |

| Bold | CTRL + B |

| Italic | CTRL + I |

| Underline | CTRL + U |

| Strikethrough | CTRL + SHIFT + S |

| Uppercase | CTRL + SHIFT + U |

| Lowercase | CTRL + SHIFT + L |

| Superscript | CTRL + SHIFT + = |

| Subscript | CTRL + = |

| Indents | CTRL +] |

| Outdents | CTRL + [|

| HTML source | CTRL + SHIFT + H |

| Full screen | CTRL + SHIFT + F |

| Exit Full screen | Esc |

| Justify center | CTRL + E |

| Justify full | CTRL + `J` |

| Justify left | CTRL + L |

| Justify right | CTRL + R |

| Clear format | CTRL + SHIFT + R |

| Ordered list | CTRL + SHIFT + O |

| Unordered list | CTRL + ALT + O |

Markdown formation shortcut key

You can use the following key shortcuts when the Rich Text Editor renders with **Markdown** editMode.

| Actions | Keyboard shortcuts |

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

| Toolbar focus | ALT + F10 |

| Insert link | CTRL + K |

| Insert image | CTRL + SHIFT + I |

| Insert table | CTRL + SHIFT + E |

| Undo | CTRL + Z |

| Redo | CTRL + Y |

| Copy | CTRL + C |

| Cut | CTRL + X |

| Paste | CTRL + V |

| Bold | CTRL + B |

| Italic | CTRL + i |

| Strikethrough | CTRL + SHIFT + S |

| Uppercase | CTRL + SHIFT + U |

| Lowercase | CTRL + SHIFT + L |

| Superscript | CTRL + SHIFT + = |

| Subscript | CTRL + = |

| Full screen | CTRL + SHIFT + F |

| Exit Full screen | Esc |

| Ordered list | CTRL + SHIFT + O |

| Unordered list | CTRL + ALT + O |

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
```

```
<SfRichTextEditor EditorMode="EditorMode.Markdown">
<p>The Rich Text Editor component is WYSIWYG ('what you see is what you
get') editor that provides the best user experience to create and update the
content. Users can format their content using standard toolbar commands.</p>
<p><b>Key features:</b></p>
<ul>
<li><p> Provides <b>IFRAME</b> and <b>DIV</b> modes </p></li>
<li><p> Capable of handling markdown editing.</p></li>
<li><p> Contains a modular library to load the necessary functionality on
demand.</p></li>
<li><p> Provides a fully customizable toolbar.</p></li><li><p> Provides HTML
view to edit the source directly for developers.</p></li>
<li><p> Supports third - party library integration.</p></li>
<li><p> Allows preview of modified content before saving it.</p></li>
</ul>
</SfRichTextEditor>
```



Custom key config

Customize the key config for the keyboard interaction of Rich Text Editor, using the `KeyConfigure` property.

In the following code block, customize the bold and italic, toolbar actions with `ctrl+1`, `ctrl+2` respectively.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor KeyConfigure="@KeyConfig">
<p>The Rich Text Editor component is WYSIWYG ('what you see is what you
get') editor that provides the best user experience to create and update the
content. Users can format their content using standard toolbar commands.</p>
<p><b>Key features:</b></p>
<ul>
<li><p> Provides <b>IFRAME</b> and <b>DIV</b> modes </p></li>
<li><p> Capable of handling markdown editing.</p></li>
<li><p> Contains a modular library to load the necessary functionality on
demand.</p></li>
<li><p> Provides a fully customizable toolbar.</p></li>
<li><p> Provides HTML view to edit the source directly for
developers.</p></li>
<li><p> Supports third - party library integration.</p></li>
<li><p> Allows preview of modified content before saving it.</p></li>
</ul>
</SfRichTextEditor>
```

```
@code {
private ShortcutKeys KeyConfig = new ShortcutKeys()
{
    Bold = "ctrl+1",
    Italic = "ctrl+2"
};
}
```

You can refer to our [Blazor Rich Text Editor](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor Rich Text Editor](#) example to know how to render and configure the rich text editor tools.

See Also

- [Globalization](#)
- [Accessibility](#)

Miscellaneous in Blazor RichTextEditor Component

Placeholder

Specifies the placeholder for the Rich Text Editor's content used when the Rich Text Editor body is empty through the `Placeholder` property.

Use the `e-rte-placeholder` class to define the custom font family, font color, and styles to the placeholder text.

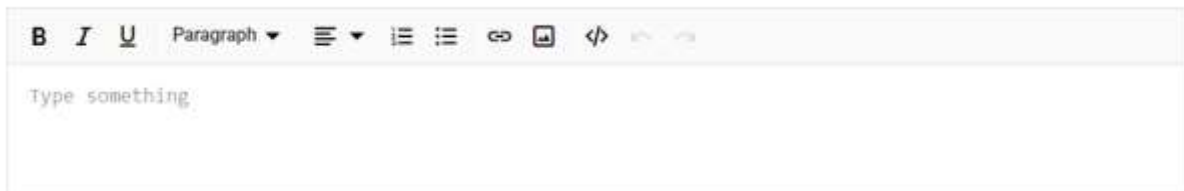
CSS

```
.e-richtexteditor .e-rte-placeholder {
font-family: monospace;
}
```

The following sample demonstrates the placeholder option in Rich Text Editor.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor Placeholder="Type something" />
<style>
.e-richtexteditor .e-rte-placeholder {
font-family: monospace;
}
</style>
```



Character count

The Rich Text Editor automatically counts the number of characters in the content while typing, using the `ShowCharCount` property. The characters count displayed at the bottom of the editor. You can limit the number of characters in your content using the `MaxLength` property. By default, the editor sets the characters limit value to infinity.

The character count color will be modified based on the characters in the Rich Text Editor.

| Status | Description |

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

| Normal | Till 70% of given `maxLength` count reach, character count color is black. |

| Warning | Once the number of character count in the Rich Text Editor reached 70% of given `maxLength` count, the character count color will be orange, indicating that, the Rich Text Editor value going to reach the maximum count. |

| Error | Once the number of character count in the Rich Text Editor reached 90% of given `maxLength` count, the character count color will be red, indicating that, the Rich Text Editor value reached the maximum count. |

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor ShowCharCount="true" MaxLength="500">
<p>The Rich Text Editor component is WYSIWYG ('what you see is what you get') editor that provides the best user experience to create and update the content. Users can format their content using standard toolbar commands.</p>
<p><b>Key features:</b></p>
<ul>
<li><p>Provides <b>IFRAME</b> and <b>DIV</b> modes </p></li>
<li><p>Capable of handling markdown editing.</p></li>
</ul>
</SfRichTextEditor>
```



Code view

Rich Text Editor includes the ability for users to directly edit HTML code via `Source View` in the text area. If you made any modification in Source view directly, the changes will be reflected in the Rich Text Editor's content. So, the users will have more flexibility over the content they have created.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
```

```

<p>The Rich Text Editor component is WYSIWYG ('what you see is what you get') editor that provides the best user experience to create and update the content. Users can format their content using standard toolbar commands.</p>
<p><b>Key features:</b></p>
<ul>
<li><p> Provides <b>IFRAME</b> and <b>DIV</b> modes </p></li>
<li><p> Capable of handling markdown editing.</p></li>
</ul>
</SfRichTextEditor>

```



Undo/Redo Manager

Undo and redo tools allows to edit the text by disregard or cancel the recently made changes and restore it to previous state. It is a useful tool to restore the performed action which got changed by mistake. By default, upto 30 actions can be undo/redo in the editor.

To undo and redo operations, do one of the following:

- Press the undo/redo button on the toolbar
- Press the Ctrl + Z/Ctrl + Y combination on the keyboard

Customize the undo/redo step count using `UndoRedoSteps` property. By default, undo/redo actions take 300ms time interval for storing the action to the undo redo manager. The time interval can be customized by using the `UndoRedoTimer`.

ASPX-CS

```

@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor UndoRedoSteps="50" UndoRedoTimer="400">
<p>The Rich Text Editor component is WYSIWYG ('what you see is what you get') editor that provides the best user experience to create and update the content. Users can format their content using standard toolbar commands.</p>
<p><b>Key features:</b></p>
<ul>
<li><p> Provides <b>IFRAME</b> and <b>DIV</b> modes </p></li>
<li><p> Capable of handling markdown editing.</p></li>
</ul>
</SfRichTextEditor>

```



Resizable support

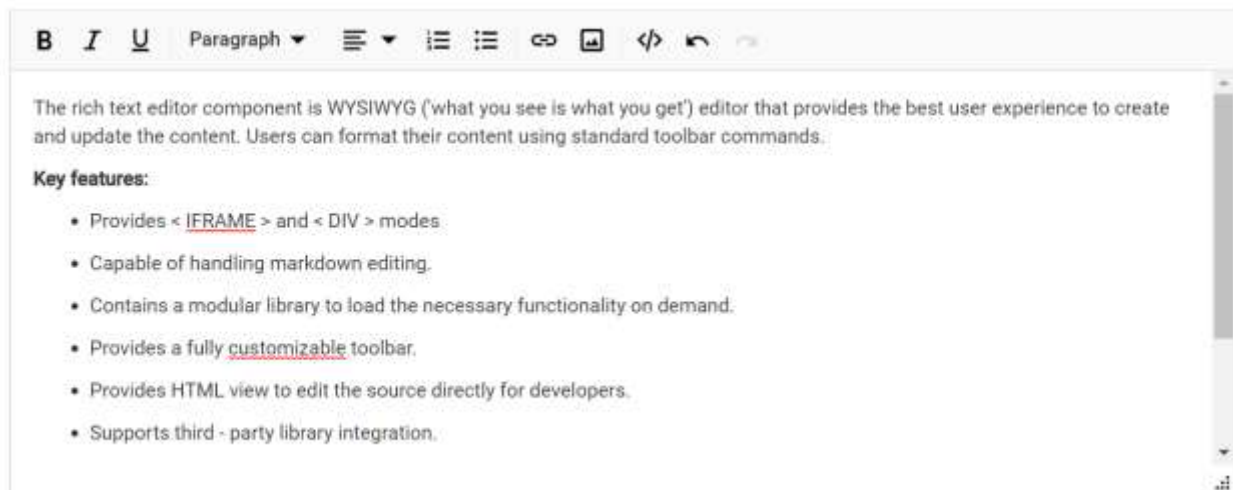
This feature allows the editor to be resized dynamically. The users can enable or disable this feature using the `EnableResize` property in the Rich Text Editor. If `EnableResize` is set to true, the Rich Text Editor component creates grip at the bottom right corner, which allows resizing the component in the diagonal direction. The following sample demonstrates the resizable feature.

Enabling the resizable support

To render the Rich Text Editor in the resizable mode, set the `EnableResize` property to true.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor EnableResize="true">
<p>Rich Text Editor allows to insert images from online source as well as
local computer where you want to insert the image in your
content.</p><p><b>Get started Quick Toolbar to click on the
image</b></p><p>It is possible to add custom style on the selected image
inside the Rich Text Editor through quick toolbar.</p><img alt='Logo'
style='width: 300px; height: 300px; transform: rotate(0deg);'
src='images/RichTextEditor/RTEImage-Feather.png' />
</SfRichTextEditor>
```



Specifying the Minimum and Maximum width and height for Resize

To have a restricted resizable area for the Rich Text Editor, you need to specify the min-width, max-width, min-height and max-height CSS properties for the control's container element. By default, the

control is capable of resizing upto the current viewport. The `e-richtexteditor` CSS class will be available in the component's container and can be used for applying the above mentioned styles.

CSS

```
<style>
.e-richtexteditor {
min-width: 200px;
max-width: 800px;
min-height: 100px;
max-height: 300px;
}
</style>
```

You can refer to our [Blazor Rich Text Editor](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor Rich Text Editor](#) example to know how to render and configure the rich text editor tools.

Number and Bullet Format Lists

This feature allows the user to change the appearance of the Numbered and Bulleted lists. Users can also apply different numbering or bullet formats lists such as lowercase greek, upper Alpha, square and circles. You can also customize the style type of the lists to be populated in the dropdown from the toolbar by using the `NumberFormatList` and `BulletFormatList` properties in the Rich Text Editor.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorToolbarSettings Items="@Tools" />
<p>The Rich Text Editor component is WYSIWYG ('what you see is what you
get') editor that provides the best user experience to create and update the
content. Users can format their content using standard toolbar commands.</p>
<p><b>Key features:</b></p>
<ul>
<li><p> Provides <b>IFRAME</b> and <b>DIV</b> modes </p></li>
<li><p> Capable of handling markdown editing.</p></li>
</ul>
</SfRichTextEditor>
@code {
private List<ToolbarItemModel> Tools = new List<ToolbarItemModel>()
{
new ToolbarItemModel() { Command = ToolbarCommand.NumberFormatList },
new ToolbarItemModel() { Command = ToolbarCommand.BulletFormatList },
};
}
```

Accessibility in Blazor RichTextEditor Component

The Rich Text Editor component has been designed, keeping in mind the WAI-ARIA specifications, and applies the WAI-ARIA roles, states, and properties. This component is characterized by complete ARIA accessibility support that makes it easy for people who use assistive technologies (AT) or those who completely rely on keyboard navigation.

ARIA attributes

The toolbar of Rich Text Editor, assigned the role of Toolbar and has the following list of ARIA attributes:

| Roles and Attributes | Functionalities |

| --- | --- |

| role="toolbar" | This attribute added to the toolbar element describes the actual role of the element. |

| aria-orientation | Indicates the toolbar orientation. Default value is horizontal. |

| aria-haspopup | Indicates the popup mode of the toolbar. The default value is false. When popup mode is enabled, attribute value has to be changed to true. |

| aria-disabled | Indicates the disabled state of the toolbar. |

For further details of toolbar ARIA attributes, refer to the accessibility of [Toolbar](#) documentation.

The Rich Text Editor element is assigned the role of application.

| Roles and Attributes | Functionalities |

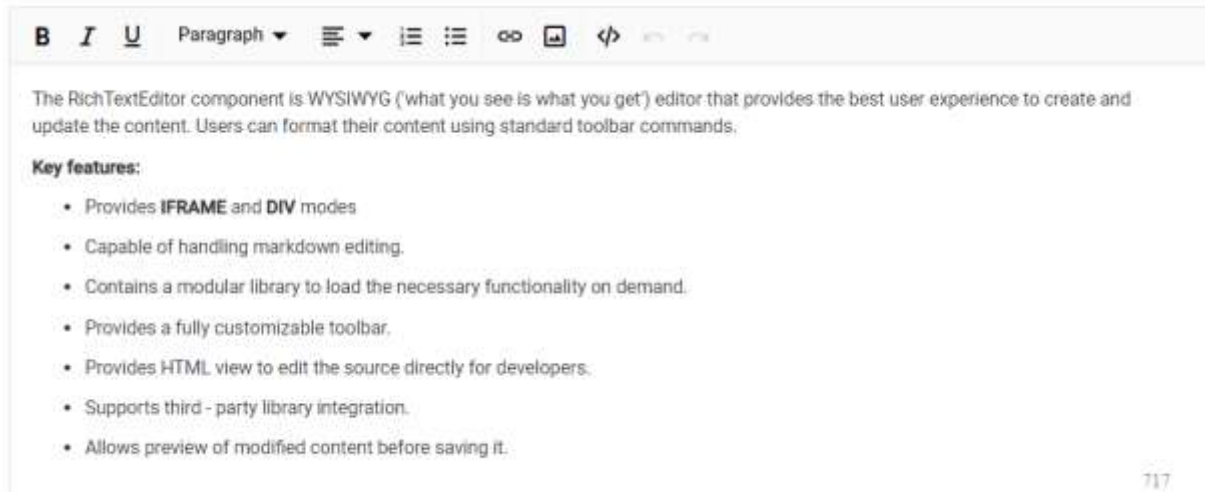
| --- | --- |

| role="application" | This attribute added to the Rich Text Editor element describes the actual role of the element. |

| aria-disabled | Indicates the disabled state of the Rich Text Editor. |

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor ShowCharCount="true">
<p>The Rich Text Editor component is WYSIWYG ('what you see is what you get') editor that provides the best user experience to create and update the content. Users can format their content using standard toolbar commands.</p>
<p><b>Key features:</b></p>
<ul>
<li><p>Provides <b>IFRAME</b> and <b>DIV</b> modes </p></li>
<li><p>Capable of handling markdown editing.</p></li>
<li><p>Contains a modular library to load the necessary functionality on demand.</p></li>
<li><p>Provides a fully customizable toolbar.</p></li>
<li><p>Provides HTML view to edit the source directly for developers.</p></li>
<li><p>Supports third - party library integration.</p></li>
<li><p>Allows preview of modified content before saving it.</p></li>
</ul>
</SfRichTextEditor>
```

You can refer to our [Blazor Rich Text Editor](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor Rich Text Editor](#) example to know how to render and configure the rich text editor tools.

Events in Blazor RichTextEditor Component

This section explains the list of events of the RichTextEditor component which will be triggered for an appropriate RichTextEditor actions.

OnActionBegin

OnActionBegin event triggers before command execution using the toolbar items.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorEvents OnActionBegin="@OnActionBeginHandler"
</RichTextEditorEvents>
</SfRichTextEditor>
@code{
public void OnActionBeginHandler(ActionBeginEventArgs args)
{
// Here you can customize your code
}
}
```

OnActionComplete

OnActionComplete event triggers after command execution using the toolbar items.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorEvents
OnActionComplete="@OnActionCompleteHandler"><RichTextEditorEvents>
</SfRichTextEditor>
@code{
public void OnActionCompleteHandler(ActionCompleteEventArgs args)
{
}
```

```
// Here you can customize your code
}
```

OnDialogOpen

OnDialogOpen event triggers when the dialog is being opened.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorEvents
OnDialogOpen="@OnDialogOpenHandler"><RichTextEditorEvents>
</SfRichTextEditor>
@code{
public void OnDialogOpenHandler(BeforeOpenEventArgs args)
{
// Here you can customize your code
}
}
```

DialogOpened

DialogOpened event triggers when a dialog is opened.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorEvents
DialogOpened="@DialogOpenedHandler"><RichTextEditorEvents>
</SfRichTextEditor>
@code{
public void DialogOpenedHandler(DialogOpenEventArgs args)
{
// Here you can customize your code
}
}
```

OnDialogClose

OnDialogClose event triggers when the dialog is being closed.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorEvents
OnDialogClose="@OnDialogCloseHandler"><RichTextEditorEvents>
</SfRichTextEditor>
@code{
public void OnDialogCloseHandler(BeforeCloseEventArgs args)
{
// Here you can customize your code
}
}
```

DialogClosed

DialogClosed event triggers after the dialog has been closed.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
  <RichTextEditorEvents
    DialogClosed="@DialogClosedHandler"><RichTextEditorEvents>
  </SfRichTextEditor>
  @code{
    public void DialogClosedHandler(DialogCloseEventArgs args)
    {
      // Here you can customize your code
    }
  }
```

OnQuickToolbarOpen

OnQuickToolbarOpen event triggers when the quick toolbar is being opened.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
  <RichTextEditorEvents OnQuickToolbarOpen="@OnQuickToolbarOpenHandler">
  <RichTextEditorEvents>
  </SfRichTextEditor>
  @code{
    public void OnQuickToolbarOpenHandler(BeforeQuickToolbarOpenEventArgs args)
    {
      // Here you can customize your code
    }
  }
```

QuickToolbarOpened

QuickToolbarOpened event triggers when a quick toolbar is opened.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
  <RichTextEditorEvents QuickToolbarOpened="@QuickToolbarOpenedHandler">
  <RichTextEditorEvents>
  </SfRichTextEditor>
  @code{
    public void QuickToolbarOpenedHandler(QuickToolbarEventArgs args)
    {
      // Here you can customize your code
    }
  }
```

QuickToolbarClosed

QuickToolbarClosed event triggers after the quick toolbar has been closed.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
  <RichTextEditorEvents QuickToolbarClosed="@QuickToolbarClosedHandler">
  </RichTextEditorEvents>
</SfRichTextEditor>
@code{
public void QuickToolbarClosedHandler(QuickToolbarEventArgs args)
{
  // Here you can customize your code
}
}
```

OnImageSelected

OnImageSelected event triggers when the image is selected or dragged into the insert image dialog.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
  <RichTextEditorEvents OnImageSelected="@OnImageSelectedHandler">
  </RichTextEditorEvents>
</SfRichTextEditor>
@code{
public void OnImageSelectedHandler(SelectedEventArgs args)
{
  // Here you can customize your code
}
}
```

BeforeUploadImage

BeforeUploadImage event triggers before the image upload process.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
  <RichTextEditorEvents BeforeUploadImage = "@BeforeUploadImageHandler">
  </RichTextEditorEvents>
</SfRichTextEditor>
@code{
public void BeforeUploadImageHandler(ImageUploadingEventArgs args)
{
  // Here you can customize your code
}
}
```

OnImageUploadSuccess

OnImageUploadSuccess event triggers when the image is successfully uploaded to the server side.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
```

```
<RichTextEditorEvents OnImageUploadSuccess="@OnImageUploadSuccessHandler">
<RichTextEditorEvents>
</SfRichTextEditor>
@code{
public void OnImageUploadSuccessHandler(ImageSuccessEventArgs args)
{
// Here you can customize your code
}
}
```

OnImageUploadFailed

OnImageUploadFailed event triggers when there is an error in the image upload.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorEvents OnImageUploadFailed="@OnImageUploadFailedHandler">
<RichTextEditorEvents>
</SfRichTextEditor>
@code{
public void OnImageUploadFailedHandler(ImageFailedEventArgs args)
{
// Here you can customize your code
}
}
```

OnImageRemoving

OnImageRemoving event triggers when the selected image is cleared from the insert image dialog.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorEvents OnImageRemoving="@OnImageRemovingHandler">
<RichTextEditorEvents>
</SfRichTextEditor>
@code{
public void OnImageRemovingHandler(RemovingEventArgs args)
{
// Here you can customize your code
}
}
```

OnImageDelete

OnImageDelete event triggers when the selected image is cleared from the Rich Text Editor Content.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorEvents OnImageDelete="@OnImageDeleteHandler">
<RichTextEditorEvents>
</SfRichTextEditor>
@code{
```

```
public void OnImageDeleteHandler(AfterImageDeleteEventArgs args)
{
    // Here you can customize your code
}
```

Created

Created event triggers when the Rich Text Editor is rendered.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorEvents Created="@CreatedHandler"><RichTextEditorEvents>
</SfRichTextEditor>
@code{
public void CreatedHandler(Object args)
{
    // Here you can customize your code
}
```

Destroyed

Destroyed event triggers when the Rich Text Editor is destroyed.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorEvents Destroyed="@DestroyedHandler"><RichTextEditorEvents>
</SfRichTextEditor>
@code{
public void DestroyedHandler(DestroyedEventArgs args)
{
    // Here you can customize your code
}
```

Blur

Blur event triggers when Rich Text Editor is focused out.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorEvents Blur="@BlurHandler"><RichTextEditorEvents>
</SfRichTextEditor>
@code{
public void BlurHandler(BlurEventArgs args)
{
    // Here you can customize your code
}
```

OnToolbarClick

OnToolbarClick event triggers when Rich Text Editor Toolbar items is clicked.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
  <RichTextEditorEvents
    OnToolbarClick="@OnToolbarClickHandler"><RichTextEditorEvents>
  </SfRichTextEditor>
@code{
  public void OnToolbarClickHandler(ToolbarClickEventArgs args)
  {
    // Here you can customize your code
  }
}
```

Focus

Focus event triggers when Rich Text Editor is focused in.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
  <RichTextEditorEvents Focus="@FocusHandler"><RichTextEditorEvents>
</SfRichTextEditor>
@code{
  public void FocusHandler(FocusEventArgs args)
  {
    // Here you can customize your code
  }
}
```

ValueChange

ValueChange event triggers only when Rich Text Editor is blurred and changes are done to the content.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
  <RichTextEditorEvents
    ValueChange="@ValueChangeHandler"><RichTextEditorEvents>
  </SfRichTextEditor>
@code{
  public void ValueChangeHandler(ChangeEventArgs args)
  {
    // Here you can customize your code
  }
}
```

OnResizeStart

OnResizeStart event triggers only when resizing the image is started.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorEvents
OnResizeStart="@OnResizeStartHandler"><RichTextEditorEvents>
</SfRichTextEditor>
@code{
public void OnResizeStartHandler(ResizeArgs args)
{
// Here you can customize your code
}
}
```

OnResizeStop

OnResizeStop event triggers only when resizing the image is stopped.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorEvents
OnResizeStop="@OnResizeStopHandler"><RichTextEditorEvents>
</SfRichTextEditor>
@code{
public void OnResizeStopHandler(ResizeArgs args)
{
// Here you can customize your code
}
}
```

AfterPasteCleanup

AfterPasteCleanup event triggers after cleaning up the copied content.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorEvents
AfterPasteCleanup="@AfterPasteCleanupHandler"><RichTextEditorEvents>
</SfRichTextEditor>
@code{
public void AfterPasteCleanupHandler(PasteCleanupArgs args)
{
// Here you can customize your code
}
}
```

How To

<!-- markdownlint-disable MD024 -->

Blazor RichTextEditor Component in WebAssembly App using Visual Studio

This article provides a step-by-step instructions to configure Syncfusion [Blazor Rich Text Editor](#) in a simple Blazor WebAssembly application using [Visual Studio 2019](#).

Starting with version 17.4.0.39 (2019 Volume 4), you need to include a valid license key (either paid or trial key) within your applications. Please refer to this help topic for more information.

Prerequisites

- Visual Studio 2019
- .NET Core SDK 3.1.3

.NET Core SDK 3.1.3 requires Visual Studio 2019 16.6 or later.

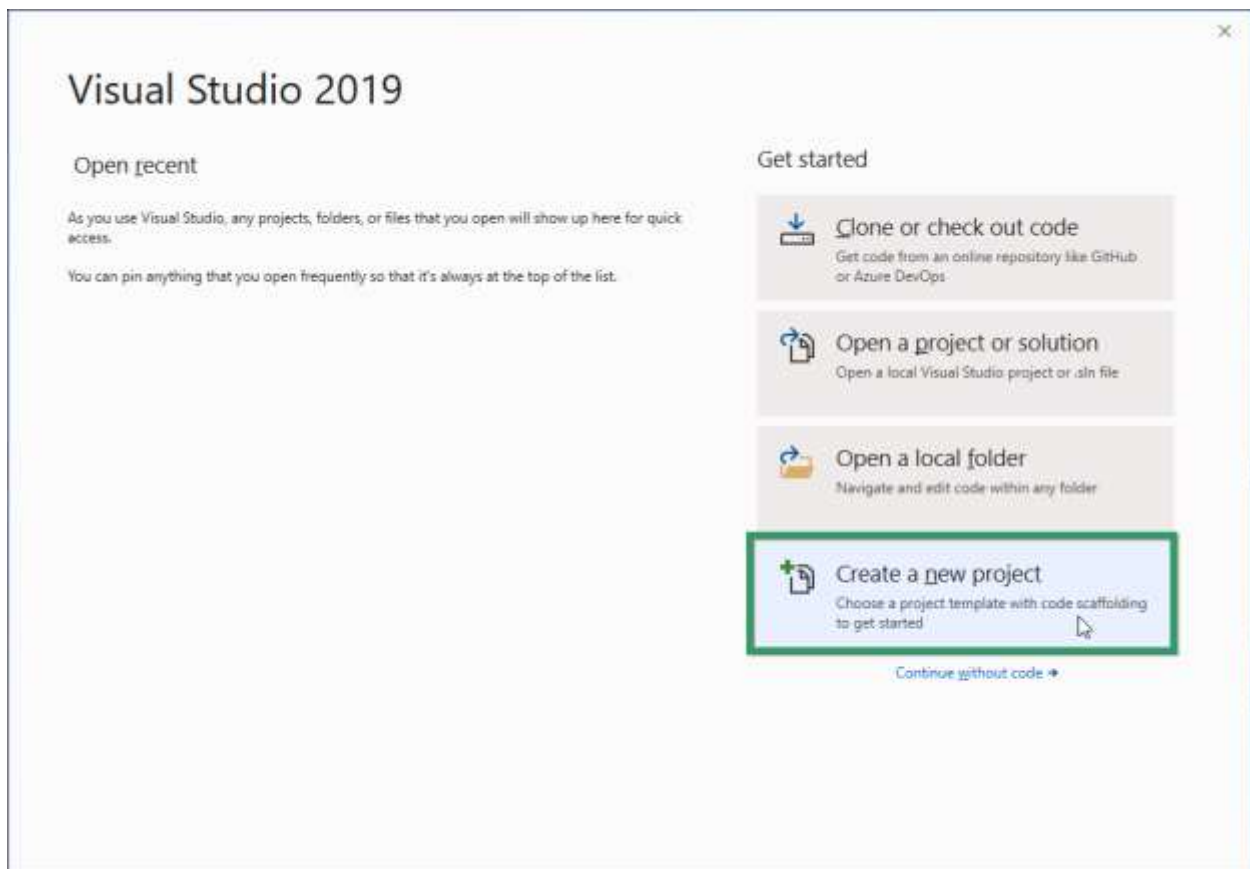
Syncfusion Blazor components are compatible with .NET Core 5.0 Preview 6 and it requires Visual Studio 16.7 Preview 1 or later.

Create a Blazor WebAssembly project in Visual Studio 2019

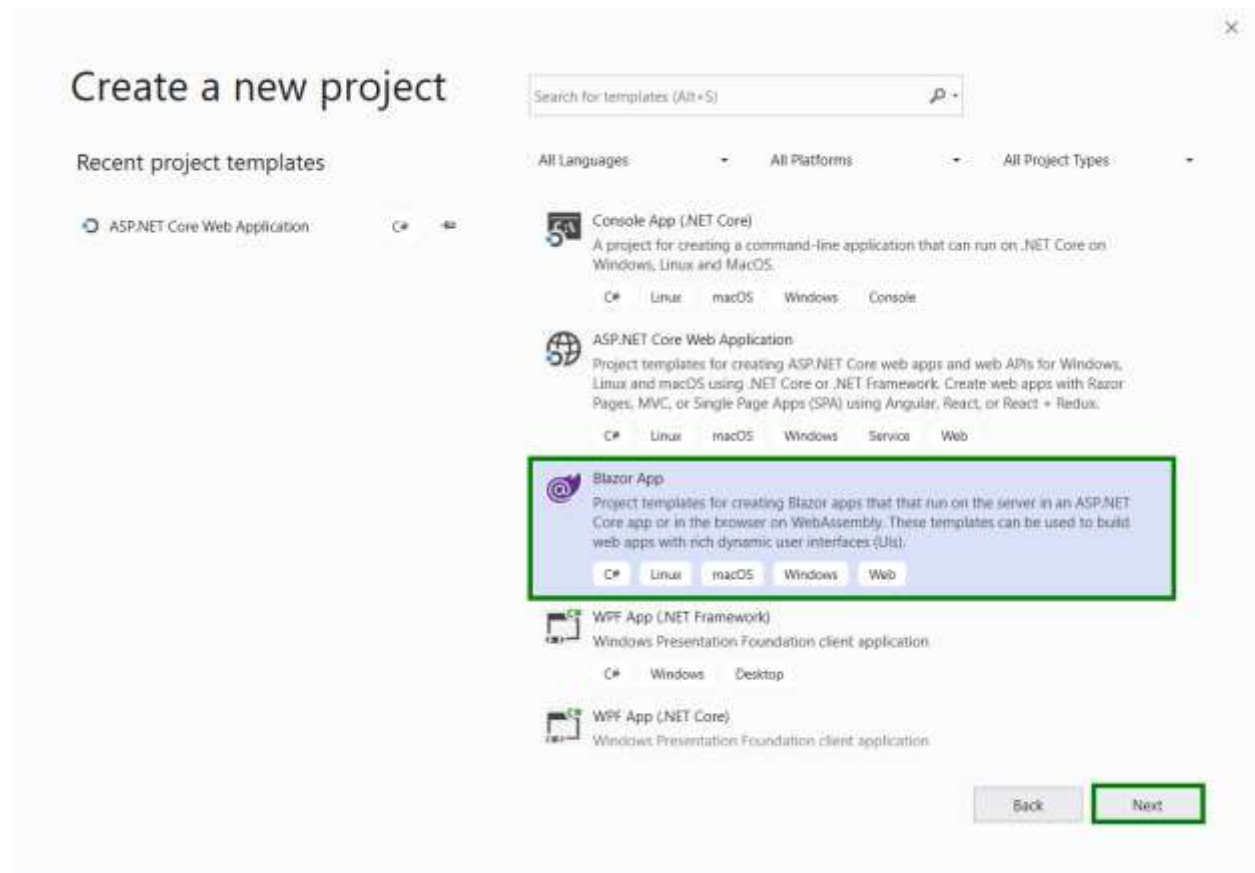
1. Install the essential project templates in the Visual Studio 2019 by running the below command line in the command prompt.

```
dotnet new -i Microsoft.AspNetCore.Components.WebAssembly.Templates::3.2.0-rc1.20223.4
```

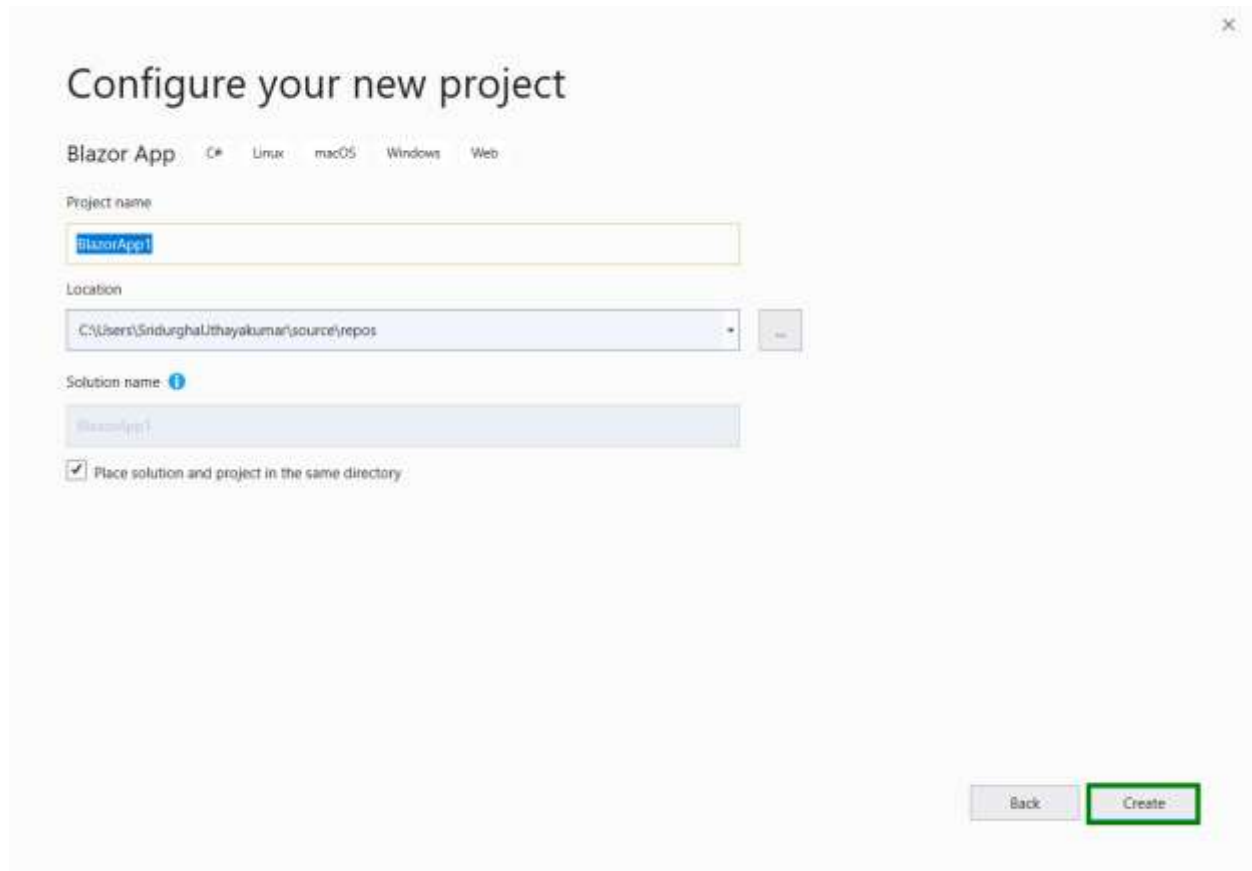
2. Choose **Create a new project** from the Visual Studio dashboard.



3. Select **Blazor App** from the template and click **Next** button.



4. Now, the project configuration window will popup. Click **Create** button to create a new project with the default project configuration.



Configure your new project

Blazor App C# Linux macOS Windows Web

Project name

BlazorApp1

Location

C:\Users\SindurghaUthayakumar\source\repos

Solution name ⓘ

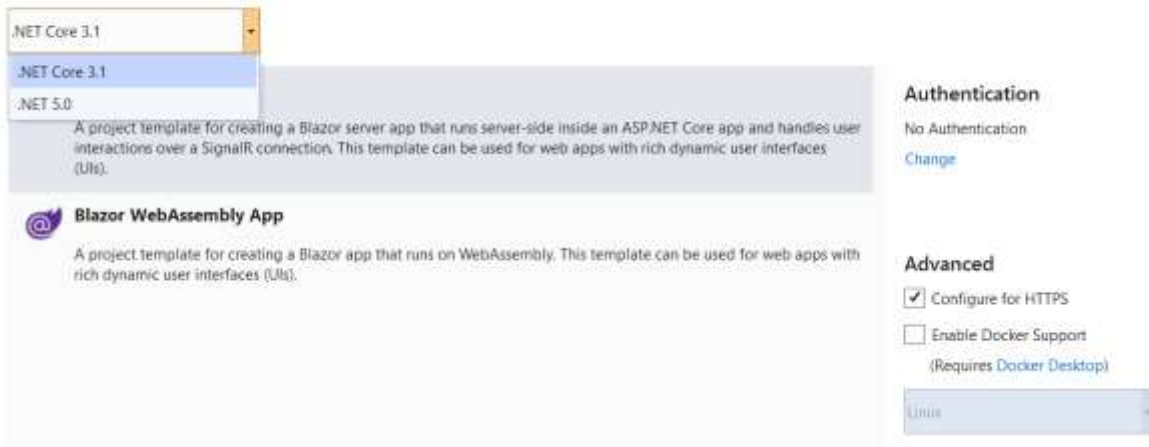
BlazorApp1

☒ Place solution and project in the same directory

Back Create

5. Choose **Blazor WebAssembly App** from the dashboard and click **Create** button to create a new Blazor WebAssembly application. Make sure **.NET Core and ASP.NET Core 3.1** is selected at the top.

Create a new Blazor app




.NET Core 3.1

.NET Core 3.1

.NET 5.0

A project template for creating a Blazor server app that runs server-side inside an ASP.NET Core app and handles user interactions over a SignalR connection. This template can be used for web apps with rich dynamic user interfaces (UIs).

 **Blazor WebAssembly App**

A project template for creating a Blazor app that runs on WebAssembly. This template can be used for web apps with rich dynamic user interfaces (UIs).

Authentication

No Authentication

[Change](#)

Advanced

☒ Configure for HTTPS

☐ Enable Docker Support
(Requires Docker Desktop)

Linux

ASP.NET Core 3.1 available in Visual Studio 2019 version.

Importing Syncfusion Blazor component in the application

You can use any one of the below standard to install the Syncfusion Blazor library in your application.

Importing Syncfusion Blazor component in the application

1. Install Syncfusion.Blazor.RichTextEditor NuGet package to the application by using the NuGet Package Manager.

Please ensure to check the Include prerelease option for our Beta release.

2. You can add the client-side resources through CDN or from NuGet package in the element of the ~/Pages/_Host.cshtml page.

HTML

```
<head>
<link href="_content/Syncfusion.Blazor.Themes/bootstrap4.css"
rel="stylesheet" />
</head>
```

For Internet Explorer 11 kindly refer the polyfills. Refer the [documentation](#) for more information.

HTML

```
<head>
<link href="_content/Syncfusion.Blazor.Themes/bootstrap4.css"
rel="stylesheet" />
<script
src="https://github.com/Daddoon/Blazor.Polyfill/releases/download/3.0.1/blaz
or.polyfill.min.js"></script>
</head>
```

Adding component package to the application

Open ~/_Imports.razor file and import the Syncfusion.Blazor.RichTextEditor package.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
```

Add SyncfusionBlazor service in Startup.cs

Open the **Startup.cs** file and add services required by Syncfusion components using `services.AddSyncfusionBlazor()` method. Add this method in the `ConfigureServices` function as follows.

C#

```
using Syncfusion.Blazor;
namespace BlazorApplication
{
    public class Startup
    {
        . . .
        . . .
    }
}
```

```
public void ConfigureServices(IServiceCollection services)
{
    ....
    ....
    services.AddSyncfusionBlazor();
}
}
```

Add Rich Text Editor component

To initialize the Rich Text Editor component, add the below code to your **Index.razor** view page which is present under **~/Pages** folder.

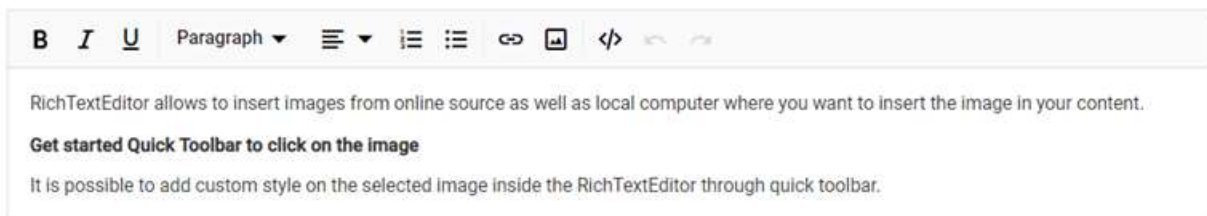
The following code explains how to initialize a simple Rich Text Editor in Razor page.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<p>Rich Text Editor allows to insert images from online source as well as local computer where you want to insert the image in your content.</p>
<p><b>Get started Quick Toolbar to click on the image</b></p>
<p>It is possible to add custom style on the selected image inside the Rich Text Editor through quick toolbar.</p>
</SfRichTextEditor>
```

Run the application

After successful compilation of your application, run the application.



Add Google fonts in Blazor RichTextEditor Component

To use web fonts in Rich Text Editor, the web fonts need not be present in the local machine. To add the web fonts to Rich Text Editor, refer to the web font links and add the font names in the **RichTextEditorFontFamily** tag.

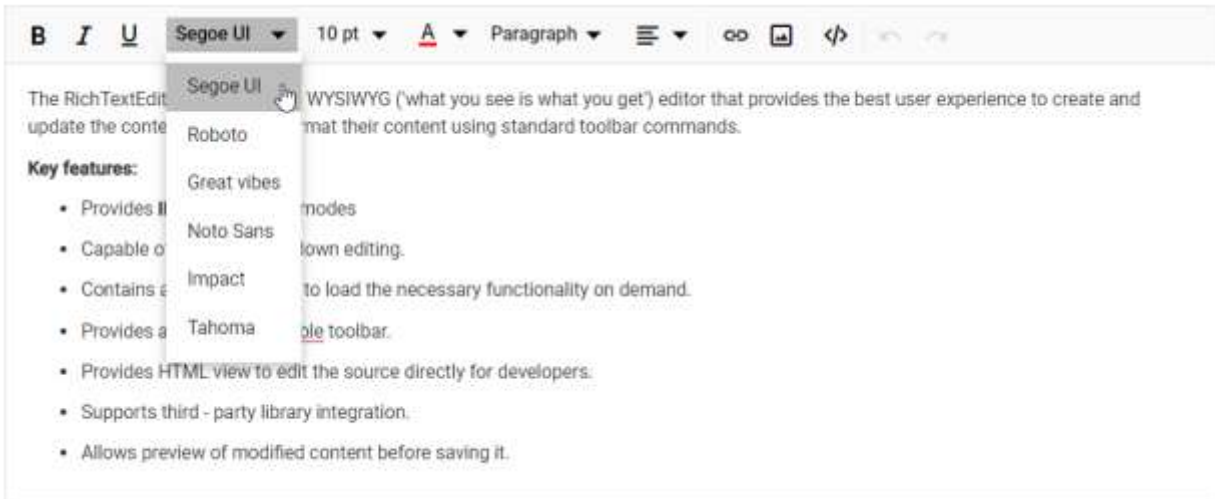
ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorToolbarSettings Items="@Tools" />
<RichTextEditorFontFamily Items="@FontFamilyItems" />
<p>The Rich Text Editor component is WYSIWYG ('what you see is what you get') editor that provides the best user experience to create and update the content. Users can format their content using standard toolbar commands.</p>
<p><b>Key features:</b></p>
<ul>
<li><p>Provides <b>IFRAME</b> and <b>DIV</b> modes </p></li>
<li><p>Capable of handling markdown editing.</p></li>
</ul>
```

```

</li><p> Contains a modular library to load the necessary functionality on
demand.</p></li>
<li><p> Provides a fully customizable toolbar.</p></li>
<li><p> Provides HTML view to edit the source directly for
developers.</p></li>
<li><p> Supports third - party library integration.</p></li>
<li><p> Allows preview of modified content before saving it.</p></li>
</ul>
</SfRichTextEditor>
@code {
private List<DropDownItemModel> FontFamilyItems = new
List<DropDownItemModel>()
{
new DropDownItemModel() { CssClass = "e-segoe-ui", Command = "Font",
SubCommand = "FontName", Text = "Segoe UI", Value = "Arial,Helvetica,sans-
serif" },
new DropDownItemModel() { CssClass = "e-arial", Command = "Font", SubCommand
= "FontName", Text = "Arial", Value = "Roboto" },
new DropDownItemModel() { CssClass = "e-georgia", Command = "Font",
SubCommand = "FontName", Text = "Georgia", Value = "Georgia,serif" },
new DropDownItemModel() { CssClass = "e-impact", Command = "Font",
SubCommand = "FontName", Text = "Impact", Value = "Impact,Charcoal,sans-
serif" },
new DropDownItemModel() { CssClass = "e-tahoma", Command = "Font",
SubCommand = "FontName", Text = "Tahoma", Value = "Tahoma,Geneva,sans-serif"
}
};
private List<ToolbarItemModel> Tools = new List<ToolbarItemModel>()
{
new ToolbarItemModel() { Command = ToolbarCommand.Bold },
new ToolbarItemModel() { Command = ToolbarCommand.Italic },
new ToolbarItemModel() { Command = ToolbarCommand.Underline },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.FontName },
new ToolbarItemModel() { Command = ToolbarCommand.FontSize },
new ToolbarItemModel() { Command = ToolbarCommand.FontColor },
new ToolbarItemModel() { Command = ToolbarCommand.BackgroundColor },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.Formats },
new ToolbarItemModel() { Command = ToolbarCommand.Alignments },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.CreateLink },
new ToolbarItemModel() { Command = ToolbarCommand.Image },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.SourceCode },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.Undo },
new ToolbarItemModel() { Command = ToolbarCommand.Redo }
};
}

```



The following font style links are referred in the page.

HTML

```
<link rel="stylesheet" href="http://fonts.googleapis.com/css?family=Roboto">
<link rel="stylesheet"
href="http://fonts.googleapis.com/css?family=Great+Vibes">
```

You can also add the two Google web fonts (Roboto and Great vibes) to Rich Text Editor.

Change default font-family in Blazor RichTextEditor Component

By using **Default** property, you can change the default font-family of the Rich Text Editor. To change the font-family of the Rich Text Editor content while loading, the **RichTextEditorFontFamily** has to be given in the style section with the help of **CssClass** property.

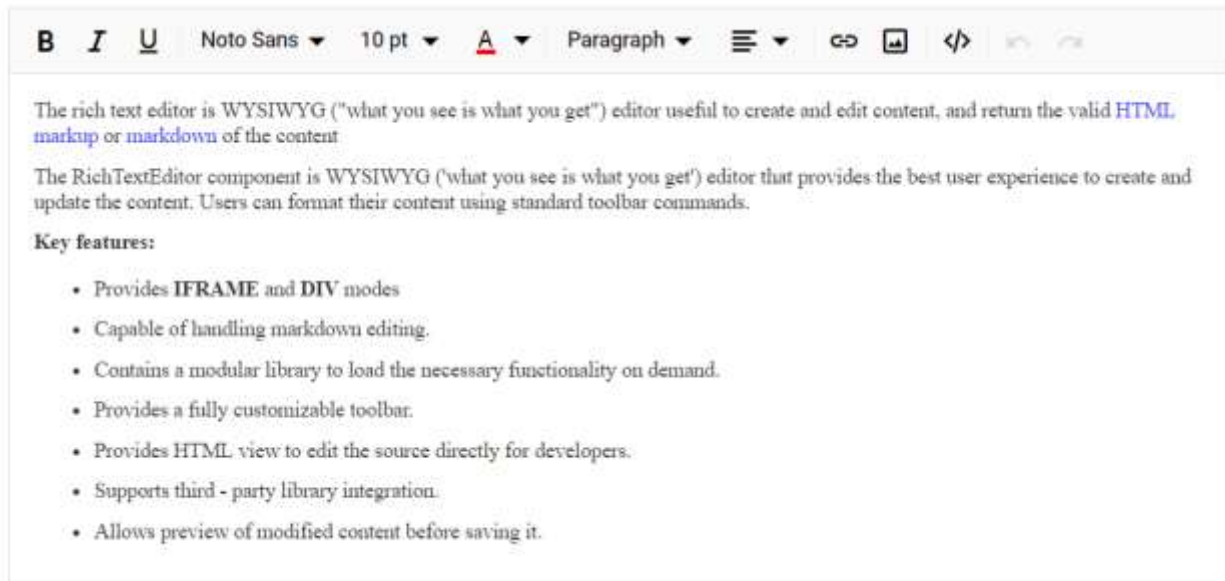
ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor CssClass="customClass">
<RichTextEditorToolbarSettings Items="@Tools" />
<RichTextEditorFontFamily Default="Georgia" Items="@FontFamilyItems" />
<p>The Rich Text Editor component is WYSIWYG ('what you see is what you get') editor that provides the best user experience to create and update the content. Users can format their content using standard toolbar commands.</p>
<p><b>Key features:</b></p>
<ul>
<li><p>Provides <b>IFRAME</b> and <b>DIV</b> modes </p></li>
<li><p>Capable of handling markdown editing.</p></li>
<li><p>Contains a modular library to load the necessary functionality on demand.</p></li>
<li><p>Provides a fully customizable toolbar.</p></li>
<li><p>Provides HTML view to edit the source directly for developers.</p></li>
<li><p>Supports third - party library integration.</p></li>
<li><p>Allows preview of modified content before saving it.</p></li>
</ul>
</SfRichTextEditor>
<style>
```

```

.customClass .e-rte-content .e-content {
/* to get the desired font on intially*/
font-family: "Georgia,serif";
}
</style>
@code{
private List<DropDownItemModel> FontFamilyItems = new
List<DropDownItemModel>()
{
new DropDownItemModel() { CssClass = "e-segoe-ui", Command = "Font",
SubCommand = "FontName", Text = "Segoe UI", Value = "Arial,Helvetica,sans-
serif" },
new DropDownItemModel() { CssClass = "e-arial", Command = "Font", SubCommand
= "FontName", Text = "Arial", Value = "Roboto" },
new DropDownItemModel() { CssClass = "e-georgia", Command = "Font",
SubCommand = "FontName", Text = "Georgia", Value = "Georgia,serif" },
new DropDownItemModel() { CssClass = "e-impact", Command = "Font",
SubCommand = "FontName", Text = "Impact", Value = "Impact,Charcoal,sans-
serif" },
new DropDownItemModel() { CssClass = "e-tahoma", Command = "Font",
SubCommand = "FontName", Text = "Tahoma", Value = "Tahoma,Geneva,sans-serif"
}
};
private List<ToolbarItemModel> Tools = new List<ToolbarItemModel>()
{
new ToolbarItemModel() { Command = ToolbarCommand.Bold },
new ToolbarItemModel() { Command = ToolbarCommand.Italic },
new ToolbarItemModel() { Command = ToolbarCommand.Underline },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.FontName },
new ToolbarItemModel() { Command = ToolbarCommand.FontSize },
new ToolbarItemModel() { Command = ToolbarCommand.FontColor },
new ToolbarItemModel() { Command = ToolbarCommand.BackgroundColor },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.Formats },
new ToolbarItemModel() { Command = ToolbarCommand.Alignments },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.CreateLink },
new ToolbarItemModel() { Command = ToolbarCommand.Image },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.SourceCode },
new ToolbarItemModel() { Command = ToolbarCommand.Separator },
new ToolbarItemModel() { Command = ToolbarCommand.Undo },
new ToolbarItemModel() { Command = ToolbarCommand.Redo }
};
}

```

Check image size in Blazor RichTextEditor Component

By using the Rich text editor's **OnImageUploading** event, you can get the image size before uploading and restrict the image to upload, when the given image size is greater than the allowed size.

In the following, the image size has been validated before uploading and determined whether the image has been uploaded or not.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorImageSettings
SaveUrl="https://aspnetmvc.syncfusion.com/services/api/uploadbox/Save"
Path="./Images/" />
<RichTextEditorEvents BeforeUploadImage="@BeforeUploadImage" />
</SfRichTextEditor>
@code {
private void BeforeUploadImage(ImageUploadingEventArgs args)
{
var sizeInBytes = args.FilesData[0].Size;
var imgSize = 500000;
if (imgSize < sizeInBytes) {
args.Cancel = true;
}
}
}
```

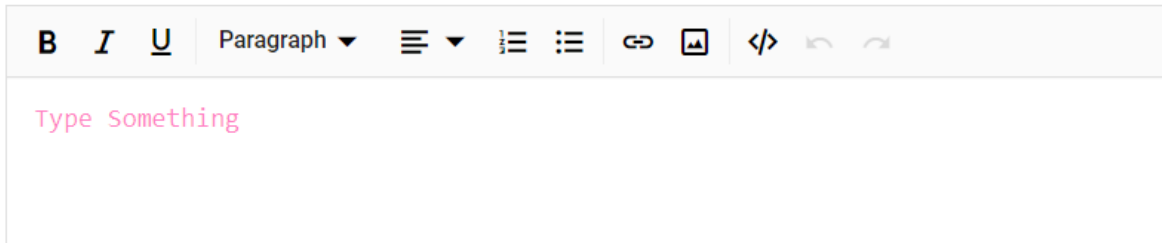
Customize placeholder style in Blazor RichTextEditor Component

By using **e-rte-placeholder** class, you can customize the placeholder style.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor Placeholder="Type Something" />
<style>
```

```
.e-richtexteditor .e-rte-placeholder {
/* placeholder style */
font-family: monospace;
color: deeppink;
}
</style>
```



Rename images before inserting it in Blazor RichTextEditor Component

By using the `RichTextEditorImageSettings` property, the server handler can be specified to upload and rename the selected image. Then, the `OnImageUploadSuccess` event could be bound, to receive the modified file name from the server and update it in the Rich Text Editor's insert image dialog.

The runnable Blazor Server app demo is available in this [Github](#) repository.

ASPX-CS

```
@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorImageSettings
SaveUrl="[SERVICE_HOSTED_PATH]/api/Image/Rename" Path="./Images/" />
<RichTextEditorEvents OnImageUploadSuccess="@ImageUploadSuccess" />
</SfRichTextEditor>
@code {
public string[] header { get; set; }
private void ImageUploadSuccess(ImageSuccessEventArgs args)
{
var headers = args.Response.Headers.ToString();
header = headers.Split("name: ");
header = header[1].Split("\r");
// Update the modified image name to display a image in the editor.
args.FileName = header[0];
}
}
```

To configure the server-side handler in the Web API service, refer the below code.

CSHARP

```
using System;
using System.IO;
using System.Net.Http.Headers;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Http;
using System.Collections.Generic;
using Microsoft.AspNetCore.Hosting;
```

```

using Microsoft.AspNetCore.Http.Features;
namespace RenameImage.Controllers
{
    [ApiController]
    public class ImageController : ControllerBase
    {
        private double x;
        private string imageFileName;
        private readonly IWebHostEnvironment hostingEnv;
        public ImageController(IWebHostEnvironment env)
        {
            this.hostingEnv = env;
        }
        [HttpPost("[action]")]
        [Route("api/Image/Rename")]
        public void Rename(IList<IFormFile> UploadFiles)
        {
            try
            {
                foreach (IFormFile file in UploadFiles)
                {
                    if (UploadFiles != null)
                    {
                        string targetPath = hostingEnv.ContentRootPath + "\\wwwroot\\Images";
                        string filename =
                            ContentDispositionHeaderValue.Parse(file.ContentDisposition).FileName.Trim('
                            ');
                        // Create a new directory, if it does not exists
                        if (!Directory.Exists(targetPath))
                        {
                            Directory.CreateDirectory(targetPath);
                        }
                        imageFileName = filename;
                        string path = hostingEnv.WebRootPath + "\\Images" + @"\" + filename;
                        // Rename a uploaded image file name
                        while (System.IO.File.Exists(path))
                        {
                            imageFileName = "rteImage" + x + "-" + filename;
                            path = hostingEnv.WebRootPath + "\\Images" + @"\" + "rteImage{x}-{filename}";
                            x++;
                        }
                        if (!System.IO.File.Exists(path))
                        {
                            using (FileStream fs = System.IO.File.Create(path))
                            {
                                file.CopyTo(fs);
                                fs.Flush();
                                fs.Close();
                            }
                            // Modified file name shared through response header by adding custom header
                            Response.Headers.Add("name", imageFileName);
                            Response.StatusCode = 200;
                            Response.ContentType = "application/json; charset=utf-8";
                        }
                    }
                }
            }
        }
    }
}

```

```

catch (Exception e)
{
    Response.Clear();
    Response.ContentType = "application/json; charset=utf-8";
    Response.HttpContext.Features.Get<IHttpResponseFeature>().ReasonPhrase =
    e.Message;
}
}
}
}

```

Format code block using toolbar in Blazor RichTextEditor Component

You can configure code block formatting as a separate toolbar button by adding the **InsertCode** Command within the **RichTextEditorToolbarSettings - Items** property. The InsertCode button has a toggle state to apply code block formatting to the editor and remove code block formatting from the editor.

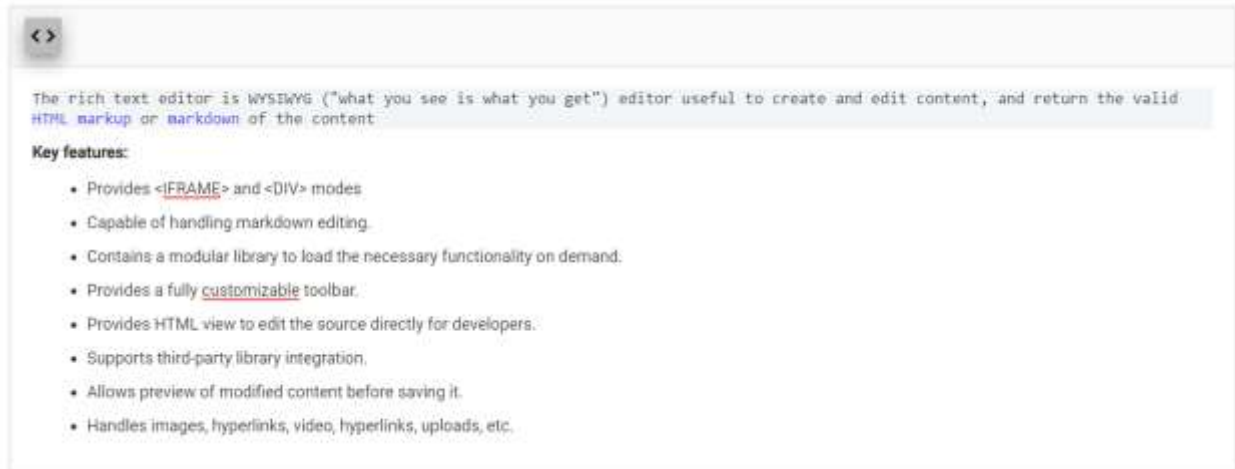
The following code will configure the InsertCode button in toolbar and set the background color to “pre” tag for highlighting the code block.

ASPX-CS

```

@using Syncfusion.Blazor.RichTextEditor
<SfRichTextEditor>
<RichTextEditorToolbarSettings Items="@Tools" />
<p>The Rich Text Editor component is WYSIWYG ('what you see is what you
get') editor that provides the best user experience to create and update the
content. Users can format their content using standard toolbar commands.</p>
<p><b>Key features:</b></p>
<ul>
<li><p>Provides <b>IFRAME</b> and <b>DIV</b> modes </p></li>
<li><p>Capable of handling markdown editing.</p></li>
<li><p>Contains a modular library to load the necessary functionality on
demand.</p></li>
<li><p>Provides a fully customizable toolbar.</p></li>
<li><p>Provides HTML view to edit the source directly for
developers.</p></li>
<li><p>Supports third - party library integration.</p></li>
<li><p>Allows preview of modified content before saving it.</p></li>
</ul>
</SfRichTextEditor>
<style>
.e-richtexteditor .e-rte-content .e-content pre {
padding: 10px;
background: #F4F5F7 !important;
}
</style>
@code{
private List<ToolbarItemModel> Tools = new List<ToolbarItemModel>()
{
    new ToolbarItemModel() { Command = ToolbarCommand.InsertCode }
};
}

```



Scheduler

Getting Started with Blazor Scheduler Component

This section briefly explains about how to include a simple Scheduler Component in the Blazor Server-Side and Client-Side application. Refer to Getting Started with [Blazor Server-Side Scheduler](#) and [Blazor WebAssembly Scheduler](#) documentation pages for configuration specifications..

To get start quickly with Blazor Scheduler, you can check on this video:

{% youtube

"youtube:https://www.youtube.com/watch?v=bXaHu6qjxV8"%}

Importing Syncfusion Blazor component in the application

You can use any one of the below standards to install the Syncfusion Blazor library in your application.

Using Syncfusion Blazor individual NuGet Packages [New standard]

Starting with Volume 4, 2020 (v18.4.0.30) release, Syncfusion provides [individual NuGet packages](#) for our Syncfusion Blazor components. We highly recommend this new standard for your Blazor production applications. Refer to [this section](#) to know the benefits of the individual NuGet packages.

1. Install **Syncfusion.Blazor.Schedule** NuGet package to the application using the **NuGet Package Manager**.
2. Add the client-side style resources through [CDN](#) or from [NuGet](#) package in the `element` of the `~/Pages/_Host.cshtml` page.

HTML

```
<head>
...
...
<link href="_content/Syncfusion.Blazor.Themes/bootstrap4.css"
rel="stylesheet" />
</head>
```

Warning: Syncfusion.Blazor package should not be installed along with [individual NuGet packages](#). Hence, you have to add the above Syncfusion.Blazor.Themes static web assets (styles) in the application.

Using Syncfusion.Blazor NuGet Package [Old standard]

Warning: If you prefer the above new standard (individual NuGet packages), then skip this section. Using both old and new standards in the same application will throw ambiguous compilation errors.

1. Install **Syncfusion.Blazor** NuGet package to the newly created application by using the **NuGet Package Manager**.
2. Add the client-side style resources through [CDN](#) or from [NuGet](#) package in the element of the ~/Pages/_Host.cshtml page.

HTML

```
<head>
...
...
<link href="_content/Syncfusion.Blazor/styles/bootstrap4.css"
rel="stylesheet" />
@*<link href="https://cdn.syncfusion.com/blazor/{{ site.blazorversion
}}/styles/bootstrap4.css" rel="stylesheet" />*@
</head>
```

For Internet Explorer 11 kindly refer the polyfills. Refer the [documentation](#) for more information.

HTML

```
<head>
...
<link href="https://cdn.syncfusion.com/blazor/{{ site.blazorversion
}}/styles/bootstrap4.css" rel="stylesheet" />
<script
src="https://github.com/Daddoon/Blazor.Polyfill/releases/download/3.0.1/blaz
or.polyfill.min.js"></script>
...
</head>
```

[Adding component package to the application](#)

Open ~/_Imports.razor file and import the Syncfusion.Blazor.Schedule package.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
```

[Add SyncfusionBlazor service in Startup file](#)

Open the **Startup.cs** file and add services required by Syncfusion components using **services.AddSyncfusionBlazor()** method. Add this method in the **ConfigureServices** function as follows.

CSHARP

```
using Syncfusion.Blazor;
namespace BlazorApplication
```

```

{
    public class Startup
    {
        ....
        ....
        public void ConfigureServices(IServiceCollection services)
        {
            ....
            ....
            services.AddSyncfusionBlazor();
        }
    }
}

```

Initialize the Scheduler component

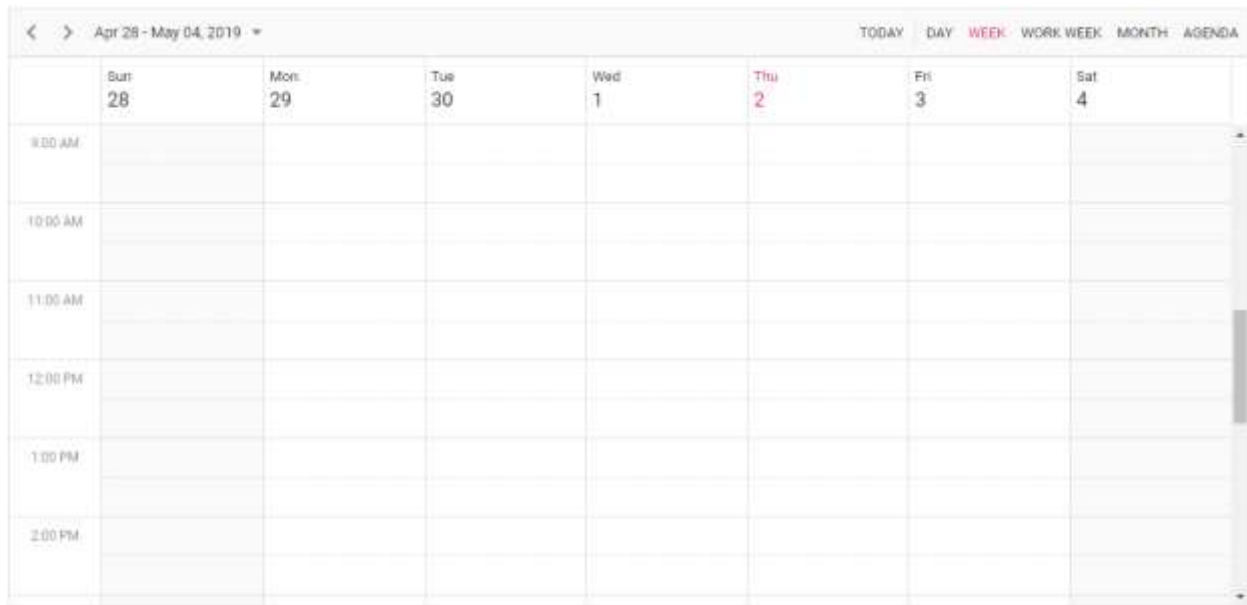
The [Blazor Scheduler](#) component can be rendered on the page by defining the `SfSchedule` tag helper. And also, define the required Scheduler views in the `ScheduleView` tag helper. Add the following code example to the `index.razor` page which is available within the `~/Pages/` folder, to initialize the Scheduler component.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue=AppointmentData>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code {
    public class AppointmentData
    {
        public int Id { get; set; }
        public string Subject { get; set; }
        public string Location { get; set; }
        public DateTime StartTime { get; set; }
        public DateTime EndTime { get; set; }
        public string Description { get; set; }
        public bool IsAllDay { get; set; }
        public string RecurrenceRule { get; set; }
        public string RecurrenceException { get; set; }
        public Nullable<int> RecurrenceID { get; set; }
    }
}

```



Populating appointments

To populate the Scheduler with appointments, bind the event data to it by assigning the `DataSource` property under `ScheduleEventSettings`.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="650px" @bind-
SelectedDate="@CurrentDate">
  <ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
  <ScheduleViews>
    <ScheduleView Option="View.Day"></ScheduleView>
    <ScheduleView Option="View.Week"></ScheduleView>
    <ScheduleView Option="View.WorkWeek"></ScheduleView>
    <ScheduleView Option="View.Month"></ScheduleView>
    <ScheduleView Option="View.Agenda"></ScheduleView>
  </ScheduleViews>
</SfSchedule>

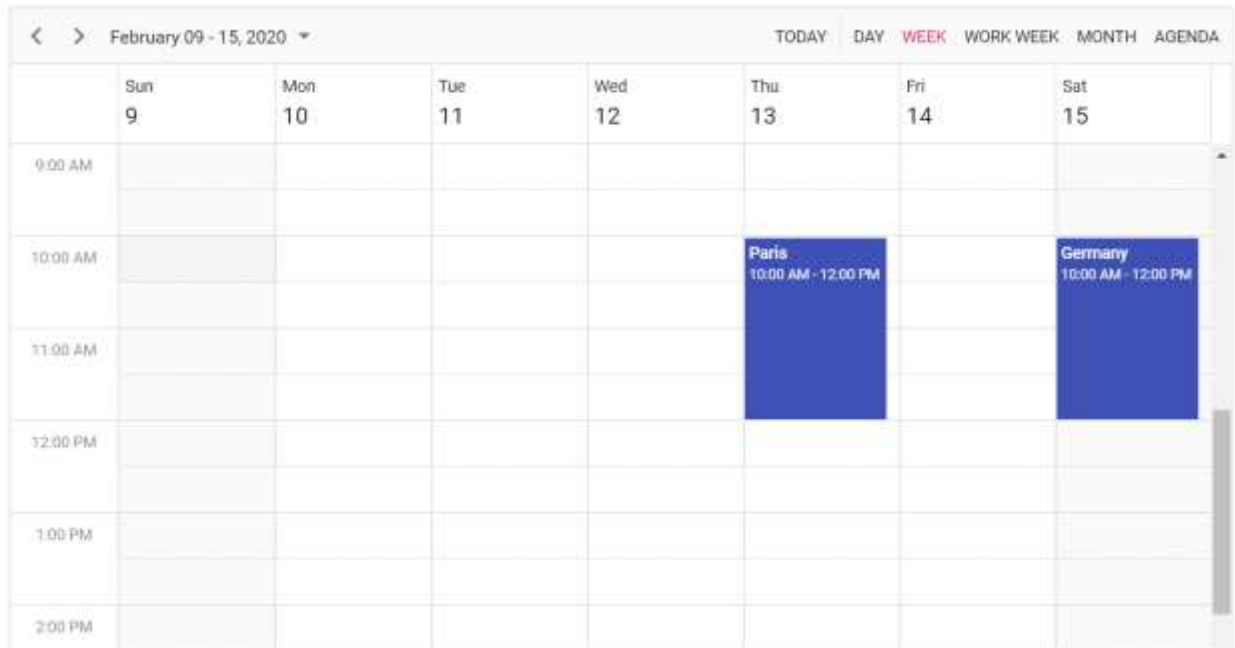
@code{
  DateTime CurrentDate = new DateTime(2020, 2, 14);
  List<AppointmentData> DataSource = new List<AppointmentData>
  {
    new AppointmentData { Id = 1, Subject = "Paris", StartTime = new
    DateTime(2020, 2, 13, 10, 0, 0) , EndTime = new DateTime(2020, 2, 13, 12, 0,
    0) },
    new AppointmentData { Id = 2, Subject = "Germany", StartTime = new
    DateTime(2020, 2, 15, 10, 0, 0) , EndTime = new DateTime(2020, 2, 15, 12, 0,
    0) }
  };
  public class AppointmentData
  {
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
  }
}
```



```

public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```



Setting date

The [Blazor Scheduler](#) usually displays the system date as its current date. To change the current date of Scheduler with specific date, define the two-way binding for `SelectedDate` property.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="650px" @bind-
SelectedDate="@CurrentDate">
  <ScheduleViews>
    <ScheduleView Option="View.Day"></ScheduleView>
    <ScheduleView Option="View.Week"></ScheduleView>
    <ScheduleView Option="View.WorkWeek"></ScheduleView>
    <ScheduleView Option="View.Month"></ScheduleView>
    <ScheduleView Option="View.Agenda"></ScheduleView>
  </ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 10);
public class AppointmentData
{
  public int Id { get; set; }
  public string Subject { get; set; }
  public string Location { get; set; }
  public DateTime StartTime { get; set; }
  public DateTime EndTime { get; set; }
}

```

```

public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}

```

Setting view

The Scheduler displays **Week** view by default. To change the current view, define the applicable view name to the two-way binding of **CurrentView** property. The applicable view names are,

- Day
- Week
- WorkWeek
- Month
- Agenda
- MonthAgenda
- TimelineDay
- TimelineWeek
- TimelineWorkWeek
- TimelineMonth
- TimelineYear
- Year

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="650px" @bind-
CurrentView="@CurrentView">
  <ScheduleViews>
    <ScheduleView Option="View.Day"></ScheduleView>
    <ScheduleView Option="View.Week"></ScheduleView>
    <ScheduleView Option="View.WorkWeek"></ScheduleView>
    <ScheduleView Option="View.Month"></ScheduleView>
    <ScheduleView Option="View.Agenda"></ScheduleView>
  </ScheduleViews>
</SfSchedule>
@code{
View CurrentView = View.Month;
public class AppointmentData
{
  public int Id { get; set; }
  public string Subject { get; set; }
  public string Location { get; set; }
  public DateTime StartTime { get; set; }
  public DateTime EndTime { get; set; }
  public string Description { get; set; }
  public bool IsAllDay { get; set; }
  public string RecurrenceRule { get; set; }
  public string RecurrenceException { get; set; }
  public Nullable<int> RecurrenceID { get; set; }
}

```

```
}

```

Individual view customization

Each individual Scheduler views can be customized with its own options such as setting different start and end hour on Week and Work Week views, whereas hiding the weekend days on Month view alone which can be achieved by defining the `ScheduleView`.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="650px" @bind-
SelectedDate="@CurrentDate">
  <ScheduleViews>
    <ScheduleView Option="View.Week" StartHour="07:00"
EndHour="15:00"></ScheduleView>
    <ScheduleView Option="View.WorkWeek" StartHour="10:00"
EndHour="18:00"></ScheduleView>
    <ScheduleView Option="View.Month" MaxEventsPerRow="2"
ShowWeekend="false"></ScheduleView>
  </ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 2, 13);
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
}
}
```

You can refer to our [Blazor Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor Scheduler example](#) to understand how to manage appointments with multiple resources.

See Also

1. [Getting Started with Syncfusion Blazor for client-side in .NET Core CLI](#)
2. [Getting Started with Syncfusion Blazor for client-side in Visual Studio 2019](#)
3. [Getting Started with Syncfusion Blazor for server-side in .NET Core CLI](#)

Scheduler Interactions in Blazor Scheduler Component

The following table describes the Scheduler actions and also illustrates how those actions are carried out through mouse and touch interactions on Scheduler.

| Actions | Mouse interaction | Touch interaction |

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

| Single click or tap on cells | Single click on a cell to select a cell. | Single tapping on cells, will display a + icon on the cell. Tapping on it again will open the new event editor window. |

| Multiple cell selection | Single click on a cell and drag the selection to other cells to enable multiple cell selection. | No multiple cell selection is allowed using touch gestures. |

| Event selection | Single click on an event to select it. | Tap holding on events, select an event and opens a small popup at the top holding the options to edit or delete. The popup also displays the selected event's subject. |

| Multiple event selection and deletion | Pressing **Ctrl** key and altogether single clicking on multiple events one after the other will enable multiple event selection. Pressing **Delete** key after event selection will delete all the selected events. | Tap hold an event to select it, which opens a small popup at the top holding the options to edit or delete. As a continuation of this action, keep on single tapping on other events, to enable multiple event selection. Also, the popup displayed at the top remains in opened state, showing the count of the number of selected events. Pressing **Delete** option from the popup will delete all the selected events. |

| Date navigation | Clicking on the previous or next date navigation icons in the header bar allows to navigate between dates. | Swiping the scheduler view port to the left or right will allow to navigate between the dates on touch devices. NOTE: Swiping does not work when horizontal scroller present in the Scheduler. You can also make use of the previous and next navigation icons at the header bar to navigate. |

| View navigation | Click on an event and try moving it over the Scheduler to enable drag and drop action. | The view options are available within the popup options at the top right extreme end of the header bar and you can choose the view from it. |

| Drag and drop | Click on an event and try moving it over the Scheduler to enable drag and drop action. | Tap hold the event and try moving it over the Scheduler to enable drag and drop action. |

| Event resizing | Hover the mouse across the extremities or edges of the Scheduler events and when the mouse pointer changes into resize handler, now click and start resizing an event to the desired time range. | Touch the event extremities and start resizing the events directly. |

| Tooltip | Hover the mouse pointer over the events or resource header and the tooltip will be displayed. | Tap holding the events will open the tooltip on events. |

| Open editor window | Double click on cells or events to open the editor window. | Double click on cells or events to open the editor window. Single tap on cells, which displays a + icon on the cell. Now, tap on it again to open the new event editor window. To open the editor on events, single tap on it and then click on the edit icon to open the editor window in **Edit** mode. |

| Open quick info popup | Single clicking on a cell will open a quick popup prompting for new event creation. Single clicking on an event will open a popup displaying event information along with the option to edit and delete it. | No quick info popup is available while single tapping on cells. Single tapping on events, opens the popup showing event information. |

Appointments in Blazor Scheduler Component

Appointments can be anything that are scheduled for a specific time period. It can be created on varied time range and each appointments are categorized based on this range. The Scheduler events can be categorized as,

- Normal events
- Spanned events
- All-day events
- Recurring events

To get start quickly about appointments and how to customize it, you can check on this video:

{% youtube

"youtube:https://www.youtube.com/watch?v=Vtl1Wyuwt-0"%}

Normal events

Represents an appointment that is created for any specific time interval within a day.

Creating a normal event

The following example depicts how to define a normal event on the Scheduler, with event data being loaded from simple list of appointment collection.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 31);
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
}
}
```

Spanned events

Represents an appointment that is created for more than 24 hours, and usually displayed on the all-day row. Also, represents another type of appointment that is created for more than one day but less than 24 hours, and usually displayed appropriately on both the days.

For example, in week view if an appointment is created for two days say from November 25, 2020 – 11.00 PM to November 26, 2020 - 2.00 AM but less than 24 hours time interval, then the appointment split into two partitions and will be displayed on both the days.

All-day events

It represents an appointment that is created for an entire day such as holiday events. It is usually displayed separately in an all-day row, a separate row for all-day appointments below the date header section. In Timeline views, the all-day appointments displays in the working space area, and no separate all-day row is present in that view.

To change normal appointment into all-day event, set `IsAllDay` field to true.

Hide all-day row events

The CSS customization can be used to prevent the display of all-day row appointments on the Scheduler UI.

CSS

```
.e-schedule .e-date-header-wrap .e-schedule-table thead {
display: none;
}
```

Recurring events

It represents an appointment that is created for a certain time interval and occurring repeatedly on a daily, weekly, monthly or yearly basis at the same time interval based on the provided recurrence rule. Usually, the recurring events are indicated by a repeat marker added at the bottom-right position.

Creating a recurring event

The following example depicts how to create a recurring event on Scheduler with the specific recurrence rule. In the following example, an event is made to repeat on daily mode and ends after 5 occurrences.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 9);
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 6, 9, 30, 0) , EndTime = new DateTime(2020, 1, 6, 11, 0,
0),
RecurrenceRule = "FREQ=DAILY;INTERVAL=1;COUNT=5" }
};
public class AppointmentData
```

```
{
public int Id { get; set; }
public string Subject { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public Nullable<int> RecurrenceID { get; set; }
public string RecurrenceException { get; set; }
}
}
```

Adding exceptions

A few instance of the recurrence series can be excluded on specific dates, by adding those exceptional dates to the `RecurrenceException` field. These date values should be given in the ISO date time format with no hyphens(-) separating the date elements.

For example, 7th January 2020 can be represented as 20200107. Also, the time part being represented in UTC format needs to add "Z" after the time portion with no space. "09:30 AM" is therefore represented as "040000Z".

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 6);
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 6, 9, 30, 0) , EndTime = new DateTime(2020, 1, 6, 11, 0,
0),
RecurrenceRule = "FREQ=DAILY;INTERVAL=1;COUNT=5", RecurrenceException =
"20200107T040000Z,20200109T040000Z" }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}
```

```
}

```

Editing an occurrence from a series

To dynamically edit a particular occurrence from an event series and display it on the initial load of Scheduler, the edited occurrence needs to be added as a new event to the `dataSource` collection, with an additional `RecurrenceID` field defined to it. The `RecurrenceID` field of edited occurrence usually maps the ID value of the parent event.

In this example, a recurring instance that displays on the date 30th January 2020 is edited with different timings. Therefore, this particular date is excluded from the parent recurring event that repeats from 28th January 2020 to 1st February 2020. This can be done by adding the `RecurrenceException` field with the excluded date value on the parent event. Also, the edited occurrence event which is created as a new event should carry the `RecurrenceID` field pointing to the parent event's `Id` value.

ASPX-CS

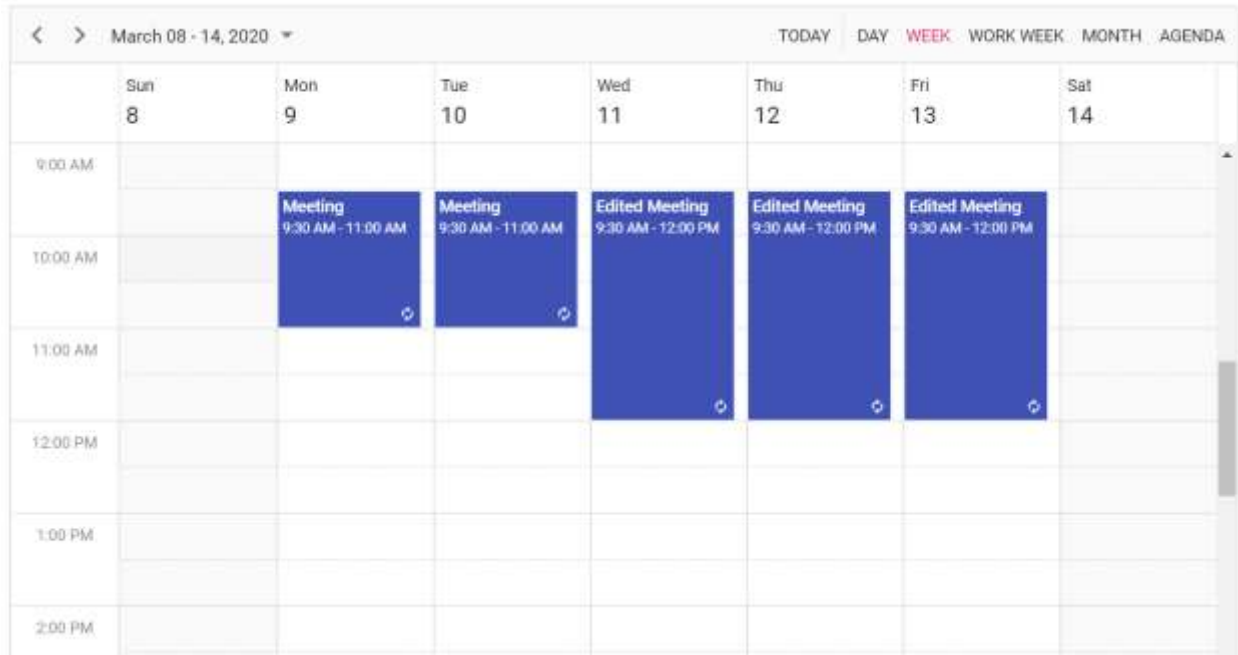
```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 31);
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Scrum Meeting", StartTime = new
DateTime(2020, 1, 28, 9, 30, 0) , EndTime = new DateTime(2020, 1, 28, 11, 0,
0),
RecurrenceRule = "FREQ=DAILY;INTERVAL=1;COUNT=5", RecurrenceException =
"20200130T040000Z" },
new AppointmentData { Id = 2, Subject = "Scrum Meeting Rescheduled",
StartTime = new DateTime(2020, 1, 30, 10, 30, 0) , EndTime = new
DateTime(2020, 1, 30, 12, 0, 0), RecurrenceID = 1 }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}
```


Edit/Delete following recurrence events

The Scheduler allows the user to edit the following recurrence events by setting true value to **AllowEditFollowingEvents** within the **ScheduleEventSettings** tag. Once the recurrence events are edited/ deleted as following events, then the following recurrence events will be considered as separate series, the changes will not reflect to parent series. In the following code example, if any of the recurrence event is edited or deleted with the following events option, then the edit or delete action is applied to further recurrence events.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Width="100%" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"
AllowEditFollowingEvents="true"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 3, 10);
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 3, 9, 9, 30, 0) , EndTime = new DateTime(2020, 3, 9, 11, 0,
0), RecurrenceRule = "FREQ=DAILY;INTERVAL=1;COUNT=5" }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}
```



Recurrence options and rules

Events can be repeated on a daily, weekly, monthly or yearly basis based on the recurrence rule which accepts the string value. The following details should be assigned to the `RecurrenceRule` property to generate the recurring instances.

- Repeat type - daily/weekly/monthly/yearly.
- How many times it needs to be repeated?
- The interval duration.
- The time period to render the appointment, etc.

There are four repeat types available namely,

- **Daily** - Creates the recurring instances on daily basis.
- **Weekly** - Creates the recurring instances on weekly basis for the selected days.
- **Monthly** - Creates the recurring instances on monthly basis for the selected months and other provided recurrence criteria.
- **Yearly** - Creates the recurring instances on yearly basis.

Recurrence properties

The properties based on which the recurrence appointments are created with its respective time period are depicted in the following table. Also, the valid rule string can be referred from [iCalendar](#) specifications.

Refer [iCalendar](#) specifications for valid recurrence rule string.

Property	Purpose	Example
FREQ	Maintains the repeat type (Daily, Weekly, Monthly, Yearly) value of the appointment.	FREQ=DAILY;INTERVAL=1

| INTERVAL | Maintains the interval value of the appointments. When the daily appointment is created at an interval of 2, the appointments are rendered on the days Monday, Wednesday and Friday (Creates an appointment on all days by leaving the interval of one day gap). | `FREQ=DAILY;INTERVAL=2` |

| COUNT | It holds the appointment's count value. When the COUNT value is 10, then 10 instances of appointments are created in the recurrence series. | `FREQ=DAILY;INTERVAL=1;COUNT=10` |

| UNTIL | This property holds the end date value (in ISO format) denoting when the recurrence actually ends. | `FREQ=DAILY;INTERVAL=1;UNTIL=20200530T041343Z`; |

| BYDAY | It holds the day value(s), representing on which the appointments actually renders. Create the weekly appointment, and select the day(s) from the day options (Monday/Tuesday/Wednesday/Thursday/Friday/Saturday/Sunday). When Monday is selected, the first two letters of the selected day "MO" is saved in the BYDAY property. When multiple days are selected, the values are separated by commas. | `FREQ=WEEKLY;INTERVAL=1;BYDAY=MO,WE;COUNT=10` |

| BYMONTHDAY | This property is used to store the date value of the Month, while creating the Month recurrence appointment. When a Monthly recurrence appointment is created for every 3rd day of the month, then BYMONTHDAY holds the value 3 and creates the appointment on 3rd day of every month. | `FREQ=MONTHLY;BYMONTHDAY=3;INTERVAL=1;COUNT=10` |

| BYMONTH | This property is used to store the index value of the selected Month while creating the yearly appointments. When the yearly appointment is created on June month, the index value of June month 6 will get stored in the BYMONTH field. The appointment is created on every 6th month of a year. | `FREQ=YEARLY;BYMONTHDAY=16;BYMONTH=6;INTERVAL=1;COUNT=10` |

| BYSETPOS | This property is used to store the index value of the week. When the monthly appointment is created in second week of a month, the index value of the second week (2) is stored in BYSETPOS. | `FREQ=MONTHLY;BYDAY=MO;BYSETPOS=2;COUNT=10` |

The default recurrence related validation has been included for recurrence appointments similar to the one available in Outlook. The validation usually occurs during the recurrence appointment creation, editing, drag and drop or resizing of the recurrence appointments and also if any single occurrence changes.

Daily Frequency

| Description | Example |

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

| Daily recurring event that never ends | `FREQ=DAILY; INTERVAL=1` |

| Daily recurring event that ends after 5 occurrences | `FREQ=DAILY; INTERVAL=1; COUNT=5` |

| Daily recurring event that ends exactly on 12/12/2020 | `FREQ=DAILY; INTERVAL=1; UNTIL=20201212T041343Z` |

| Daily event that recurs on alternative days and repeats for 10 occurrences | `FREQ=DAILY; INTERVAL=2; COUNT=10` |

Weekly Frequency

| Description | Example |

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

| Weekly recurring event that repeats on every Monday, Wednesday and Friday and never ends | `FREQ=WEEKLY; INTERVAL=1; BYDAY=MO,WE,FR` |

| Repeats every week Thursday and ends after 10 occurrences | FREQ=WEEKLY; INTERVAL=1; BYDAY=TH; COUNT=10 |

| Repeats every week Monday and ends on 12/12/2020 | FREQ=WEEKLY; INTERVAL=1; BYDAY=MO; UNTIL=20201212T041343Z |

| Repeats on Monday, Wednesday and Friday of alternative weeks and ends after 10 occurrences | FREQ=WEEKLY; INTERVAL=2; BYDAY=MO, WE, FR; COUNT=10 |

Monthly Frequency

| Description | Example |

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

| Monthly recurring event that repeats on every 15th day of a month and never ends | FREQ=MONTHLY; BYMONTHDAY=15; INTERVAL=1 |

| Monthly recurring event that repeats on every 16th day of a month and ends after 10 occurrences | FREQ=MONTHLY; BYMONTHDAY=16; INTERVAL=1; COUNT=10 |

| Repeats every 17th day of a month and ends on 12/12/2020 | FREQ=MONTHLY; BYMONTHDAY=17; INTERVAL=1; UNTIL=20201212T041343Z |

| Repeats every 2nd Friday of a month and never ends | FREQ=MONTHLY; BYDAY=FR; BYSETPOS=2; INTERVAL=1 |

| Repeats every 4th Wednesday of a month and ends after 10 occurrences | FREQ=MONTHLY; BYDAY=WE; BYSETPOS=4; INTERVAL=1; COUNT=10 |

| Repeats every 4th Friday of a month and ends on 12/12/2020 | FREQ=MONTHLY; BYDAY=FR; BYSETPOS=4; INTERVAL=1; UNTIL=20201212T041343Z; |

Yearly Frequency

| Description | Example |

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

| Yearly event that repeats on every 15th day of December month and never ends | FREQ=YEARLY; BYMONTHDAY=15; BYMONTH=12; INTERVAL=1 |

| Event that repeats on every 10th day of December month and ends after 10 occurrences | FREQ=YEARLY; BYMONTHDAY=10; BYMONTH=12; INTERVAL=1; COUNT=10 |

| Repeats on every 12th day of December month and ends on 12/12/2025 | FREQ=YEARLY; BYMONTHDAY=12; BYMONTH=12; INTERVAL=1; UNTIL=20251212T041343Z |

| Repeats on every 3rd Friday of December month and never ends | FREQ=YEARLY; BYDAY=FR; BYMONTH=12; BYSETPOS=3; INTERVAL=1 |

| Repeats on every 3rd Tuesday of December month and ends after 10 occurrences | FREQ=YEARLY; BYDAY=TU; BYMONTH=12; BYSETPOS=3; INTERVAL=1; COUNT=10 |

| Repeats on every 4th Wednesday of December month and ends on 12/12/2028 | FREQ=YEARLY; BYDAY=WE; BYMONTH=12; BYSETPOS=4; INTERVAL=1; UNTIL=20281212T041343Z |

Recurrence Validation

The built-in validation support has been added by default for recurring appointments during its creation, edit, drag and drop or resize action. The following are the possible validation alerts that displays on Scheduler while creating or editing the recurring events.

Validation messages	Description
----- -----	The recurrence pattern is not valid. This alert will raise, when the selected recurrence rule value is not a valid one. For example, when you try to select the end date value (using Until option) for a recurring event, which occurs before the start date, an alert will popup out saying that the chosen pattern is invalid.
	The changes made to specific instances of this series will be canceled and those events will match the series again. This alert will raise, when you try to edit the whole series, whose occurrence might have been already edited. For example, If there are five occurrences and one of the occurrence is already edited. Now, when you try to edit the entire series, you will get this validation alert.
	The duration of the event must be shorter than how frequently it occurs. Shorten the duration, or change the recurrence pattern in the recurrence event editor. This validation will occur, if the event duration is longer than the selected frequency. For example, if you create a recurring appointment with two days duration in Daily frequency with no intervals set to it, you may get this alert.
	Some months have fewer than the selected date. For these months, the occurrence will fall on the last date of the month. When you try to create a recurring appointment on 31st of every month, where few months won't have 31 days and in this scenario, you will get this alert.
	Two occurrences of the same event cannot occur on the same day. This validation will occur, when you try to edit or move any single occurrence to some other date, where another occurrence of the same event is already present.

Event fields

The Scheduler dataSource usually holds the event instances, where each of the instance includes a collection of appropriate [fields](#). It is mandatory to map these fields with the equivalent fields of database, when remote data is bound to it. When the local data is bound, then the field names defined within the instances needs to be mapped with the scheduler event fields correctly.

To create an event on Scheduler, it is enough to define the **StartTime** and **EndTime** fields. In case, if remote data is bound to Scheduler, then **Id** field becomes mandatory to process the CRUD actions on appropriate events.

Built-in fields

The built-in fields available on Scheduler event object are as follows.

Field name	Description
----- -----	Id The Id field needs to be defined as mandatory, and usually assigns a unique ID value to each of the events.
	Subject The Subject field is optional, and usually assigns the summary text to each of the events.

| **StartTime** | The **StartTime** field defines the start time of an event and it is mandatory to provide it for any of the valid event objects. |

| **EndTime** | The **EndTime** field defines the end time of an event and it is mandatory to provide the end time for any of the valid event objects. |

| **StartTimeZone** | It maps the **StartTimeZone** field from the dataSource and usually accepts the valid IANA timezone names. It is assumed that the value provided for this field is taken into consideration while processing the **StartTime** field. When this field is not mapped with any timezone names, then the events will be processed based on the timezone assigned to the Scheduler. |

| **EndTimeZone** | It maps the **EndTimeZone** field from the dataSource and usually accepts the valid IANA timezone names. It is assumed that the value provided for this field is taken into consideration while processing the **EndTime** field. When this field is not mapped with any timezone names, then the events will be processed based on the timezone assigned to the Scheduler. |

| **Location** | It maps the **Location** field from the dataSource and the location text value will be displayed over the events. |

| **Description** | It maps the **Description** field from the dataSource and denotes the event description which is optional. |

| **IsAllDay** | The **IsAllDay** field is mapped from the dataSource and is used to denote whether an event is created for an entire day or for specific time alone. Usually, an event with **IsAllDay** field set to true will be considered as an all-day event. |

| **RecurrenceID** | It maps the **RecurrenceID** field from dataSource and usually holds the ID value of the parent recurrence event. This field is applicable only for the edited occurrence events. |

| **RecurrenceRule** | It maps the **RecurrenceRule** field from dataSource and holds the recurrence rule value in a string format. Also, it uniquely identifies whether the event belongs to a recurring type or normal ones. |

| **RecurrenceException** | It maps the **RecurrenceException** field from dataSource and is used to hold the collection of exception dates, on which the recurring occurrences needs to be excluded. |

| **IsReadOnly** | It maps the **IsReadOnly** field from dataSource. It is mainly used to make specific appointments as readonly when set to true. |

| **IsBlock** | It maps the **IsBlock** field from dataSource. It is used to block the particular time ranges in the Scheduler and prevents the event creation on those time slots. |

| **CssClass** | It maps the **CssClass** field from the dataSource. It is used to customize the particular events. |

Binding different field names

When the fields of event instances has the default mapping name, it is not mandatory to map them manually. If a Scheduler's dataSource holds the events collection with different field names, then it is necessary to map them with its equivalent field name within the **EventSettings** property.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
```

```

<ScheduleEventSettings DataSource="@DataSource">
  <ScheduleField Id="TravelId">
    <FieldSubject Name="TravelSummary"></FieldSubject>
    <FieldLocation Name="Source"></FieldLocation>
    <FieldDescription Name="Comments"></FieldDescription>
    <FieldIsAllDay Name="FullDay"></FieldIsAllDay>
    <FieldStartTime Name="DepartureTime"></FieldStartTime>
    <FieldEndTime Name="ArrivalTime"></FieldEndTime>
    <FieldStartTimestzone Name="Origin"></FieldStartTimestzone>
    <FieldEndTimestzone Name="Destination"></FieldEndTimestzone>
  </ScheduleField>
</ScheduleEventSettings>
<ScheduleViews>
  <ScheduleView Option="View.Day"></ScheduleView>
  <ScheduleView Option="View.Week"></ScheduleView>
  <ScheduleView Option="View.WorkWeek"></ScheduleView>
  <ScheduleView Option="View.Month"></ScheduleView>
  <ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
  DateTime CurrentDate = new DateTime(2020, 1, 10);
  List<AppointmentData> DataSource = new List<AppointmentData>
  {
    new AppointmentData { TravelId = 1, TravelSummary = "Paris", DepartureTime =
    new DateTime(2020, 1, 10, 10, 0, 0) , ArrivalTime = new DateTime(2020, 1,
    10, 12, 30, 0),
    Source = "London", Comments = "Summer vacation planned for outstation.",
    Origin= "Asia/Yekaterinburg", Destination= "Asia/Yekaterinburg" }
  };
  public class AppointmentData
  {
    public int TravelId { get; set; }
    public string TravelSummary { get; set; }
    public DateTime DepartureTime { get; set; }
    public DateTime ArrivalTime { get; set; }
    public bool FullDay { get; set; }
    public string Source { get; set; }
    public string Comments { get; set; }
    public string Origin { get; set; }
    public string Destination { get; set; }
  }
}

```

The mapper field `Id` is of string type and has no additional validation options, whereas all other fields has additional options.

Event field settings

Each field of the Scheduler events are provided with additional settings such as options to set default value, to map with appropriate data source fields, to validate every event fields and to provide label values for those fields in the event window.

| Options | Description |

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

| Default | Accepts the default value to the applicable fields (Subject, Location and Description), when no values are provided to them from dataSource. |

| Name | Accepts the field name to be mapped from the dataSource fields. |

| Title | Accepts the label values to be displayed for the fields of event editor. |

| Validation | Defines the validation rules to be applied on the event fields within the event editor. |

In following example, the Subject field in event editor will display its appropriate label as **Summary**. When no subject value is provided while saving an event, then the appointment will be saved with the default subject value as **Add Summary**.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEventSettings TValue="AppointmentData" DataSource="@DataSource">
<ScheduleField Id="Id">
<FieldSubject Name="Subject" Title="Summary" Default="Add
Summary"></FieldSubject>
<FieldLocation Name="Location"></FieldLocation>
<FieldDescription Name="Description"></FieldDescription>
<FieldStartTime Name="StartTime"></FieldStartTime>
<FieldEndTime Name="EndTime"></FieldEndTime>
</ScheduleField>
</ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 31);
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}
```


Adding Custom fields

Apart from the default Scheduler fields, the user can include 'n' number of custom fields for appointments. The following code example shows how to include two custom fields namely **Status** and **Priority** within event collection. It is not necessary to bind the custom fields within the **EventSettings**. However, those additional fields can be accessed easily, for internal processing as well as from application end.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource">
</ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 31);
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
0),
Status = "Completed", Priority = "High"}
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
public string Status { get; set; }
public string Priority { get; set; }
}
}
```

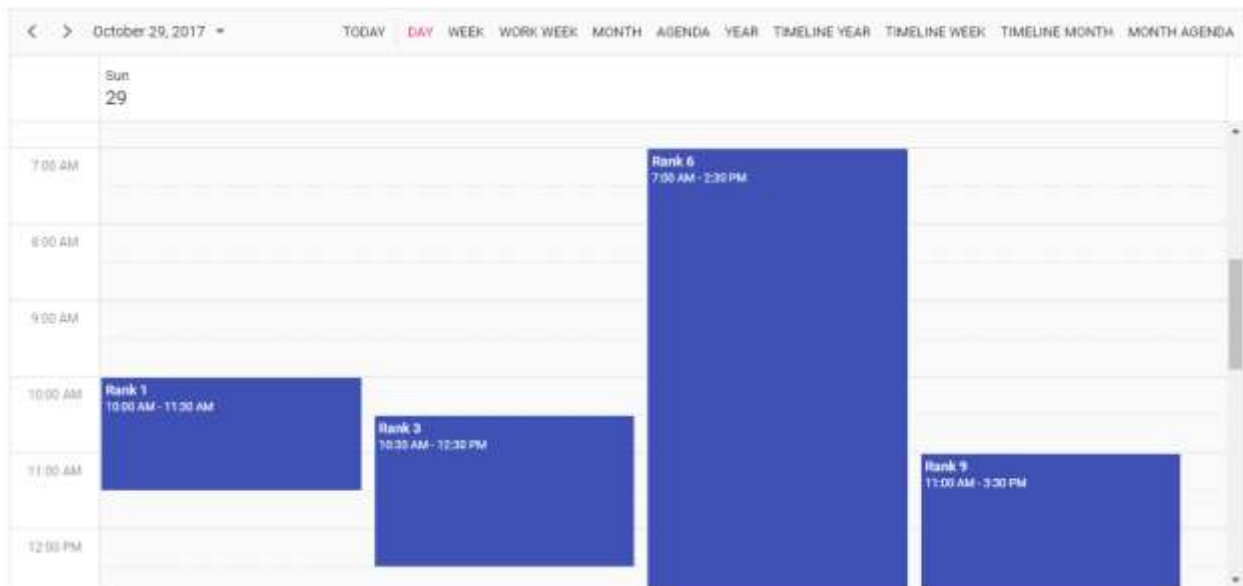
Customize the order of the overlapping events

By default, the scheduler will render the overlapping events based on the start and end time. Now, the order of the overlapping events can be customized based on the custom fields by using the **SortBy**

property grouped under the `EventSettings` property. The following code example shows how to sort the appointments based on the custom field **RankId** as follows.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentSortData" Width="100%"
EnableAutoRowHeight="true" @bind-SelectedDate="@CurrentDate" @bind-
CurrentView="@SelectedView">
<ScheduleEventSettings SortBy="RankId"
DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.TimelineDay"></ScheduleView>
<ScheduleView Option="View.TimelineWeek"></ScheduleView>
<ScheduleView Option="View.TimelineWorkWeek"></ScheduleView>
<ScheduleView Option="View.TimelineMonth"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 2, 14);
View SelectedView = View.Week;
DataSource = new List<AppointmentSortData>
{
new AppointmentSortData { Id = 1, Subject = "Rank A", RankId="A", StartTime
= new DateTime(2020, 2, 13, 10, 0, 0) , EndTime = new DateTime(2020, 2, 13,
12, 0, 0) },
new AppointmentSortData { Id = 2, Subject = "Rank B", RankId="B", StartTime
= new DateTime(2020, 2, 13, 7, 0, 0) , EndTime = new DateTime(2020, 2, 13,
15, 0, 0) },
new AppointmentSortData { Id = 3, Subject = "Rank C", RankId="C", StartTime
= new DateTime(2020, 2, 13, 9, 0, 0) , EndTime = new DateTime(2020, 2, 13,
10, 30, 0) },
new AppointmentSortData { Id = 4, Subject = "Rank D", RankId="D", StartTime
= new DateTime(2020, 2, 13, 9, 30, 0) , EndTime = new DateTime(2020, 2, 13,
14, 0, 0) }
};
public class AppointmentSortData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
public string RankId { get; set; }
}
}
```



Drag and drop appointments

Appointments can be rescheduled to any time by dragging and dropping them onto the desired location. To work with drag and drop functionality make sure that `AllowDragAndDrop` is set to **true** on Scheduler. In mobile mode, you can drag and drop the events by tap holding an event and dropping them on to the desired location.

By default, drag and drop action is applicable on all Scheduler views, except Agenda and Month-Agenda view.

To get start quickly about drag options available in our Scheduler, you can check on this video:

{% youtube

"youtube:https://www.youtube.com/watch?v=Vtl1Wyuwt-0"%}

Drag and drop multiple appointments

Multiple appointments can be dragged and dropped by enabling the `AllowMultiDrag` property. Multiple appointments can be selected by holding the CTRL key. Once the events are selected, leave the CTRL key and start dragging the event.

Multiple events can also be dragged from one resource to another resource. In this case, if all the selected events are in the different resources, then all the events should be moved to the single resource that is related to the target event.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" AllowMultiDrag="true"
@bind-SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
```

```

</SfSchedule>
@code{
    DateTime CurrentDate = new DateTime(2020, 1, 31);
    List<AppointmentData> DataSource = new List<AppointmentData>
    {
        new AppointmentData{ Id = 1, Subject = "Meeting", StartTime = new
        DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
        0)},
        new AppointmentData{ Id = 2, Subject = "Testing", StartTime = new
        DateTime(2020, 1, 31, 12, 0, 0) , EndTime = new DateTime(2020, 1, 31, 13, 0,
        0)}
    };
    public class AppointmentData
    {
        public int Id { get; set; }
        public string Subject { get; set; }
        public string Location { get; set; }
        public DateTime StartTime { get; set; }
        public DateTime EndTime { get; set; }
        public string Description { get; set; }
        public bool IsAllDay { get; set; }
        public string RecurrenceRule { get; set; }
        public string RecurrenceException { get; set; }
        public Nullable<int> RecurrenceID { get; set; }
    }
}

```

Disable the drag action

By default, the events can be dragged and dropped within any of the applicable scheduler views, and to disable it, set **false** to the **AllowDragAndDrop** property.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" AllowDragAndDrop="false"
@bind-SelectedDate="@CurrentDate">
    <ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
    <ScheduleViews>
        <ScheduleView Option="View.Day"></ScheduleView>
        <ScheduleView Option="View.Week"></ScheduleView>
        <ScheduleView Option="View.WorkWeek"></ScheduleView>
        <ScheduleView Option="View.Month"></ScheduleView>
        <ScheduleView Option="View.Agenda"></ScheduleView>
    </ScheduleViews>
</SfSchedule>
@code{
    DateTime CurrentDate = new DateTime(2020, 1, 31);
    List<AppointmentData> DataSource = new List<AppointmentData>
    {
        new AppointmentData{ Id = 1, Subject = "Meeting", StartTime = new
        DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
        0)}
    };
    public class AppointmentData
    {
        public int Id { get; set; }

```

```

public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

Preventing drag and drop on specific targets

It is possible to prevent the drag action on particular target, by passing the target to be excluded in the `ExcludeSelectors` option within `OnDragStart` event.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEvents TValue="AppointmentData"
OnDragStart="OnAppointmentDrag"></ScheduleEvents>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 31);
public void OnAppointmentDrag(DragEventArgs<AppointmentData> args)
{
args.ExcludeSelectors = "e-header-cells,e-all-day-cells";
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData{ Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
0)}
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

```
}
}
```

Disable scrolling on drag action

By default, while dragging an appointment to the edges, either top/bottom in the vertical Scheduler or left/right in the timeline Scheduler, scrolling action takes place automatically. To prevent this scrolling, set `false` to the `Scroll` value within the `OnDragStart` event arguments.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px"
SelectedDate="@CurrentDate">
<ScheduleEvents TValue="AppointmentData"
OnDragStart="OnAppointmentDrag"></ScheduleEvents>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 31);
public void OnAppointmentDrag(DragEventArgs<AppointmentData> args)
{
args.Scroll.Enable = false;
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData{ Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
0)}
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}
```

Controlling scroll speed while dragging an event

The speed of the scrolling action while dragging an appointment to the Scheduler edges can be controlled within the `OnDragStart` event by setting the desired value to the `ScrollBy` and `TimeDelay` option, whereas its default value is 30 minutes and 100ms.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEvents TValue="AppointmentData"
OnDragStart="OnAppointmentDrag"></ScheduleEvents>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 31);
public void OnAppointmentDrag(DragEventArgs<AppointmentData> args)
{
args.Scroll.ScrollBy = 5;
args.Scroll.TimeDelay = 200;
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData{ Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
0)}
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}
```

Auto navigation of date ranges on dragging an event

When an event is dragged either to the left or right extreme edges of the Scheduler and kept hold for few seconds without dropping, the auto navigation of date ranges will be enabled allowing the Scheduler to navigate from current date range to back and forth respectively. This action is set to `false` by default and to enable it, set `Navigation` to true within the `OnDragStart` event.

By default, the navigation delay is set to 2000ms. The navigation delay decides how long the user needs to drag and hold the appointments at the extremities. You can also set your own delay value for letting the users to navigate based on it, using the `TimeDelay` within the `OnDragStart` event.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEvents TValue="AppointmentData"
OnDragStart="OnAppointmentDrag"></ScheduleEvents>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 31);
public void OnAppointmentDrag(DragEventArgs<AppointmentData> args)
{
args.Navigation.Enable = true;
args.Navigation.TimeDelay = 4000;
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData{ Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
0)}
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}
```

Setting drag time interval

By default, while dragging an appointment, it moves at an interval of 30 minutes. To change the dragging time interval, pass the appropriate values to the `Interval` option within the `OnDragStart` event.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
```



```

<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
  <ScheduleEvents TValue="AppointmentData"
  OnDragStart="OnAppointmentDrag"></ScheduleEvents>
  <ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
  <ScheduleViews>
    <ScheduleView Option="View.Day"></ScheduleView>
    <ScheduleView Option="View.Week"></ScheduleView>
    <ScheduleView Option="View.WorkWeek"></ScheduleView>
    <ScheduleView Option="View.Month"></ScheduleView>
    <ScheduleView Option="View.Agenda"></ScheduleView>
  </ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 31);
public void OnAppointmentDrag(DragEventArgs<AppointmentData> args)
{
args.Interval = 10;
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData{ Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
0)}
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

Drag and drop items from external source

It is possible to drag and drop the unplanned items from any of the external source into the scheduler, by manually saving those dropped item as a new appointment data through `AddEventAsync` method of Scheduler.

To get start quickly about dropping items from external source to our Scheduler, you can check on this video:

{% youtube

"youtube:https://www.youtube.com/watch?v=QxBBZYef6cg"%}

In this example, the tree view control is used as an external source and the child nodes from the tree view component are dragged and dropped onto the Scheduler. Therefore, it is necessary to make use of

the `OnNodeDragStop` event of the `TreeView` component, where an event object can be formed and save it using the `AddEventAsync` method.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Navigations
<div class="row">
<div class="col-lg-8 e-droppable">
<SfSchedule @ref="ScheduleRef" TValue="AppointmentData" Height="650px"
@bind-SelectedDate="@CurrentDate">
<ScheduleGroup Resources="@GroupData"></ScheduleGroup>
<ScheduleResources>
<ScheduleResource TItem="ResourceData" TValue="int"
DataSource="@Consultants" Field="ConsultantID" Title="Consultant"
Name="Consultants" TextField="Text" IdField="Id"
ColorField="Color"></ScheduleResource>
</ScheduleResources>
<ScheduleEventSettings DataSource="@DataSource">
</ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
</div>
<div class="col-lg-4">
<h3>Waiting list</h3>
<SfTreeView TValue="EmployeeData" AllowDragAndDrop="true">
<TreeViewFieldsSettings DataSource="@WaitingListData" Id="Id"
Text="Name"></TreeViewFieldsSettings>
<TreeViewEvents TValue="EmployeeData"
OnNodeDragStop="DragStop"></TreeViewEvents>
</SfTreeView>
</div>
</div>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 31);
SfSchedule<AppointmentData> ScheduleRef;
public string[] GroupData = new string[] { "Consultants" };
public List<ResourceData> Consultants { get; set; } = new List<ResourceData>
{
new ResourceData { Text = "Margaret", Id = 1, Color = "#1aaa55" },
new ResourceData { Text = "Robert", Id = 2, Color = "#357cd2" },
new ResourceData { Text = "Laura", Id = 3, Color = "#7fa900" },
new ResourceData { Text = "Robson", Id = 4, Color = "#9e5fff" },
new ResourceData { Text = "Laura", Id = 5, Color = "#bbdc00" }
};
public List<EmployeeData> WaitingListData { get; set; } = new
List<EmployeeData>() {
new EmployeeData { Id = 1, Name = "Johnson" },
new EmployeeData { Id = 2, Name = "Sourav" },
new EmployeeData { Id = 3, Name = "Sanjay" }
};
};
```

```

List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData{ Id = 1, Subject = "General-Check up", StartTime = new
    DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
    0), ConsultantID=1 }
};
async void DragStop(DragAndDropEventArgs args)
{
    args.Cancel = true;
    CellClickEventArgs cellData = await
    ScheduleRef.GetTargetCellAsync((int)args.Left, (int)args.Top);
    if (cellData != null)
    {
        var resourceDetails = ScheduleRef.GetResourceByIndex(cellData.GroupIndex);
        Random rnd = new Random();
        AppointmentData eventData = new AppointmentData
        {
            Id = rnd.Next(1000),
            Subject = args.DraggedNodeData.Text,
            StartTime = cellData.StartTime,
            EndTime = cellData.EndTime,
            IsAllDay = cellData.IsAllDay,
            ConsultantID = resourceDetails.GroupData.ConsultantID,
        };
        await ScheduleRef.OpenEditorAsync(eventData, CurrentAction.Add);
    }
}
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string RecurrenceRule { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
    public string RecurrenceException { get; set; }
    public bool? IsAllDay { get; set; }
    public Nullable<int> ConsultantID { get; set; }
}
public class EmployeeData
{
    public int Id { get; set; }
    public string Name { get; set; }
}
public class ResourceData
{
    public int Id { get; set; }
    public string Text { get; set; }
    public string Color { get; set; }
}
}

```

Drag and drop items to external source

You can drag and drop the events to external source by setting the target to the property **EventDragArea**. In the following code example, we have two Scheduler and events from the first

scheduler that can be dropped to the second scheduler. In the **Dragged** event of the first scheduler, the dragged event has been deleted from the first scheduler and added to the second scheduler.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<div class="row">
<div class="col-lg-6">
<SfSchedule @ref="Schedule1Ref" Height="550px" TValue="AppointmentData"
EventDragArea=".ScheduleClass" @bind-SelectedDate="@CurrentDate">
<ScheduleEvents TValue="AppointmentData"
Dragged="OnDragged"></ScheduleEvents>
<ScheduleEventSettings DataSource="@ScheduleData"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
</div>
<div class="col-lg-6">
<SfSchedule @ref="Schedule2Ref" Height="550px" TValue="AppointmentData"
CssClass="ScheduleClass" SelectedDate="@ (new DateTime(2020, 1, 6))">
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
</div>
</div>
@code {
DateTime CurrentDate = new DateTime(2020, 1, 6);
SfSchedule<AppointmentData> Schedule1Ref;
SfSchedule<AppointmentData> Schedule2Ref;
List<AppointmentData> ScheduleData = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 6, 9, 30, 0) , EndTime = new DateTime(2020, 1, 6, 11, 0,
0) }
};
public async Task OnDragged(DragEventArgs<AppointmentData> args)
{
await Schedule1Ref.DeleteEventAsync(args.Data.Id);
Random random = new Random();
args.Data.Id = Convert.ToInt32(random.Next());
await Schedule2Ref.AddEventAsync(args.Data);
}
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
}
```

```

public string Description { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public Nullable<bool> IsAllDay { get; set; }
public bool IsReadOnly { get; set; }
public string RecurrenceRule { get; set; }
public Nullable<int> RecurrenceID { get; set; }
public string RecurrenceException { get; set; }
public string StartTimezone { get; set; }
public string EndTimezone { get; set; }
}
}

```

Opening the editor window on drag stop

There are scenarios where you want to open the editor filled with data on newly dropped location and may need to proceed to save it, only when Save button is clicked on the editor. Clicking on the cancel button should revert these changes. This can be achieved using the **Dragged** event of Scheduler.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" @ref="ScheduleRef" Height="550px"
@bind-SelectedDate="@CurrentDate">
<ScheduleEvents TValue="AppointmentData"
Dragged="OnAppointmentDragStop"></ScheduleEvents>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 31);
SfSchedule<AppointmentData> ScheduleRef;
public async Task OnAppointmentDragStop(DragEventArgs<AppointmentData> args)
{
args.Cancel = true;
await this.ScheduleRef.OpenEditorAsync(args.Data, CurrentAction.Save);
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData{ Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
0)}
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
}
}

```

```
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}
```

Appointment Resizing

Another way of rescheduling an appointment can be done by resizing it through either of its handlers. To work with resizing functionality make sure that `AllowResizing` property is set to `true`.

To get start quickly about resize options available in our Scheduler, you can check on this video:

{% youtube

"youtube:https://www.youtube.com/watch?v=Vtl1Wyuwt-0"%}

Disable the resize action

By default, resizing of events is allowed on all Scheduler views except Agenda and Month-Agenda view. To disable this event resizing action, set false to the `AllowResizing` property.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" AllowResizing="false"
@bind-SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 31);
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
0)}
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
```

```
}

```

Disable scrolling on resize action

By default, while resizing an appointment, when its handler reaches the extreme edges of the Scheduler, scrolling action will takes place along with event resizing. To prevent this scrolling action, set false to `Scroll` value within the `OnResizeStart` event.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEvents TValue="AppointmentData"
OnResizeStart="OnAppointmentResize"></ScheduleEvents>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 31);
public void OnAppointmentResize(ResizeEventArgs<AppointmentData> args)
{
args.Scroll.Enable = false;
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}
```

Controlling scroll speed while resizing an event

The speed of the scrolling action while resizing an appointment to the Scheduler edges, can be controlled within the `OnResizeStart` event by setting the desired value to the `ScrollBy` option.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEvents TValue="AppointmentData"
OnResizeStart="OnAppointmentResize"></ScheduleEvents>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 31);
public void OnAppointmentResize(ResizeEventArgs<AppointmentData> args)
{
args.Scroll.ScrollBy = 15;
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

Setting resize time interval

By default, while resizing an appointment, it extends or shrinks at an interval of 30 minutes. To change this default resize interval, set appropriate values to `Interval` option within the `OnResizeStart` event.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEvents TValue="AppointmentData"
OnResizeStart="OnAppointmentResize"></ScheduleEvents>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>

```



```

<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
    DateTime CurrentDate = new DateTime(2020, 1, 31);
    public void OnAppointmentResize(ResizeEventArgs<AppointmentData> args)
    {
        args.Interval = 10;
    }
    List<AppointmentData> DataSource = new List<AppointmentData>
    {
        new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
        DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
        0) }
    };
    public class AppointmentData
    {
        public int Id { get; set; }
        public string Subject { get; set; }
        public string Location { get; set; }
        public DateTime StartTime { get; set; }
        public DateTime EndTime { get; set; }
        public string Description { get; set; }
        public bool IsAllDay { get; set; }
        public string RecurrenceRule { get; set; }
        public string RecurrenceException { get; set; }
        public Nullable<int> RecurrenceID { get; set; }
    }
}

```

Appointment customization

The look and feel of the Scheduler events can be customized using any one of the following ways.

- [Using event template](#)
- [Using EventRendered event](#)
- [Using CssClass property](#)

Using template

Any kind of text, images and links can be added to customize the look of the events. The user can format and change the default appearance of the events by making use of the **Template** option available within the **ScheduleEventSettings** tag helper.

To get start quickly on customizing events using template, you can check on this video:

{% youtube

"youtube:https://www.youtube.com/watch?v=8kBXcBjL12A"%}

The following code example customizes the appointment.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
```

```

<SfSchedule TValue="AppointmentData" Height="650px" @bind-
SelectedDate="@CurrentDate">
  <ScheduleEventSettings DataSource="@DataSource">
    <Template>
      <div>Subject: @((context as AppointmentData).Subject)</div>
      <div>StartTime: @((context as AppointmentData).StartTime)</div>
      <div>EndTime: @((context as AppointmentData).EndTime)</div>
    </Template>
  </ScheduleEventSettings>
  <ScheduleViews>
    <ScheduleView Option="View.Day"></ScheduleView>
    <ScheduleView Option="View.Week"></ScheduleView>
    <ScheduleView Option="View.WorkWeek"></ScheduleView>
    <ScheduleView Option="View.Month"></ScheduleView>
    <ScheduleView Option="View.Agenda"></ScheduleView>
  </ScheduleViews>
</SfSchedule>
@code {
  DateTime CurrentDate = new DateTime(2020, 1, 31);
  List<AppointmentData> DataSource = new List<AppointmentData>
  {
    new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
    DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
    0) }
  };
  public class AppointmentData
  {
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
  }
}

```

All the built-in fields that are mapped to the appropriate field properties within the `ScheduleEventSettings`, as well as custom mapped fields from the Scheduler `DataSource` can be accessed within the template code.

Using `EventRendered` event

The `EventRendered` event triggers before the appointment renders on the Scheduler. Therefore, this event can be utilized to customize the look of events based on any specific criteria, before rendering them on the scheduler.

In the following code example, the custom class has been added to events using `CssClasses` to apply color to the events.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
```

```

<SfSchedule TValue="AppointmentData" Height="650px" @bind-
SelectedDate="@CurrentDate">
  <ScheduleEvents TValue="AppointmentData"
  EventRendered="OnEventRendered"></ScheduleEvents>
  <ScheduleViews>
    <ScheduleView Option="View.Day"></ScheduleView>
    <ScheduleView Option="View.Week"></ScheduleView>
    <ScheduleView Option="View.Month"></ScheduleView>
  </ScheduleViews>
  <ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
</SfSchedule>
@code {
  DateTime CurrentDate = new DateTime(2020, 1, 31);
  public List<string> CustomClass = new List<string>() { "custom-class" };
  public void OnEventRendered(EventRenderedArgs<AppointmentData> args)
  {
    args.CssClasses = CustomClass;
  }
  List<AppointmentData> DataSource = new List<AppointmentData>
  {
    new AppointmentData{ Id = 1, Subject = "Meeting", StartTime = new
    DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
    0) }
  };
  public class AppointmentData
  {
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
  }
}
<style>
.e-schedule .e-vertical-view .e-all-day-appointment-wrapper .e-
appointment.custom-class,
.e-schedule .e-vertical-view.e-day-wrapper .e-appointment.custom-class,
.e-schedule .e-month-view .e-appointment.custom-class {
background: #32CD32;
}
</style>

```

Also, we can customize the events by adding or modifying its element attribute using **Attributes**. In the following code example, event attributes have been modified through the **Attributes** to apply color to the events.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="650px" @bind-
SelectedDate="@CurrentDate">

```

```

<ScheduleEvents TValue="AppointmentData"
EventRendered="OnEventRendered"></ScheduleEvents>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
</ScheduleViews>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
</SfSchedule>
@code {
DateTime CurrentDate = new DateTime(2020, 1, 31);
public List<string> CustomClass = new List<string>() { "custom-class" };
public void OnEventRendered(EventRenderedArgs<AppointmentData> args)
{
Dictionary<string, object> attributes = new Dictionary<string, object>();
attributes.Add("style", "background: green");
args.Attributes = attributes;
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData{ Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

Using CssClass

The customization of events can also be achieved using the built-in field `CssClass` in which you can pass the class name to be applied to specific appointments. In the following example, the background of appointments has been changed.

ASPX-CS

```

@using Syncfusion.Blazor.Scheduler
@using Syncfusion.Blazor.Scheduler
<SfSchedule TValue="AppointmentData" Height="650px" @bind-
SelectedDate="@CurrentDate">
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
</ScheduleViews>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>

```

```

</SfSchedule>
@code {
    DateTime CurrentDate = new DateTime(2020, 1, 31);
    List<AppointmentData> DataSource = new List<AppointmentData>
    {
        new AppointmentData{ Id = 1, Subject = "Meeting", StartTime = new
        DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
        0), CssClass = "progress" },
        new AppointmentData{ Id = 2, Subject = "Meeting-postponed", StartTime = new
        DateTime(2020, 1, 28, 9, 30, 0) , EndTime = new DateTime(2020, 1, 28, 11, 0,
        0), CssClass = "delayed" }
    };
    public class AppointmentData
    {
        public int Id { get; set; }
        public string Subject { get; set; }
        public string Location { get; set; }
        public DateTime StartTime { get; set; }
        public DateTime EndTime { get; set; }
        public string Description { get; set; }
        public string CssClass { get; set; }
        public bool IsAllDay { get; set; }
        public string RecurrenceRule { get; set; }
        public string RecurrenceException { get; set; }
        public Nullable<int> RecurrenceID { get; set; }
    }
}

<style>
.e-schedule .e-vertical-view .e-all-day-appointment-wrapper .e-
appointment.progress,
.e-schedule .e-vertical-view .e-day-wrapper .e-appointment.progress,
.e-schedule .e-month-view .e-appointment.progress {
background: #32CD32;
}
.e-schedule .e-vertical-view .e-all-day-appointment-wrapper .e-
appointment.delayed,
.e-schedule .e-vertical-view .e-day-wrapper .e-appointment.delayed,
.e-schedule .e-month-view .e-appointment.delayed {
background: #CD5C5C;
}
</style>

```

Also, the customization of events can be achieved using **CssClass** property of the Scheduler. In the following example, the background of appointments has been changed using the **CssClass**.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" CssClass="custom-class"
@bind-SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>

```

```

<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 31);
List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
    DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
    0) }
};
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
}
}
<style>
.custom-class.e-schedule .e-vertical-view .e-all-day-appointment-wrapper .e-
appointment,
.custom-class.e-schedule .e-vertical-view .e-day-wrapper .e-appointment,
.custom-class.e-schedule .e-month-view .e-appointment {
background: #32CD32;
}
</style>

```

The events can't be customized using the styles that are `height`, `width`, `top`, `left`, `right`, and `display`.

Block Date and Time

It is possible to block a set of dates or a particular time ranges on the Scheduler. To do so, define an appointment object within `EventSettings` along with the required time range to block and set the `IsBlock` field to **true**. Usually, the event objects defined with `IsBlock` field set to true will block the entire time cells lying within the appropriate time ranges specified through `StartTime` and `EndTime` fields.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>

```

```

</SfSchedule>
@code{
    DateTime CurrentDate = new DateTime(2020, 1, 31);
    List<AppointmentData> DataSource = new List<AppointmentData>
    {
        new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
        DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
        0),
        IsBlock = true }
    };
    public class AppointmentData
    {
        public int Id { get; set; }
        public string Subject { get; set; }
        public string Location { get; set; }
        public DateTime StartTime { get; set; }
        public DateTime EndTime { get; set; }
        public string Description { get; set; }
        public bool IsAllDay { get; set; }
        public bool IsBlock { get; set; }
        public string RecurrenceRule { get; set; }
        public string RecurrenceException { get; set; }
        public Nullable<int> RecurrenceID { get; set; }
    }
}

```

Block events can also be defined to repeat on several days as shown in the following code example.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
    DateTime CurrentDate = new DateTime(2020, 1, 31);
    List<AppointmentData> DataSource = new List<AppointmentData>
    {
        new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
        DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
        0),
        IsBlock = true, RecurrenceRule = "FREQ=DAILY;INTERVAL=1;COUNT=5" }
    };
    public class AppointmentData
    {
        public int Id { get; set; }
        public string Subject { get; set; }
        public string Location { get; set; }
    }
}

```

```

public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public bool IsBlock { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}

```

Readonly

An interaction with the appointments of Scheduler can be enabled/disabled using the **Readonly** property. With this property enabled, you can simply navigate between the Scheduler dates, views and can be able to view the appointment details in the quick info window. Most importantly, the users are not allowed to perform any CRUD actions on Scheduler, when this property is set to true. By default, it is set as **false**.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" Readonly="true" @bind-
SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 31);
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```


Make specific events readonly

There are scenarios where you need to restrict the CRUD action on specific appointments alone based on certain conditions. In the following example, the events that has occurred on the past hours from the current date of the Scheduler are made as read-only and the CRUD actions has been prevented only on those appointments. This can be achieved by setting `IsReadOnly` field of read-only events to `true`.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="DataSource"> </ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 31);
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Paris", StartTime = new
DateTime(2020, 1, 28, 10, 0, 0) , EndTime = new DateTime(2020, 1, 28, 12, 0,
0),
IsReadOnly = true },
new AppointmentData { Id = 2, Subject = "Germany", StartTime = new
DateTime(2020, 1, 31, 10, 0, 0) , EndTime = new DateTime(2020, 1, 31, 12, 0,
0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public bool IsReadOnly { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}
```

By default, the event editor is prevented to open on the read-only events when `IsReadOnly` field is set to `true`.

Differentiate the past time events

To differentiate the appearance of the appointments based on specific criteria such as displaying the past hour appointments with different colors on Scheduler, `EventRendered` event can be used which triggers before the appointment renders on the Scheduler.

In the following code example, the appointments beyond current date of the scheduler were differentiated with chocolate brown color.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Width="100%" Height="550px" @bind-
SelectedDate="@SelectedDate">
  <ScheduleViews>
    <ScheduleView Option="View.Day"></ScheduleView>
    <ScheduleView Option="View.Week"></ScheduleView>
    <ScheduleView Option="View.WorkWeek"></ScheduleView>
    <ScheduleView Option="View.Month"></ScheduleView>
  </ScheduleViews>
  <ScheduleEvents TValue="AppointmentData"
  EventRendered="OnEventRendered"></ScheduleEvents>
  <ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
</SfSchedule>
@code{
public DateTime SelectedDate = new DateTime(2020,1,10);
public List<string> CustomClass = new List<string>() { "e-past-app" };
public void OnEventRendered(EventRenderedArgs<AppointmentData> args)
{
  if(args.Data.StartTime < SelectedDate)
  {
    args.CssClasses = CustomClass;
  }
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
  new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
  DateTime(2020, 1, 10, 9, 30, 0) , EndTime = new DateTime(2020, 1, 10, 11, 0,
  0) },
  new AppointmentData { Id = 1, Subject = "Conference", StartTime = new
  DateTime(2020, 1, 9, 11, 30, 0) , EndTime = new DateTime(2020, 1, 9, 13, 0,
  0) }
};
public class AppointmentData
{
  public int Id { get; set; }
  public string Subject { get; set; }
  public DateTime StartTime { get; set; }
  public DateTime EndTime { get; set; }
}
}
<style>
.e-schedule .e-vertical-view .e-day-wrapper .e-appointment.e-past-app, .e-
schedule .e-month-view .e-appointment.e-past-app{
background-color: chocolate;
}
</style>
```

Appointments occupying entire cell

The Scheduler allows the event to occupy the full height of the cell without its header part by setting `true` for `EnableMaxHeight` Property.

More indicator can be shown if more than one appointment is available in a same cell by setting `true` to `EnableIndicator` property whereas its default value is false.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Width="100%" Height="550px"
SelectedDate="@ (new DateTime(2020, 3, 11))" CurrentView="View.Month">
<ScheduleViews>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.TimelineWeek"></ScheduleView>
<ScheduleView Option="View.TimelineMonth"></ScheduleView>
</ScheduleViews>
<ScheduleEventSettings DataSource="@DataSource" EnableMaxHeight="true"
EnableIndicator="true"></ScheduleEventSettings>
</SfSchedule>
@code{
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 3, 11, 9, 30, 0) , EndTime = new DateTime(2020, 3, 11, 11, 0,
0) },
new AppointmentData { Id = 2, Subject = "Conference", StartTime = new
DateTime(2020, 3, 11, 9, 30, 0) , EndTime = new DateTime(2020, 3, 11, 11, 0,
0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}
```

The `EnableIndicator` property will work, only when the `EnableMaxHeight` property value is set to true.

Display tooltip for appointments

The tooltip shows the Scheduler appointment's information in a formatted style by making use of the tooltip related options.

Show or hide built-in tooltip

The tooltip can be displayed for appointments by setting `true` to the `EnableTooltip` option within the `ScheduleEventSettings` tag helper.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
  <ScheduleEventSettings DataSource="@DataSource"
  EnableTooltip="true"></ScheduleEventSettings>
  <ScheduleViews>
    <ScheduleView Option="View.Day"></ScheduleView>
    <ScheduleView Option="View.Week"></ScheduleView>
    <ScheduleView Option="View.WorkWeek"></ScheduleView>
    <ScheduleView Option="View.Month"></ScheduleView>
    <ScheduleView Option="View.Agenda"></ScheduleView>
  </ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 10);
List<AppointmentData> DataSource = new List<AppointmentData>
{
  new AppointmentData { Id = 1, Subject = "Paris", StartTime = new
  DateTime(2020, 1, 8, 10, 0, 0) , EndTime = new DateTime(2020, 1, 8, 12, 0,
  0),
  IsReadOnly = true },
  new AppointmentData { Id = 2, Subject = "Germany", StartTime = new
  DateTime(2020, 1, 10, 10, 0, 0) , EndTime = new DateTime(2020, 1, 10, 12, 0,
  0) }
};
public class AppointmentData
{
  public int Id { get; set; }
  public string Subject { get; set; }
  public string Location { get; set; }
  public DateTime StartTime { get; set; }
  public DateTime EndTime { get; set; }
  public string Description { get; set; }
  public bool IsAllDay { get; set; }
  public bool IsReadOnly { get; set; }
  public string RecurrenceRule { get; set; }
  public string RecurrenceException { get; set; }
  public Nullable<int> RecurrenceID { get; set; }
}
}
```

Customizing event tooltip using template

After enabling the default tooltip, it is possible to customize the display of needed event information on tooltip by making use of the `TooltipTemplate` option within the `ScheduleEventSettings`.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
```

```

<ScheduleEventSettings DataSource="@DataSource" EnableTooltip="true">
  <TooltipTemplate>
    <div class="tooltip-wrap">
      <div>@((context as AppointmentData).Subject)</div>
      <div>From&#160;:&#160;@((context as AppointmentData).StartTime)</div>
      <div>To&#160;:&#160;&#160;&#160;&#160;&#160;&#160;:&#160;@((context as
AppointmentData).EndTime) </div>
    </div>
  </TooltipTemplate>
</ScheduleEventSettings>
<ScheduleViews>
  <ScheduleView Option="View.Day"></ScheduleView>
  <ScheduleView Option="View.Week"></ScheduleView>
  <ScheduleView Option="View.WorkWeek"></ScheduleView>
  <ScheduleView Option="View.Month"></ScheduleView>
  <ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfScheduler>
@code{
  DateTime CurrentDate = new DateTime(2020, 1, 13);
  List<AppointmentData> DataSource = new List<AppointmentData>
  {
    new AppointmentData { Id = 1, Subject = "Paris", StartTime = new
DateTime(2020, 1, 14, 10, 0, 0) , EndTime = new DateTime(2020, 1, 14, 12, 0,
0) },
    new AppointmentData { Id = 2, Subject = "Germany", StartTime = new
DateTime(2020, 1, 15, 10, 0, 0) , EndTime = new DateTime(2020, 1, 15, 12, 0,
0) }
  };
  public class AppointmentData
  {
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
  }
}

```

All the field names that are mapped from the Scheduler dataSource to the appropriate field properties such as subject, description, location, startTime and endTime within the `ScheduleEventSettings` can be accessed within the template.

Appointment filtering

The appointments can be filtered by passing the predicate value to `Query` option in `ScheduleEventSettings`. The following code example shows how to filter and render the selected appointments alone in the Scheduler.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Data
@using Syncfusion.Blazor.Buttons
<SfCheckBox TChecked="bool" @bind-Checked="MargretChecked"
Value="@MargretId" ValueChange="@OnChange" Label="Margaret"
CssClass="margaret"></SfCheckBox>
<SfCheckBox TChecked="bool" @bind-Checked="RobertChecked" Value="@RobertId"
ValueChange="@OnChange" Label="Robert" CssClass="robert"></SfCheckBox>
<SfCheckBox TChecked="bool" @bind-Checked="LauraChecked" Value="@LauraId"
ValueChange="@OnChange" Label="Laura" CssClass="laura"></SfCheckBox>
<SfSchedule TValue="AppointmentData" CssClass='schedule-resource'
Width="100%" Height="650px" @bind-SelectedDate="@CurrentDate">
<ScheduleResources>
<ScheduleResource TValue="int[]" TItem="ResourceData"
DataSource="@OwnersData" Field="OwnerId" Title="Owners" Name="Owners"
TextField="OwnerText" IdField="OwnerId" ColorField="Color"
AllowMultiple="true"></ScheduleResource>
</ScheduleResources>
<ScheduleEventSettings DataSource="@DataSource"
Query="@ScheduleQuery"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
public DateTime CurrentDate { get; set; } = new DateTime(2020, 6, 5);
public bool MargretChecked { get; set; } = true;
public bool RobertChecked { get; set; } = true;
public bool LauraChecked { get; set; } = true;
public string MargretId { get; set; } = "1";
public string RobertId { get; set; } = "2";
public string LauraId { get; set; } = "3";
public dynamic predicate;
public Query ScheduleQuery { get; set; } = null;
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData {
Id = 1,
Subject = "Burning Man",
StartTime = new DateTime(2020, 5, 29, 15, 0, 0),
EndTime = new DateTime(2020, 5, 29, 17, 0, 0),
OwnerId = 1},
new AppointmentData{
Id = 2,
Subject = "Marketing Forum",
StartTime = new DateTime(2020, 5, 31, 10, 0, 0),
EndTime = new DateTime(2020, 5, 31, 11, 30, 0),
OwnerId = 2},
new AppointmentData{
Id = 3,
Subject = "Business Factory",
StartTime = new DateTime(2020, 5, 31, 13, 30, 0),
EndTime = new DateTime(2020, 5, 31, 15, 0, 0),

```

```
OwnerId = 3},
new AppointmentData{
Id = 4,
Subject = "Burning Man",
StartTime = new DateTime(2020, 6, 1, 11, 30, 0),
EndTime = new DateTime(2020, 6, 1, 13, 0, 0),
OwnerId = 1},
new AppointmentData{
Id = 5,
Subject = "Funnel Hacking",
StartTime = new DateTime(2020, 6, 2, 9, 30, 0),
EndTime = new DateTime(2020, 6, 2, 11, 0, 0),
OwnerId = 3},
new AppointmentData{
Id = 6,
Subject = "The human gathering",
StartTime = new DateTime(2020, 6, 2, 13, 0, 0),
EndTime = new DateTime(2020, 6, 2, 14, 30, 0),
OwnerId = 2},
new AppointmentData{
Id = 7,
Subject = "Techweek",
StartTime = new DateTime(2020, 6, 3, 11, 0, 0),
EndTime = new DateTime(2020, 6, 3, 12, 30, 0),
OwnerId = 2},
new AppointmentData{
Id = 8,
Subject = "Grow Conference",
StartTime = new DateTime(2020, 6, 4, 10, 0, 0),
EndTime = new DateTime(2020, 6, 4, 11, 30, 0),
OwnerId = 1},
new AppointmentData{
Id = 9,
Subject = "Data Science Conference",
StartTime = new DateTime(2020, 6, 4, 13, 30, 0),
EndTime = new DateTime(2020, 6, 4, 15, 0, 0),
OwnerId = 1},
new AppointmentData{
Id = 10,
Subject = "Blogcademy",
StartTime = new DateTime(2020, 6, 5, 12, 0, 0),
EndTime = new DateTime(2020, 6, 5, 13, 30, 0),
OwnerId = 3},
new AppointmentData{
Id = 11,
Subject = "World Domination Summit",
StartTime = new DateTime(2020, 6, 6, 9, 30, 0),
EndTime = new DateTime(2020, 6, 6, 11, 0, 0),
OwnerId = 2},
new AppointmentData{
Id = 12,
Subject = "Content Marketing",
StartTime = new DateTime(2020, 6, 6, 13, 0, 0),
EndTime = new DateTime(2020, 6, 6, 14, 30, 0),
OwnerId = 1},
new AppointmentData{
Id = 13,
```

```

Subject = "Mobile World Conference",
StartTime = new DateTime(2020, 6, 12, 18, 0, 0),
EndTime = new DateTime(2020, 6, 12, 20, 0, 0),
OwnerId = 1}
};
public void OnChange(ChangeEventArgs<bool> args)
{
    predicate = null;
    if (MargretChecked)
    {
        if (predicate != null)
        {
            predicate = predicate.Or("OwnerId", "equal", Convert.ToInt32(MargretId));
        }
        else
        {
            predicate = new WhereFilter() { Field = "OwnerId", Operator = "equal", value
            = Convert.ToInt32(MargretId) };
        }
    }
    if (RobertChecked)
    {
        if (predicate != null)
        {
            predicate = predicate.Or("OwnerId", "equal", Convert.ToInt32(RobertId));
        }
        else
        {
            predicate = new WhereFilter() { Field = "OwnerId", Operator = "equal", value
            = Convert.ToInt32(RobertId) };
        }
    }
    if (LauraChecked)
    {
        if (predicate != null)
        {
            predicate = predicate.Or("OwnerId", "equal", Convert.ToInt32(LauraId));
        }
        else
        {
            predicate = new WhereFilter() { Field = "OwnerId", Operator = "equal", value
            = Convert.ToInt32(LauraId) };
        }
    }
    if (predicate == null)
    {
        predicate = new WhereFilter() { Field = "OwnerId", Operator = "notequal",
        value = Convert.ToInt32(MargretId) }.And("OwnerId", "notequal",
        Convert.ToInt32(RobertId)).And("OwnerId", "notequal",
        Convert.ToInt32(LauraId));
    }
    ScheduleQuery = new Query().Where(predicate);
}
public List<ResourceData> OwnersData { get; set; } = new List<ResourceData>
{
    new ResourceData { OwnerText = "Margaret", OwnerId = 1, Color = "#ea7a57" },
    new ResourceData { OwnerText = "Robert", OwnerId = 2, Color = "#df5286" },

```



```

new ResourceData { OwnerText = "Laura", OwnerId = 3, Color = "#865fcf" }
};
public class ResourceData
{
    public int OwnerId { get; set; }
    public string OwnerText { get; set; }
    public string Color { get; set; }
}
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
    public int OwnerId { get; set; }
}
}
<style>
.e-checkbox-wrapper.margaret .e-frame {
background-color: #ea7a57;
border-color: transparent;
}
.e-checkbox-wrapper.robert .e-frame {
background-color: #df5286;
border-color: transparent;
}
.e-checkbox-wrapper.laura .e-frame {
background-color: #865fcf;
border-color: transparent;
}
</style>

```

Appointment selection

Appointment selection can be done either through mouse or keyboard actions. The selected events in UI will have a box shadow effect around to differentiate it from other appointments.

| Action | Description |

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

| Mouse click or Single tap on appointments | Selects single appointment. |

| Ctrl + [Mouse click] or [Single tap] on appointments | Selects multiple appointments. |

Deleting multiple appointments

With the options available to select multiple appointments, it is also possible to delete the multiple selected appointments simply by pressing the **delete** key. In case of deleting multiple selected occurrences of an event series, only those occurrences will be deleted and not the entire series.

Retrieve event details from the UI of an event

It is possible to access the information about the event fields of an appointment based on the X and Y co-ordinates. This can be achieved by passing an X and Y co-ordinates to the public method

`GetTargetEventAsync`. You can also get the selected appointment details using

`GetSelectedEventsAsync` method.

Get the current view appointments

To retrieve the appointments present in the current view of the Scheduler, the `GetCurrentViewEvents` public method can be used. In the following example, current view appointment collection rendered has been traced in `DataBound` event.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" @ref="ScheduleRef" Width="100%"
Height="550px" @bind-SelectedDate="@CurrentDate">
<ScheduleEvents TValue="AppointmentData"
DataBound="OnDataBound"></ScheduleEvents>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>

@code{
DateTime CurrentDate = new DateTime(2020, 1, 10);
SfSchedule<AppointmentData> ScheduleRef;
public void OnDataBound(DataBoundEventArgs<AppointmentData> args)
{
List<AppointmentData> eventCollection = ScheduleRef.GetCurrentViewEvents();
//You can get the current view appointment collections in the
EventCollection variable
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 10, 9, 30, 0) , EndTime = new DateTime(2020, 1, 10, 11,
30, 0) },
new AppointmentData { Id = 1, Subject = "Conference", StartTime = new
DateTime(2020, 1, 9, 11, 30, 0) , EndTime = new DateTime(2020, 1, 9, 13, 0,
0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
}
```

```
public Nullable<int> RecurrenceID { get; set; }
}
}
```

Get the entire appointment collections

The entire collection of appointments rendered on the Scheduler can be accessed using the `GetEventsAsync` public method. In the following example, entire appointment collection rendered on the Scheduler has been traced in `DataBound` event.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" @ref="ScheduleRef" Width="100%"
Height="550px" SelectedDate="@ (new DateTime(2020,1,10)) ">
<ScheduleEvents TValue="AppointmentData"
DataBound="OnDataBound"></ScheduleEvents>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
SfSchedule<AppointmentData> ScheduleRef;
public async void OnDataBound(DataBoundEventArgs<AppointmentData> args)
{
List<AppointmentData> EventCollection = await ScheduleRef.GetEventsAsync();
//You can get the entire appointment collections in the EventCollection
variable
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 10, 9, 30, 0) , EndTime = new DateTime(2020, 1, 10, 11,
30, 0) },
new AppointmentData { Id = 1, Subject = "Conference", StartTime = new
DateTime(2020, 2, 9, 11, 30, 0) , EndTime = new DateTime(2020, 2, 9, 13, 0,
0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}
```

You can also get the specific range of appointments by passing the start and end time in the `GetEventsAsync` method. To get the block events, you can make use of the method `GetBlockEventsAsync`.

Refresh appointments

If the requirement is to simply refresh the appointments instead of refreshing the entire Scheduler elements from the application end, make use of the `RefreshEventsAsync` public method.

CSHARP

```
ScheduleRef.RefreshEventsAsync();
```

Data Binding in Blazor Scheduler Component

The Scheduler uses `DataManager`, which supports both RESTful data service binding and datasource collections. The `DataSource` property of Scheduler can be assigned either with the instance of `DataManager` or list of datasource collection, as it supports the following two kinds of data binding methods:

- Local data
- Remote data

You can check out the following video to bind the appointments in the Blazor Scheduler.

{% youtube

"youtube:https://www.youtube.com/watch?v=EwfxPrqxma8"%}

Binding local data

To bind local data to the Scheduler, you can simply assign a list of datasource collections to the `DataSource` option of the scheduler within the `ScheduleEventSettings` tag. The local data source can also be provided as an instance of the `DataManager`.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" SelectedDate="@ (new
DateTime(2020, 2, 12))">
  <ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
</SfSchedule>
@code{
List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData { Id = 1, Subject = "Testing", StartTime = new
DateTime(2020, 2, 13, 9, 30, 0) , EndTime = new DateTime(2020, 2, 13, 10,
30, 0)},
    new AppointmentData { Id = 2, Subject = "Conference", StartTime = new
DateTime(2020, 2, 11, 10, 30, 0) , EndTime = new DateTime(2020, 2, 11, 12,
0, 0)},
    new AppointmentData { Id = 3, Subject = "Meeting", StartTime = new
DateTime(2020, 2, 9, 9, 30, 0) , EndTime = new DateTime(2020, 2, 9, 11, 30,
0)},
}
```

```

new AppointmentData { Id = 4, Subject = "Vacation", StartTime = new
DateTime(2020, 2, 14, 11, 30, 0) , EndTime = new DateTime(2020, 2, 14, 13,
0, 0)}
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

By default, **DataManager** uses **BlazorAdaptor** for binding local data.

You can also bind different field names to the default event fields as well as include additional custom fields to the event object collection which can be referred [here](#).

Binding remote data

Any kind of remote data services can be bound to the Scheduler. To do so, provide the service URL to the **Url** option of **SfDataManager** within **ScheduleEventSettings** tag.

Using ODataV4Adaptor

ODataV4 is a standardized protocol for creating and consuming data. Refer to the following code example to retrieve the data from ODataV4 service using the **DataManager**. To connect with ODataV4 service end points, it is necessary to make use of **ODataV4Adaptor** within **DataManager**.

ASPX-CS

```

@using Syncfusion.Blazor
@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Data
<SfSchedule TValue="Restful_Crud.Models.EventData" Height="550px"
SelectedDate="@ (new DateTime(2020, 3, 11))">
<ScheduleEventSettings TValue="Restful_Crud.Models.EventData"
Query="@QueryData">
<SfDataManager Url="http://localhost:25255/odata"
Adaptor="Adaptors.ODataV4Adaptor"></SfDataManager>
</ScheduleEventSettings>
</SfSchedule>
@code{
public Query QueryData = new Query().From("EventDatas");
}

```

Using custom adaptor

It is possible to create your own **CustomAdaptor** by extending the built-in available adaptors. The following example demonstrates the custom adaptor usage and how to bind the data with custom

service and the CRUD operations for custom bounded data is performed using the methods of [DataAdaptor](#) abstract class.

ASPX-CS

```
@using Syncfusion.Blazor
@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Data
<SfSchedule TValue="AppointmentData" Width="100%" Height="650px"
SelectedDate="@ (new DateTime(2020, 1, 9))">
<ScheduleResources>
<ScheduleResource TItem="ResourceData" TValue="int"
DataSource="@ProjectData" Field="ProjectId" Title="Choose Project"
Name="Projects" TextField="Text" IdField="Id" ColorField="Color">
</ScheduleResource>
</ScheduleResources>
<ScheduleEventSettings TValue="AppointmentData">
<SfDataManager AdaptorInstance="@typeof(CustomAdaptor)"
Adaptor="Adaptors.CustomAdaptor"></SfDataManager>
</ScheduleEventSettings>
</SfSchedule>
@code {
public class CustomAdaptor : DataAdaptor
{
List<AppointmentData> EventData = DataList();
public async override Task<object> ReadAsync(DataManagerRequest
dataManagerRequest, string key = null)
{
await Task.Delay(100); //To mimic asynchronous operation, we delayed this
operation using Task.Delay
return dataManagerRequest.RequiresCounts ? new DataResult() { Result =
EventData, Count = EventData.Count() } : (object)EventData;
}
public async override Task<object> InsertAsync(DataManager dataManager,
object data, string key)
{
await Task.Delay(100); //To mimic asynchronous operation, we delayed this
operation using Task.Delay
EventData.Insert(0, data as AppointmentData);
return data;
}
public async override Task<object> UpdateAsync(DataManager dataManager,
object data, string keyField, string key)
{
await Task.Delay(100); //To mimic asynchronous operation, we delayed this
operation using Task.Delay
var val = (data as AppointmentData);
var appointment = EventData.Where((AppointmentData) => AppointmentData.Id ==
val.Id).FirstOrDefault();
if (appointment != null)
{
appointment.Id = val.Id;
appointment.Subject = val.Subject;
appointment.StartTime = val.StartTime;
appointment.EndTime = val.EndTime;
appointment.Location = val.Location;
appointment.Description = val.Description;
```

```

appointment.IsAllDay = val.IsAllDay;
appointment.ProjectId = val.ProjectId;
appointment.RecurrenceException = val.RecurrenceException;
appointment.RecurrenceID = val.RecurrenceID;
appointment.RecurrenceRule = val.RecurrenceRule;
}
return data;
}
public async override Task<object> RemoveAsync(DataManager dataManager,
object data, string keyField, string key) //triggers on appointment deletion
through public method DeleteEvent
{
await Task.Delay(100); //To mimic asynchronous operation, we delayed this
operation using Task.Delay
int value = (int)data;
EventData.Remove(EventData.Where((AppointmentData) => AppointmentData.Id ==
value).FirstOrDefault());
return data;
}
public async override Task<object> BatchUpdateAsync(DataManager dataManager,
object changedRecords, object addedRecords, object deletedRecords, string
keyField, string key, int? dropIndex)
{
await Task.Delay(100); //To mimic asynchronous operation, we delayed this
operation using Task.Delay
object records = deletedRecords;
List<AppointmentData> deleteData = deletedRecords as List<AppointmentData>;
foreach (var data in deleteData)
{
EventData.Remove(EventData.Where((AppointmentData) => AppointmentData.Id ==
data.Id).FirstOrDefault());
}
List<AppointmentData> addData = addedRecords as List<AppointmentData>;
foreach (var data in addData)
{
EventData.Insert(0, data as AppointmentData);
records = addedRecords;
}
List<AppointmentData> updateData = changedRecords as List<AppointmentData>;
foreach (var data in updateData)
{
var val = (data as AppointmentData);
var appointment = EventData.Where((AppointmentData) => AppointmentData.Id ==
val.Id).FirstOrDefault();
if (appointment != null)
{
appointment.Id = val.Id;
appointment.Subject = val.Subject;
appointment.StartTime = val.StartTime;
appointment.EndTime = val.EndTime;
appointment.Location = val.Location;
appointment.Description = val.Description;
appointment.IsAllDay = val.IsAllDay;
appointment.ProjectId = val.ProjectId;
appointment.RecurrenceException = val.RecurrenceException;
appointment.RecurrenceID = val.RecurrenceID;
appointment.RecurrenceRule = val.RecurrenceRule;
}
}
}

```

```

    }
    records = changedRecords;
    }
    return records;
    }
    }
    private static List<AppointmentData> DataList()
    {
        List<AppointmentData> eventDatas = new List<AppointmentData>
        {
            new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
            DateTime(2020, 1, 5, 10, 0, 0) , EndTime = new DateTime(2020, 1, 5, 11, 0,
            0), ProjectId = 1, RecurrenceRule = "FREQ=DAILY;INTERVAL=1;COUNT=5;"},
            new AppointmentData { Id = 2, Subject = "Project Discussion", StartTime =
            new DateTime(2020, 1, 6, 11, 30, 0) , EndTime = new DateTime(2020, 1, 6, 13,
            0, 0), ProjectId = 2},
            new AppointmentData { Id = 3, Subject = "Work Flow Analysis", StartTime =
            new DateTime(2020, 1, 7, 12, 0, 0) , EndTime = new DateTime(2020, 1, 7, 13,
            0, 0), ProjectId = 2, RecurrenceRule = "FREQ=DAILY;INTERVAL=1;COUNT=3;"},
            new AppointmentData { Id = 4, Subject = "Report", StartTime = new
            DateTime(2020, 1, 10, 11, 30, 0) , EndTime = new DateTime(2020, 1, 10, 13,
            0, 0), ProjectId = 2}
        };
        return eventDatas;
    }
    List<ResourceData> ProjectData = ResourceList();
    private static List<ResourceData> ResourceList()
    {
        List<ResourceData> resourceDatas = new List<ResourceData>
        {
            new ResourceData { Text = "PROJECT 1", Id = 1, Color = "#cb6bb2" },
            new ResourceData { Text = "PROJECT 2", Id = 2, Color = "#56ca85" }
        };
        return resourceDatas;
    }
    public class AppointmentData
    {
        public int Id { get; set; }
        public string Subject { get; set; }
        public string Location { get; set; }
        public DateTime StartTime { get; set; }
        public DateTime EndTime { get; set; }
        public string Description { get; set; }
        public bool IsAllDay { get; set; }
        public string RecurrenceRule { get; set; }
        public string RecurrenceException { get; set; }
        public Nullable<int> RecurrenceID { get; set; }
        public int ProjectId { get; set; }
    }
    public class ResourceData
    {
        public int Id { get; set; }
        public string Text { get; set; }
        public string Color { get; set; }
    }
}

```


Binding ExpandoObject

Scheduler is a generic component which is strongly bound to a model type. There are cases when the model type is unknown during compile type. In such cases, bind the data to the scheduler as list of **ExpandoObject**.

ExpandoObject can be bound to the **DataSource** option of the scheduler within the **ScheduleEventSettings** tag. Scheduler can also perform all kinds of supported data operations and editing in ExpandoObject.

CSHARP

```
@using System.Dynamic
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="ExpandoObject" @bind-SelectedDate="@CurrentDate"
Width="100%" Height="550px">
<ScheduleEventSettings DataSource="@EventsCollection"
AllowEditFollowingEvents="true"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code {
DateTime CurrentDate = new DateTime(2021, 8, 10);
public List<ExpandoObject> EventsCollection = new List<ExpandoObject>() { };
protected override void OnInitialized()
{
DateTime scheduleStart = new DateTime(2021, 8, 8, 10, 0, 0);
EventsCollection = Enumerable.Range(1, 5).Select((x) =>
{
scheduleStart = scheduleStart.AddDays(1);
dynamic d = new ExpandoObject();
d.Id = 1000 + x;
d.Subject = (new string[] { "Project Discussion", "Work Flow Analysis",
"Report", "Meeting", "Project Demo" })[new Random().Next(5)];
d.StartTime = scheduleStart;
d.EndTime = scheduleStart.AddHours(1);
d.IsAllDay = false;
d.RecurrenceRule = null;
d.RecurrenceException = null;
d.RecurrenceID = null;
return d;
}).Cast<ExpandoObject>().ToList<ExpandoObject>();
}
}
```

Binding DynamicObject

Scheduler is a generic component which is strongly bound to a model type. There are cases when the model type is unknown during compile type. In such cases, bind the data to the scheduler as list of **DynamicObject**.

DynamicObject can be bound to the **DataSource** option of the scheduler within the **ScheduleEventSettings** tag. Scheduler can also perform all kinds of supported data operations and editing in **DynamicObject**.

The [GetDynamicMemberNames](#) method of **DynamicObject** class must be overridden and return the property names to perform data operation and editing while using **DynamicObject**.

CSHARP

```
@using System.Dynamic
@using System.Text.Json
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="DynamicDictionary" @bind-SelectedDate="@CurrentDate"
Width="100%" Height="550px">
<ScheduleEventSettings
DataSource="@EventsCollection"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code {
DateTime CurrentDate = new DateTime(2021, 8, 10);
public List<DynamicDictionary> EventsCollection = new
List<DynamicDictionary>() { };
protected override void OnInitialized()
{
DateTime scheduleStart = new DateTime(2021, 8, 8, 10, 0, 0);
EventsCollection = Enumerable.Range(1, 5).Select((x) =>
{
scheduleStart = scheduleStart.AddDays(1);
dynamic d = new DynamicDictionary();
d.Id = 1000 + x;
d.Subject = (new string[] { "Project Discussion", "Work Flow Analysis",
"Report", "Meeting", "Project Demo" })[new Random().Next(5)];
d.StartTime = scheduleStart;
d.EndTime = scheduleStart.AddHours(1);
d.RecurrenceRule = null;
d.RecurrenceException = null;
d.RecurrenceID = null;
return d;
}).Cast<DynamicDictionary>().ToList<DynamicDictionary>();
}
public class DynamicDictionary : System.Dynamic.DynamicObject
{
Dictionary<string, object> dictionary = new Dictionary<string, object>();
public override bool TryGetMember(GetMemberBinder binder, out object result)
{
string name = binder.Name;
return dictionary.TryGetValue(name, out result);
}
public override bool TrySetMember(SetMemberBinder binder, object value)
{
dictionary[binder.Name] = value;
}
```

```

return true;
}
public override System.Collections.Generic.IEnumerable<string>
GetDynamicMemberNames()
{
return this.dictionary?.Keys;
}
}
}
}

```

Binding ObservableCollection

This [ObservableCollection](#) (dynamic data collection) provides notifications when items are added, removed and moved. The implement [INotifyCollectionChanged](#) notifies when dynamic changes of add,remove, move and clear the collection. The implement [INotifyPropertyChanged](#) notifies when property value has changed in client side.

Here, AppointmentData class implements the interface of **INotifyPropertyChanged** and it raises the event when Subject property value was changed.

CSHARP

```

@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Buttons
@using System.Collections.ObjectModel
@using System.ComponentModel
<SfButton @onclick="AddRecord">Add Data</SfButton>
<SfButton @onclick="UpdateRecord" Disabled="ObservableData.Count ==
0">Update Data</SfButton>
<SfButton @onclick="DeleteRecord" Disabled="ObservableData.Count ==
0">Delete Data</SfButton>
<SfSchedule TValue="AppointmentData" @bind-SelectedDate="@CurrentDate"
Width="100%" Height="550px">
<ScheduleEventSettings DataSource="@ObservableData"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 3, 10);
public ObservableCollection<AppointmentData> ObservableData { get; set; }
List<AppointmentData> EventsCollection = new List<AppointmentData>();
int uniqueid = 1;
protected override void OnInitialized()
{
EventsCollection = Enumerable.Range(1, 4).Select(x => new AppointmentData()
{
Id = x,
Subject = (new string[] { "Project Discussion", "Work Flow Analysis",
"Report", "Meeting", "Project Demo" })[new Random().Next(5)],
StartTime = new DateTime(2020, 3, 8 + x, 9, 0, 0),
EndTime = new DateTime(2020, 3, 8 + x, 11, 0, 0)
}).ToList();
}
}

```

```

ObservableData = new
ObservableCollection<AppointmentData>(EventsCollection);
}
public void AddRecord()
{
    uniqueid++;
    ObservableData.Add(new AppointmentData() { Id = uniqueid, Subject =
    "Meeting", StartTime = new DateTime(2020, 3, 13, 9, 0, 0), EndTime = new
    DateTime(2020, 3, 13, 11, 0, 0) });
}
public void DeleteRecord()
{
    if (ObservableData.Count != 0)
    {
        ObservableData.Remove(ObservableData.First());
    }
}
public void UpdateRecord()
{
    if (ObservableData.Count != 0)
    {
        var data = ObservableData.First();
        data.Subject = "Event Updated";
    }
}
public class AppointmentData : INotifyPropertyChanged
{
    public int Id { get; set; }
    private string subject { get; set; }
    public string Subject
    {
        get { return subject; }
        set
        {
            this.subject = value;
            NotifyPropertyChanged("Subject");
        }
    }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
    public string StartTimezone { get; set; }
    public string EndTimezone { get; set; }
    public event PropertyChangedEventHandler PropertyChanged;
    private void NotifyPropertyChanged(string propertyName)
    {
        PropertyChangedEventHandler handler = PropertyChanged;
        if (handler != null)
        {
            handler(this, new PropertyChangedEventArgs(propertyName));
        }
    }
}

```

```
}
}
```

Performing CRUD using Entity Framework

You need to follow the below steps to consume data from the **Entity Framework** in our Scheduler component.

Create DbContext class

The first step is to create a DbContext class called **ScheduleDataContext** to connect to a Microsoft SQL Server database.

CSHARP

```
using System;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata;
namespace Restful_Services.Models
{
    public partial class ScheduleDataContext : DbContext
    {
        public ScheduleDataContext()
        {
        }
        public ScheduleDataContext(DbContextOptions<ScheduleDataContext> options)
        : base(options)
        {
        }
        public virtual DbSet<EventData> EventData { get; set; }
        protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
        {
            if (!optionsBuilder.IsConfigured)
            {
                optionsBuilder.UseSqlServer("Data
Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=D:\\SchedulerCRUD\\Restful_S
ervices\\App_Data\\ScheduleData.mdf;Integrated
Security=True;MultipleActiveResultSets=True;Application
Name=EntityFramework;Integrated Security=True");
            }
        }
        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<EventData>(entity =>
            {
                entity.Property(e => e.Id).ValueGeneratedNever();
                entity.Property(e => e.EndTime).HasColumnType("datetime");
                entity.Property(e => e.RecurrenceID).HasColumnName("RecurrenceID");
                entity.Property(e => e.StartTime).HasColumnType("datetime");
            });
            OnModelCreatingPartial(modelBuilder);
        }
        partial void OnModelCreatingPartial(ModelBuilder modelBuilder);
    }
}
```

Creating OData Controller

A OData Controller has to be created which allows Scheduler directly to consume data from the Entity Framework. The following code example shows how to perform CRUD operations using Entity Framework.

CSHARP

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Data.Entity.Infrastructure;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
using System.Web.Mvc;
using System.Web.Routing;
using Microsoft.AspNet.OData;
using Restful_Services.Models;
namespace Restful_Services.Controllers
{
    public class ODataV4Controller : ODataController
    {
        private ScheduleDataContext db = new ScheduleDataContext();
        // GET: odata/ODataV4
        [EnableQuery]
        [AcceptVerbs("GET")]
        public IQueryable<EventData> GetODataV4()
        {
            return db.EventData;
        }
        // GET: odata/ODataV4(5)
        [EnableQuery]
        [AcceptVerbs("GET")]
        public IQueryable<EventData> GetODataV4(string startDate, string endDate)
        {
            DateTime start = DateTime.Parse(startDate);
            DateTime end = DateTime.Parse(endDate);
            return db.EventData.Where(evt => evt.StartTime >= start && evt.EndTime <= end);
        }
        // POST: odata/ODataV4
        [AcceptVerbs("POST", "OPTIONS")]
        public void Post([FromBody]EventData eventData)
        {
            if (ModelState.IsValid)
            {
                EventData insertData = new EventData();
                insertData.Id = (db.EventData.ToList().Count > 0 ?
                db.EventData.ToList().Max(p => p.Id) : 1) + 1;
                insertData.Subject = eventData.Subject;
                insertData.StartTime = Convert.ToDateTime(eventData.StartTime);
                insertData.EndTime = Convert.ToDateTime(eventData.EndTime);
                insertData.StartTimezone = eventData.StartTimezone;
                insertData.EndTimezone = eventData.EndTimezone;
                insertData.Location = eventData.Location;
                insertData.Description = eventData.Description;
            }
        }
    }
}
```

```
insertData.IsAllDay = eventData.IsAllDay;
insertData.IsBlock = eventData.IsBlock;
insertData.IsReadOnly = eventData.IsReadOnly;
insertData.FollowingID = eventData.FollowingID;
insertData.RecurrenceID = eventData.RecurrenceID;
insertData.RecurrenceRule = eventData.RecurrenceRule;
insertData.RecurrenceException = eventData.RecurrenceException;
db.EventData.Add(insertData);
db.SaveChanges();
}
}
// PATCH: odata/ODataV4(5)
[AcceptVerbs("PATCH", "MERGE", "OPTIONS")]
public void Patch([FromBody]EventData eventData)
{
    if (ModelState.IsValid)
    {
        EventData updateData = db.EventData.First(i => i.Id ==
            Convert.ToInt32(eventData.Id));
        if (updateData != null)
        {
            updateData.Subject = eventData.Subject;
            updateData.StartTime = Convert.ToDateTime(eventData.StartTime);
            updateData.EndTime = Convert.ToDateTime(eventData.EndTime);
            updateData.StartTimezone = eventData.StartTimezone;
            updateData.EndTimezone = eventData.EndTimezone;
            updateData.Location = eventData.Location;
            updateData.Description = eventData.Description;
            updateData.IsAllDay = eventData.IsAllDay;
            updateData.IsBlock = eventData.IsBlock;
            updateData.IsReadOnly = eventData.IsReadOnly;
            updateData.FollowingID = eventData.FollowingID;
            updateData.RecurrenceID = eventData.RecurrenceID;
            updateData.RecurrenceRule = eventData.RecurrenceRule;
            updateData.RecurrenceException = eventData.RecurrenceException;
            db.SaveChanges();
        }
    }
}
// DELETE: odata/ODataV4(5)
[AcceptVerbs("DELETE", "OPTIONS")]
public void Delete([FromODataUri]int key)
{
    if (ModelState.IsValid)
    {
        EventData removeData = db.EventData.First(i => i.Id == key);
        if (removeData != null)
        {
            db.EventData.Remove(removeData);
            db.SaveChanges();
        }
    }
}
```

Configure Scheduler component using ODataV4Adaptor

Now, the Scheduler can be configured using the `SfDataManager` to interact with the created OData service and consume the data appropriately. To interact with OData, use `ODataV4Adaptor`.

ASPX-CS

```
@using Syncfusion.Blazor
@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Data
<SfSchedule TValue="Restful_Services.Models.EventData" Height="550px"
SelectedDate="@ (new DateTime(2020, 04, 14))" >
<ScheduleEventSettings TValue="Restful_Services.Models.EventData"
Query="@QueryData" >
<SfDataManager Url="http://localhost:9876/odata/"
Adaptor="Adaptors.ODataV4Adaptor"></SfDataManager>
</ScheduleEventSettings>
</SfSchedule>
@code {
public Query QueryData = new Query().From("ODataV4");
}
```

You can find the working sample [here](#).

Passing additional parameters to the server

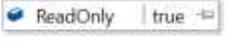
To send an additional custom parameter to the server-side post, make use of the `AddParams` method of `Query`. Now, assign this `Query` object with additional parameters to the `Query` property of Scheduler.

ASPX-CS

```
@using Syncfusion.Blazor
@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Data
<SfSchedule TValue="Restful_Crud.Models.EventData" Height="550px"
SelectedDate="@ (new DateTime(2020, 3, 11))">
<ScheduleEventSettings TValue="Restful_Crud.Models.EventData"
Query="@QueryData">
<SfDataManager Url="http://localhost:25255/odata"
Adaptor="Adaptors.ODataV4Adaptor"></SfDataManager>
</ScheduleEventSettings>
</SfSchedule>
@code{
public Query QueryData = new
Query().From("EventDatas").AddParams("ReadOnly", true);
}
```

The value passed to the additional parameter is shown in the following image.

```
[EnableQuery]
[AcceptVerbs("GET")]
public IQueryable<EventData> GetEventDatas(string StartDate, string EndDate, bool ReadOnly)
{
    DateTime start = DateTime.Parse(StartDate);
    DateTime end = DateTime.Parse(EndDate);
    return db.EventDatas.Where(evt => evt.StartTime >= start && evt.EndTime <= end);
}
```



The parameters added using the **Query** property will be sent along with the data request sent to the server on every scheduler actions.

Scheduler CRUD actions

The CRUD (Create, Read, Update and Delete) actions can be performed easily on Scheduler appointments using the various adaptors available within the **DataManager**. Most preferably, we will be using **ODataV4Adaptor** for performing CRUD actions on scheduler appointments.

ASPX-CS

```
@using Syncfusion.Blazor
@using Syncfusion.Blazor.Data
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="Restful_Crud.Models.EventData"
SelectedDate="@ (DateTime.Now.Date)" >
<ScheduleEventSettings TValue="Restful_Crud.Models.EventData"
Query="@QueryData">
<SfDataManager Url="http://localhost:25255/odata"
Adaptor="Adaptors.ODataV4Adaptor"></SfDataManager>
</ScheduleEventSettings>
</SfSchedule>
@code{
public Query QueryData { get; set; } = new Query().From("EventDatas");
}
```

The server-side controller code to handle the CRUD operations are as follows.

SH

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Data.Entity.Infrastructure;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
using System.Web.Http.ModelBinding;
using System.Web.Http.OData;
using System.Web.Http.OData.Routing;
using Restful_Crud.Models;
namespace Restful_Crud.Controllers
{
public class EventDatasController : ODataController
{
private ScheduleDataEntities1 db = new ScheduleDataEntities1();
// GET: odata/EventDatas
[EnableQuery]
[AcceptVerbs("GET")]
public IQueryable<EventData> GetEventDatas()
{
return db.EventDatas;
}
// GET: odata/EventDatas(5)
[EnableQuery]
```

```

[AcceptVerbs("GET")]
public IQueryable<EventData> GetEventDatas(string StartDate, string EndDate,
bool ReadOnly)
{
    DateTime start = DateTime.Parse(StartDate);
    DateTime end = DateTime.Parse(EndDate);
    return db.EventDatas.Where(evt => evt.StartTime >= start && evt.EndTime <=
end);
}
// POST: odata/EventDatas
[AcceptVerbs("POST", "OPTIONS")]
public void Post([FromBody]CrudData eventData)
{
    EventData insertData = new EventData();
    insertData.Id = (db.EventDatas.ToList().Count > 0 ?
db.EventDatas.ToList().Max(p => p.Id) : 1) + 1;
    insertData.StartTime =
Convert.ToDateTime(eventData.StartTime).ToLocalTime();
    insertData.EndTime = Convert.ToDateTime(eventData.EndTime).ToLocalTime();
    insertData.Subject = eventData.Subject;
    insertData.IsAllDay = eventData.IsAllDay;
    insertData.Location = eventData.Location;
    insertData.Description = eventData.Description;
    insertData.RecurrenceRule = eventData.RecurrenceRule;
    insertData.RecurrenceID = eventData.RecurrenceID;
    insertData.RecurrenceException = eventData.RecurrenceException;
    insertData.StartTimezone = eventData.StartTimezone;
    insertData.EndTimezone = eventData.EndTimezone;
    db.EventDatas.Add(insertData);
    db.SaveChanges();
}
// PATCH: odata/EventDatas(5)
[AcceptVerbs("PATCH", "MERGE", "OPTIONS")]
public void Patch([FromBody]CrudData eventData)
{
    EventData updateData = db.EventDatas.Find(Convert.ToInt32(eventData.Id));
    if (updateData != null)
    {
        updateData.StartTime =
Convert.ToDateTime(eventData.StartTime).ToLocalTime();
        updateData.EndTime = Convert.ToDateTime(eventData.EndTime).ToLocalTime();
        updateData.Subject = eventData.Subject;
        updateData.IsAllDay = eventData.IsAllDay;
        updateData.Location = eventData.Location;
        updateData.Description = eventData.Description;
        updateData.RecurrenceRule = eventData.RecurrenceRule;
        updateData.RecurrenceID = eventData.RecurrenceID;
        updateData.RecurrenceException = eventData.RecurrenceException;
        updateData.StartTimezone = eventData.StartTimezone;
        updateData.EndTimezone = eventData.EndTimezone;
        db.SaveChanges();
    }
}
// DELETE: odata/EventDatas(5)
[AcceptVerbs("DELETE", "OPTIONS")]
public void Delete([FromODataUri]int key)
{

```

```
EventData removeData = db.EventDatas.Find(key);
if (removeData != null)
{
    db.EventDatas.Remove(removeData);
    db.SaveChanges();
}
}
}
}
```

Configuring Scheduler with Google API service

We have assigned the dataSource that is retrieved from the Google services within the `OnInitializedAsync` method. And, the CRUD actions are performed within the `ActionCompleted` event.

We have to write our own service to connect retrieve the events from the Google calendar.

The runnable sample for the above code will be available [here](#).

CRUD actions in Blazor Scheduler Component

Events, a.k.a. Appointments, play an important role in Scheduler with which the users mostly interact. You can easily manipulate (add/edit/delete) the desired appointments as and when required either using the editor window or through the drag and resize action.

Add

Any kind of appointments such as normal, all-day, spanned or recurring events can be easily added on Scheduler using any one of the following ways.

- [Creation using editor window](#)
- [Creation using AddEventAsync method](#)

Creation using editor window

The default editor window opens when you double click on the Scheduler cells. It provides with event related options such as Subject, Location, Start and End time, All-day, Timezone, Description and other recurrence options. With these available fields, you can choose to provide detailed information to the events. Once the fields are filled with proper values, enter the `Save` button to add an event.

In case, if you want to simply provide the Subject alone for appointments, just single click on the required cells which will open the quick popup expecting you to enter subject alone and save it. You can also select multiple cells and press `Enter` key to open the quick popup for selected time range and save the appointment for that time range.

Creation using AddEventAsync method

The appointments can be created dynamically by using `AddEventAsync` method.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Buttons
<SfButton Content="ADD" OnClick="OnClick"></SfButton>
<SfSchedule @ref="ScheduleRef" TValue="AppointmentData" Height="550px"
@bind-SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
```

```

<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
    DateTime CurrentDate = new DateTime(2020, 1, 6);
    SfSchedule<AppointmentData> ScheduleRef;
    List<AppointmentData> DataSource = new List<AppointmentData>
    {
        new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
        DateTime(2020, 1, 6, 9, 30, 0) , EndTime = new DateTime(2020, 1, 6, 11, 0,
        0),
        RecurrenceRule = "FREQ=DAILY;INTERVAL=1;COUNT=5" }
    };
    public async Task OnClick()
    {
        AppointmentData eventData = new AppointmentData
        {
            Id = 10,
            Subject = "Added Event",
            StartTime = new DateTime(2020, 1, 7, 9, 30, 0),
            EndTime = new DateTime(2020, 1, 7, 11, 30, 0),
        };
        await ScheduleRef.AddEventAsync(eventData);
    }
    public class AppointmentData
    {
        public int Id { get; set; }
        public string Subject { get; set; }
        public string Location { get; set; }
        public DateTime StartTime { get; set; }
        public DateTime EndTime { get; set; }
        public string Description { get; set; }
        public bool IsAllDay { get; set; }
        public string RecurrenceRule { get; set; }
        public string RecurrenceException { get; set; }
        public Nullable<int> RecurrenceID { get; set; }
    }
}

```

Inline creation

Another easier way to create the appointments is enabling the **AllowInline** property. By single clicking on the scheduler cells or pressing **enter** key on selected cells, the appointment like textbox will be displayed in which you can enter the Subject of the appointment. Pressing enter key or clicking out of the text box will create the appointment in the scheduler.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" AllowInline="true"
@bind-SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>

```

```

<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfScheduler>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 6);
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 6, 9, 30, 0) , EndTime = new DateTime(2020, 1, 6, 11, 0,
0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

Inserting events into database at server-side

While adding the normal or recurring events to the Scheduler, **insert** action takes place and the following code example describes how to add a new event into database at server side.

SH

```

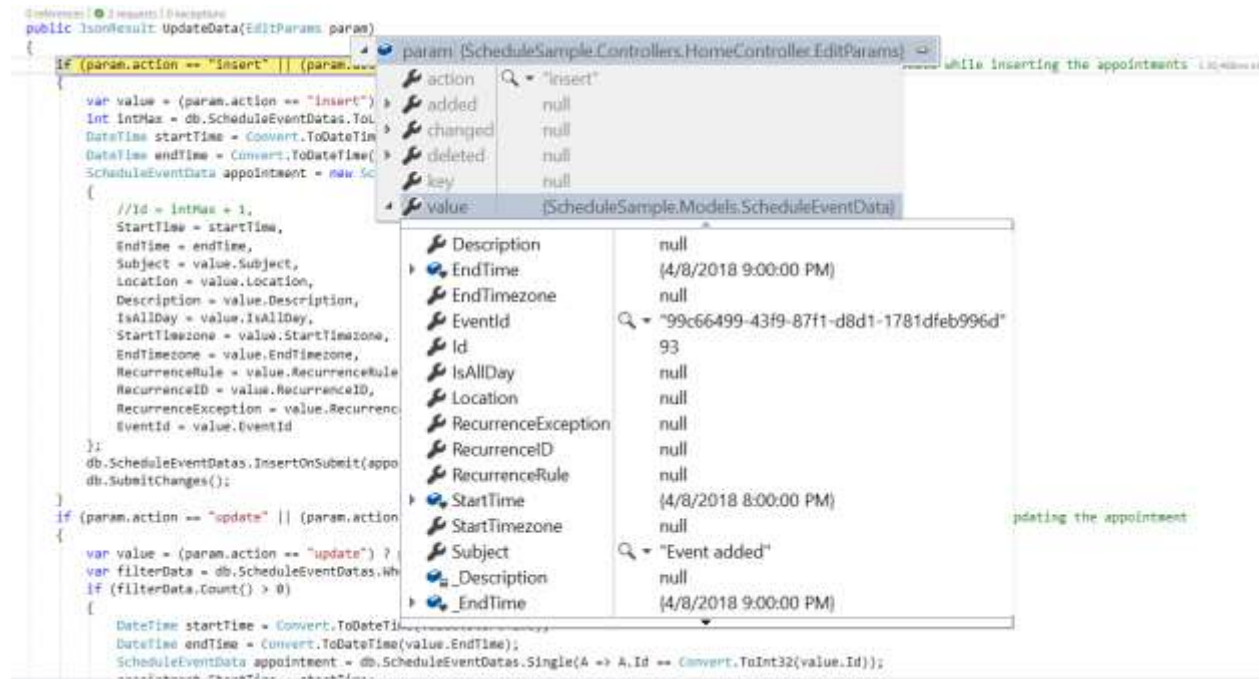
if (param.action == "insert" || (param.action == "batch" && param.added !=
null)) // this block of code will execute while inserting the appointments
{
var value = (param.action == "insert") ? param.value : param.added[0];
int intMax = db.ScheduleEventDatas.Select(x =>
x.Id).DefaultIfEmpty(0).Max();
DateTime startTime = Convert.ToDateTime(value.StartTime);
DateTime endTime = Convert.ToDateTime(value.EndTime);
ScheduleEventData appointment = new ScheduleEventData()
{
Id = intMax + 1,
StartTime = startTime,
EndTime = endTime,
Subject = value.Subject,
IsAllDay = value.IsAllDay,
StartTimezone = value.StartTimezone,
EndTimezone = value.EndTimezone,
RecurrenceRule = value.RecurrenceRule,
RecurrenceID = value.RecurrenceID,
}
}

```

```

RecurrenceException = value.RecurrenceException
};
db.ScheduleEventDatas.InsertOnSubmit(appointment);
db.SubmitChanges();
}

```



Restricting add action based on specific criteria

In the following example, the specific fields of Scheduler editor window such as Subject and Location are made to undergo validation such that if it is left as blank, then the default **required** validation message will be displayed, while clicking on a save button.

Additionally, the regex condition has been added to the Location field, so that if any special characters are typed into it, then the custom validation message will be displayed.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource">
<ScheduleField>
<FieldSubject Name="Subject" Validation="@ValidationRules"></FieldSubject>
<FieldLocation Name="Location"
Validation="@LocationValidationRules"></FieldLocation>
<FieldDescription Name="Description"
Validation="@DescriptionValidationRules"></FieldDescription>
<FieldStartTime Name="StartTime"
Validation="@ValidationRules"></FieldStartTime>
<FieldEndTime Name="EndTime" Validation="@ValidationRules"></FieldEndTime>
</ScheduleField>
</ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>

```

```

<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
    DateTime CurrentDate = new DateTime(2020, 1, 6);
    static Dictionary<string, object> ValidationMessages = new
    Dictionary<string, object>() { { "regex", "Special character(s) not allowed
    in this field" } };
    ValidationRules ValidationRules = new ValidationRules { Required = true };
    ValidationRules LocationValidationRules = new ValidationRules { Required =
    true, RegexPattern = "^[a-zA-Z0-9- ]*$", Messages = ValidationMessages };
    ValidationRules DescriptionValidationRules = new ValidationRules { Required
    = true, MinLength = 5, MaxLength = 500 };
    List<AppointmentData> DataSource = new List<AppointmentData>
    {
        new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
        DateTime(2020, 1, 6, 9, 30, 0) , EndTime = new DateTime(2020, 1, 6, 11, 0,
        0) }
    };
    public class AppointmentData
    {
        public int Id { get; set; }
        public string Subject { get; set; }
        public string Location { get; set; }
        public DateTime StartTime { get; set; }
        public DateTime EndTime { get; set; }
        public string Description { get; set; }
        public bool IsAllDay { get; set; }
        public string RecurrenceRule { get; set; }
        public string RecurrenceException { get; set; }
        public Nullable<int> RecurrenceID { get; set; }
    }
}

```

You can also dynamically prevent the creation of appointments on Scheduler. For example, say if you want to decline the creation of appointments on weekend days, you can check for its appropriate condition within the `OnActionBegin` event.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Width="100%" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEvents TValue="AppointmentData"
OnActionBegin="OnActionBegin"></ScheduleEvents>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.Month" MaxEventsPerRow="2"></ScheduleView>
</ScheduleViews>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
</SfSchedule>
@code {

```

```

DateTime CurrentDate = new DateTime(2019, 1, 6);
public void OnActionBegin(ActionEventArgs<AppointmentData> args)
{
    if (args.ActionType == ActionType.EventCreate)
    {
        int[] weekEnds = new int[2] { 0, 6 };
        AppointmentData data = args.AddedRecords[0];
        DateTime date = data.StartTime;
        int weekDay = (int)date.DayOfWeek;
        if (weekDay == weekEnds[0] || weekDay == weekEnds[1])
        {
            args.Cancel = true;
        }
    }
}

List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
    DateTime(2019, 1, 7, 11, 0, 0) , EndTime = new DateTime(2019, 1, 7, 12, 30,
    0) },
    new AppointmentData { Id = 2, Subject = "vacation", StartTime = new
    DateTime(2019, 1, 8, 8, 30, 0) , EndTime = new DateTime(2019, 1, 8, 9, 30,
    0) },
    new AppointmentData { Id = 6, Subject = "conference", StartTime = new
    DateTime(2019, 1, 11, 18, 0, 0) , EndTime = new DateTime(2019, 1, 11, 19,
    30, 0) }
};

public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
}

```

Edit

The same way the appointments such as normal, all-day, spanned or recurring events are created, it can be easily edited using any of the following ways.

- [Update using editor window](#)
- [Update using SaveEventAsync method](#)

Update using editor window

The default editor window filled with appointment details can be opened by double clicking on the required events. It gets pre-filled with event options such as Subject, Location, Start and End time, All-

day, timezone, description and other recurrence options, from which you can edit the desired field values and, then enter the **Save** button to update it.

You can also single click on appointments, which opens the quick info popup with edit and delete options. Clicking on the **Edit** option will open the default editor filled with event details and **Delete** option will prompt for delete confirmation.

Update using SaveEventAsync method

The appointments can be edited and updated manually using the **SaveEventAsync** method.

Here, an event with ID **1** is edited and its subject is changed with a new text. When the modified data object is passed onto the **SaveEventAsync** method, the changes gets reflected onto the original event. The **Id** field is mandatory in this edit process, where the modified event object should hold the valid **Id** value that exists in the Scheduler data source.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Buttons
<SfButton Content="EDIT" OnClick="OnClick"></SfButton>
<SfSchedule @ref="ScheduleRef" TValue="AppointmentData" Height="550px"
@bind-SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 6);
SfSchedule<AppointmentData> ScheduleRef;
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Testing", StartTime = new
DateTime(2020, 1, 6, 9, 30, 0) , EndTime = new DateTime(2020, 1, 6, 11, 0,
0)},
new AppointmentData { Id = 2, Subject = "Conference", StartTime = new
DateTime(2020, 1, 11, 9, 30, 0) , EndTime = new DateTime(2020, 1, 11, 11, 0,
0)}
};
public async Task OnClick()
{
AppointmentData eventData = new AppointmentData
{
Id = 1,
Subject = "Edited",
StartTime = new DateTime(2020, 1, 6, 10, 30, 0),
EndTime = new DateTime(2020, 1, 6, 12, 0, 0),
};
await ScheduleRef.SaveEventAsync(eventData);
}
public class AppointmentData
{
}
```

```

public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}

```

Inline editing

Another easier way to edit the appointments is enabling the `AllowInline` property. By single clicking on the appointments, you can edit the Subject of the appointment. Pressing enter key or clicking out of the appointment will edit the existing appointment.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" AllowInline="true"
@bind-SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 6);
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 6, 9, 30, 0) , EndTime = new DateTime(2020, 1, 6, 11, 0,
0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

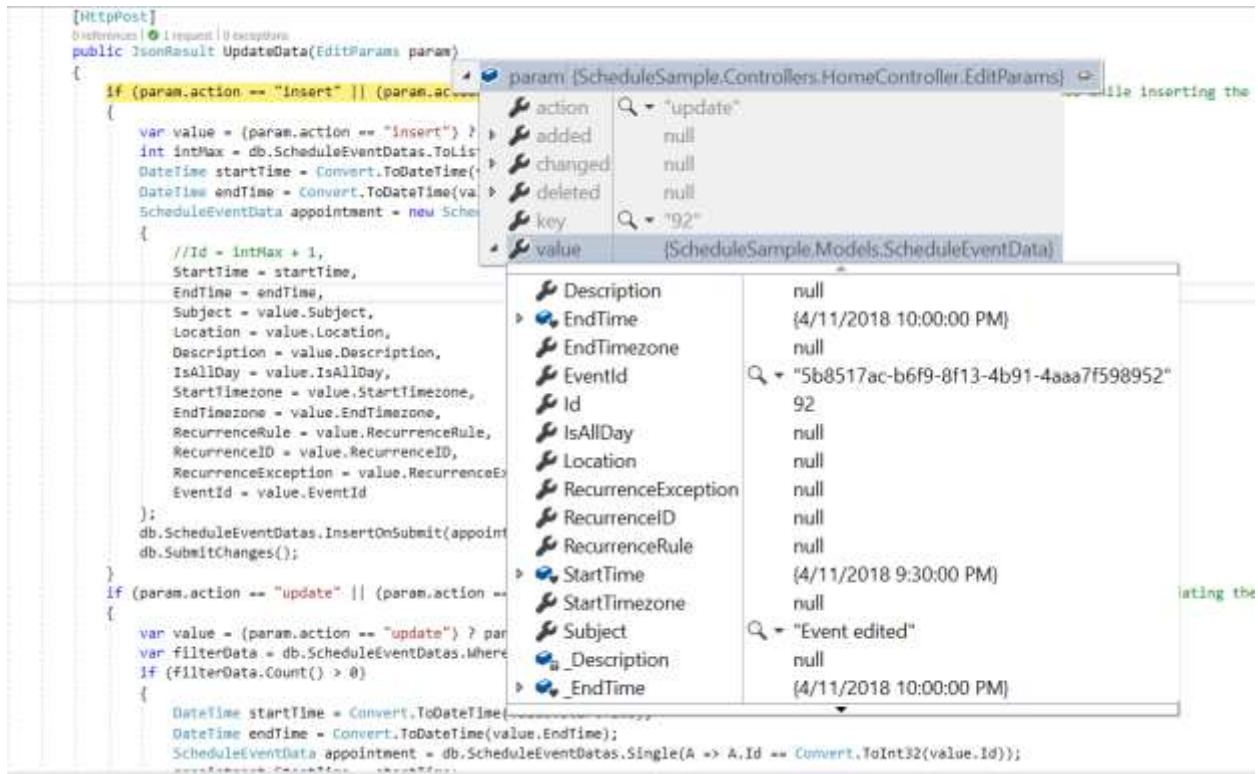
```

Updating events in database at server-side

While editing the normal events in the Scheduler, **update** action takes place and the following code example describes how to update event into database at server side.

SH

```
if (param.action == "update" || (param.action == "batch" && param.changed != null)) // this block of code will execute while updating the appointment
{
    var value = (param.action == "update") ? param.value : param.changed[0];
    var filterData = db.ScheduleEventDatas.Where(c => c.Id == Convert.ToInt32(value.Id));
    if (filterData.Count() > 0)
    {
        DateTime startTime = Convert.ToDateTime(value.StartTime);
        DateTime endTime = Convert.ToDateTime(value.EndTime);
        ScheduleEventData appointment = db.ScheduleEventDatas.Single(A => A.Id == Convert.ToInt32(value.Id));
        appointment.StartTime = startTime;
        appointment.EndTime = endTime;
        appointment.StartTimezone = value.StartTimezone;
        appointment.EndTimezone = value.EndTimezone;
        appointment.Subject = value.Subject;
        appointment.IsAllDay = value.IsAllDay;
        appointment.RecurrenceRule = value.RecurrenceRule;
        appointment.RecurrenceID = value.RecurrenceID;
        appointment.RecurrenceException = value.RecurrenceException;
    }
    db.SubmitChanges();
}
```



How to edit a single occurrence or entire series and update it in database at server-side

The recurring appointments can be edited in either of the following two ways.

- Single occurrence
- Entire series

Editing single occurrence - When a recurring event is double clicked, a popup prompts to choose either to edit the single event or entire series. From this, if you choose to select **EDIT EVENT** option, a single occurrence of the recurring appointment alone will be edited. The following process takes place while editing a single occurrence,

- A new event will be created from the parent event data and added to the Scheduler dataSource, with all its default field values overwritten with the newly modified data and additionally, the **RecurrenceID** field will be added to it, that holds the **id** value of the parent recurring event. Also, a new **Id** will be generated for this event in the dataSource.
- The parent recurring event needs to be updated with appropriate **RecurrenceException** field to hold the edited occurrence appointment's date collection.

Therefore, when a single occurrence is edited from a recurring event, the batch action takes place by allowing both the **Add** and **Edit** action requests to take place together.

In case, if you edit an existing edited occurrence of a recurring event, only those edited occurrence which present in the database as an individual event object will get updated. In this case, **update** action alone takes place on the edited occurrence object on the database.

SH

```

if (param.action == "insert" || (param.action == "batch" && param.added != null)) // this block of code will execute while inserting the appointments
{
    var value = (param.action == "insert") ? param.value : param.added[0];
    int intMax = db.ScheduleEventDatas.Select(x => x.Id).DefaultIfEmpty(0).Max();
    DateTime startTime = Convert.ToDateTime(value.StartTime);
    DateTime endTime = Convert.ToDateTime(value.EndTime);
    ScheduleEventData appointment = new ScheduleEventData()
    {
        Id = intMax + 1,
        StartTime = startTime,
        EndTime = endTime,
        Subject = value.Subject,
        IsAllDay = value.IsAllDay,
        StartTimezone = value.StartTimezone,
        EndTimezone = value.EndTimezone,
        RecurrenceRule = value.RecurrenceRule,
        RecurrenceID = value.RecurrenceID,
        RecurrenceException = value.RecurrenceException
    };
    db.ScheduleEventDatas.InsertOnSubmit(appointment);
    db.SubmitChanges();
}
if (param.action == "update" || (param.action == "batch" && param.changed != null)) // this block of code will execute while updating the appointment
{
    var value = (param.action == "update") ? param.value : param.changed[0];
    var filterData = db.ScheduleEventDatas.Where(c => c.Id == Convert.ToInt32(value.Id));
    if (filterData.Count() > 0)
    {
        DateTime startTime = Convert.ToDateTime(value.StartTime);
        DateTime endTime = Convert.ToDateTime(value.EndTime);
        ScheduleEventData appointment = db.ScheduleEventDatas.Single(A => A.Id == Convert.ToInt32(value.Id));
        appointment.StartTime = startTime;
        appointment.EndTime = endTime;
        appointment.StartTimezone = value.StartTimezone;
        appointment.EndTimezone = value.EndTimezone;
        appointment.Subject = value.Subject;
        appointment.IsAllDay = value.IsAllDay;
        appointment.RecurrenceRule = value.RecurrenceRule;
        appointment.RecurrenceID = value.RecurrenceID;
        appointment.RecurrenceException = value.RecurrenceException;
    }
    db.SubmitChanges();
}

```

Editing entire series - When an option **EDIT SERIES** is selected from the popup that opens on double clicking the recurring event, the whole recurring series will be updated with the newly provided value. When this option is chosen explicitly, if a parent event holds any edited occurrences - then all its child occurrences will be removed from the dataSource and simply the single parent data will be updated.

This action of editing entire series also leads to the batch process, as both the **Delete** and **Edit** action takes place together.

SH

```

if (param.action == "update" || (param.action == "batch" && param.changed !=
null)) // this block of code will execute while updating the appointment
{
    var value = (param.action == "update") ? param.value : param.changed[0];
    var filterData = db.ScheduleEventDatas.Where(c => c.Id ==
Convert.ToInt32(value.Id));
    if (filterData.Count() > 0)
    {
        DateTime startTime = Convert.ToDateTime(value.StartTime);
        DateTime endTime = Convert.ToDateTime(value.EndTime);
        ScheduleEventData appointment = db.ScheduleEventDatas.Single(A => A.Id ==
Convert.ToInt32(value.Id));
        appointment.StartTime = startTime;
        appointment.EndTime = endTime;
        appointment.StartTimezone = value.StartTimezone;
        appointment.EndTimezone = value.EndTimezone;
        appointment.Subject = value.Subject;
        appointment.IsAllDay = value.IsAllDay;
        appointment.RecurrenceRule = value.RecurrenceRule;
        appointment.RecurrenceID = value.RecurrenceID;
        appointment.RecurrenceException = value.RecurrenceException;
    }
    db.SubmitChanges();
}
if (param.action == "remove" || (param.action == "batch" && param.deleted !=
null)) // this block of code will execute while removing the appointment
{
    if (param.action == "remove")
    {
        int key = Convert.ToInt32(param.key);
        ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id ==
key).FirstOrDefault();
        if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
    }
    else
    {
        foreach (var apps in param.deleted)
        {
            ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id ==
apps.Id).FirstOrDefault();
            if (apps != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
        }
    }
    db.SubmitChanges();
}

```

To know more about handling recurrence exceptions, refer the [Adding exceptions](#) topic.

Restricting edit action based on specific criteria

You can also dynamically prevent the editing of appointments on Scheduler. For example, say if you want to decline the updating of appointments on non-working hours, you can check for its appropriate condition within the `OnActionBegin` event.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Width="100%" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleWorkHours Highlight="true" Start="@StartHour"
End="@EndHour"></ScheduleWorkHours>
<ScheduleEvents TValue="AppointmentData"
OnActionBegin="OnActionBegin"></ScheduleEvents>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.Month" MaxEventsPerRow="2"></ScheduleView>
</ScheduleViews>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
</SfSchedule>
@code {
DateTime CurrentDate = new DateTime(2019, 1, 6);
public string StartHour { get; set; } = "09:00";
public string EndHour { get; set; } = "18:00";
public void OnActionBegin(ActionEventArgs<AppointmentData> args)
{
if (args.ActionType == ActionType.EventChange)
{
bool workHours = false, flag = false;
int[] weekEnds = new int[2] { 0, 6 };
AppointmentData data = args.ChangedRecords[0];
DateTime date = data.StartTime;
int weekDay = (int)date.DayOfWeek;
if (weekDay == weekEnds[0] || weekDay == weekEnds[1])
{
flag = true;
}
int hour = data.StartTime.Hour;
int workHoursStart = Convert.ToInt32(StartHour.Substring(0, 2));
int workHoursEnd = Convert.ToInt32(EndHour.Substring(0, 2));
if (workHoursStart <= hour && workHoursEnd > hour)
{
workHours = true;
}
if (flag || !workHours)
{
args.Cancel = true;
}
}
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2019, 1, 6, 9, 0, 0) , EndTime = new DateTime(2019, 1, 6, 11, 30,
0) },
}
```

```

new AppointmentData { Id = 2, Subject = "vacation", StartTime = new
DateTime(2019, 1, 8, 10, 30, 0) , EndTime = new DateTime(2019, 1, 8, 12, 30,
0) },
new AppointmentData { Id = 6, Subject = "conference", StartTime = new
DateTime(2019, 1, 11, 18, 0, 0) , EndTime = new DateTime(2019, 1, 11, 19,
30, 0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

Delete

The appointments can be deleted in either of the following ways,

- Selecting an appointment and clicking the delete icon from the quick popup that opens.
- Selecting an appointment and pressing **Delete** key.
- Selecting multiple appointments by tap holding an event and then continuously single clicking on other consecutive events and then clicking the **Delete** key.
- Double clicking on an event which opens the default event editor pre-filled with event details, and then choosing **Delete** button in it.

While performing all these above mentioned actions, a pop-up with a delete confirmation message will be displayed prompting either to proceed with deleting an appointment.

Deletion using editor window

When you double click an event, the default editor window will be opened which includes a **Delete** button at the bottom left position which allows to delete that particular appointment. When deleting an appointment through this editor window, the delete alert confirmation will not be asked and the event will be deleted immediately.

Deletion using DeleteEventAsync method

The appointments can be removed manually using the **DeleteEventAsync** method. The following code examples shows how to edit the normal and recurring events.

Normal event - You can delete the normal appointments of Scheduler by simply passing its **Id** value or the entire event object collection to the **DeleteEventAsync** method.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Buttons
<SfButton Content="DELETE" OnClick="OnClick"></SfButton>

```



```

<SfSchedule @ref="ScheduleRef" TValue="AppointmentData" Height="550px"
@bind-SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 6);
SfSchedule<AppointmentData> ScheduleRef;
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Testing", StartTime = new
DateTime(2020, 1, 6, 9, 30, 0) , EndTime = new DateTime(2020, 1, 6, 11, 0,
0)},
new AppointmentData { Id = 2, Subject = "Conference", StartTime = new
DateTime(2020, 1, 8, 9, 30, 0) , EndTime = new DateTime(2020, 1, 8, 11, 0,
0)}
};
public async Task OnClick()
{
await ScheduleRef.DeleteEventAsync(2);
}
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

Recurring Event - The recurring events can be removed as an entire series or simply removing single occurrence by using the `DeleteEventAsync` method which takes in either the `DeleteSeries` or `DeleteOccurrence` parameters.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Buttons
<SfButton Content="DELETE SERIES" OnClick="OnClick"></SfButton>
<SfSchedule @ref="ScheduleRef" TValue="AppointmentData" Height="550px"
@bind-SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>

```

```

<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
    DateTime CurrentDate = new DateTime(2020, 1, 6);
    SfSchedule<AppointmentData> ScheduleRef;
    List<AppointmentData> DataSource = new List<AppointmentData>
    {
        new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
        DateTime(2020, 1, 6, 9, 30, 0) , EndTime = new DateTime(2020, 1, 6, 11, 0,
        0),
        RecurrenceRule = "FREQ=DAILY;INTERVAL=1;COUNT=3"},
        new AppointmentData { Id = 2, Subject = "Testing", StartTime = new
        DateTime(2020, 1, 7, 12, 0, 0) , EndTime = new DateTime(2020, 1, 7, 13, 30,
        0),
        RecurrenceRule = "FREQ=DAILY;INTERVAL=1;COUNT=3"}
    };
    public async Task OnClick()
    {
        AppointmentData eventData = new AppointmentData
        {
            Id = 2,
            Subject = "Testing",
            StartTime = new DateTime(2020, 1, 7, 12, 0, 0),
            EndTime = new DateTime(2020, 1, 7, 13, 30, 0),
            RecurrenceID = 2,
            RecurrenceRule = "FREQ=DAILY;INTERVAL=1;COUNT=3"
        };
        await ScheduleRef.DeleteEventAsync(eventData, CurrentAction.DeleteSeries);
    }
    public class AppointmentData
    {
        public int Id { get; set; }
        public string Subject { get; set; }
        public string Location { get; set; }
        public DateTime StartTime { get; set; }
        public DateTime EndTime { get; set; }
        public string Description { get; set; }
        public bool IsAllDay { get; set; }
        public string RecurrenceRule { get; set; }
        public string RecurrenceException { get; set; }
        public Nullable<int> RecurrenceID { get; set; }
    }
}

```

Removing events from database at server-side

While deleting the event from the Scheduler, **remove** action takes place and the following code example describes how to delete event from database at server side.

SH

```

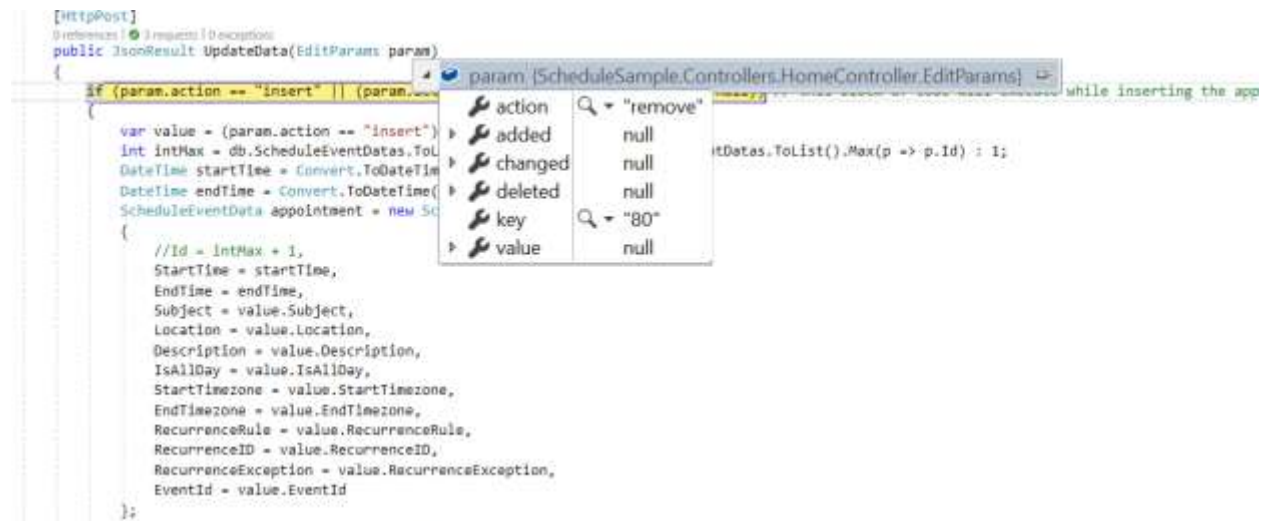
if (param.action == "remove" || (param.action == "batch" && param.deleted !=
null)) // this block of code will execute while removing the appointment

```

```

{
    if (param.action == "remove")
    {
        int key = Convert.ToInt32(param.key);
        ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == key).FirstOrDefault();
        if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
    }
    else
    {
        foreach (var apps in param.deleted)
        {
            ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == apps.Id).FirstOrDefault();
            if (apps != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
        }
    }
    db.SubmitChanges();
}

```



How to delete a single occurrence or entire series from Scheduler and update it in database at server-side
 The recurring events can be deleted in either of the following two ways.

- Single occurrence
- Entire series

Single occurrence - When you attempt to delete the recurring events, a popup prompts you to choose either to delete the single event or entire series. From this, if you choose to select **DELETE EVENT** option, a single occurrence of the recurring appointment alone will be removed. The following process takes place while removing a single occurrence,

- The selected occurrence will be deleted from the Scheduler user interface.
- In code, the parent recurring event object will be updated with appropriate `RecurrenceException` field, to hold the deleted occurrence appointment's date collection.

Therefore, when a single occurrence is deleted from a recurring event, the **update** action takes place on the parent recurring event as shown in the following code example.

In case, if you delete an existing edited occurrence of a recurring event, only those edited occurrence which present in the database as an individual event object will get removed. In this case, **delete** action takes place instead of **update** action and the parent recurring event object remains same with no changes.

SH

```
if (param.action == "update" || (param.action == "batch" && param.changed != null)) // this block of code will execute while updating the appointment
{
    var value = (param.action == "update") ? param.value : param.changed[0];
    var filterData = db.ScheduleEventDatas.Where(c => c.Id == Convert.ToInt32(value.Id));
    if (filterData.Count() > 0)
    {
        DateTime startTime = Convert.ToDateTime(value.StartTime);
        DateTime endTime = Convert.ToDateTime(value.EndTime);
        ScheduleEventData appointment = db.ScheduleEventDatas.Single(A => A.Id == Convert.ToInt32(value.Id));
        appointment.StartTime = startTime;
        appointment.EndTime = endTime;
        appointment.StartTimezone = value.StartTimezone;
        appointment.EndTimezone = value.EndTimezone;
        appointment.Subject = value.Subject;
        appointment.IsAllDay = value.IsAllDay;
        appointment.RecurrenceRule = value.RecurrenceRule;
        appointment.RecurrenceID = value.RecurrenceID;
        appointment.RecurrenceException = value.RecurrenceException;
    }
    db.SubmitChanges();
}
```

Entire series - When you select an option **DELETE SERIES** from the popup, the whole recurring series will be deleted. When this option is chosen explicitly, if a parent event holds any edited occurrences - then all its child occurrences which are maintained as separate event objects will also be removed from the dataSource. This action of deleting entire series leads to **remove** action and removes one or more event objects at the same time.

SH

```
if (param.action == "remove" || (param.action == "batch" && param.deleted != null)) // this block of code will execute while removing the appointment
{
    if (param.action == "remove")
    {
        int key = Convert.ToInt32(param.key);
        ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == key).FirstOrDefault();
        if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
    }
    else
    {

```

```
foreach (var apps in param.deleted)
{
    ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id ==
    apps.Id).FirstOrDefault();
    if (apps != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
}
}
db.SubmitChanges();
}
```

Drag and drop

When you drag and drop a normal event on the Scheduler, the event editing action takes place. When a recurring event is drag and dropped on a desired time range, the batch action explained in [Editing a single occurrence](#) process will take place - thus allowing both the [Add](#) and [Edit](#) action to take place together.

By default, when you drag a recurring instance, only the occurrence of the event gets edited and not a whole series.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Width="100%" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code {
    DateTime CurrentDate = new DateTime(2020, 1, 31);
    List<AppointmentData> DataSource = new List<AppointmentData>
    {
        new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
        DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
        0) }
    };
    public class AppointmentData
    {
        public int Id { get; set; }
        public string Subject { get; set; }
        public string Location { get; set; }
        public DateTime StartTime { get; set; }
        public DateTime EndTime { get; set; }
        public string Description { get; set; }
        public bool IsAllDay { get; set; }
        public string RecurrenceRule { get; set; }
        public string RecurrenceException { get; set; }
        public Nullable<int> RecurrenceID { get; set; }
    }
}
```

Resize

When you resize a normal event on the Scheduler, the event editing action takes place. When a recurring event is resized to a new desired time, the batch action explained in [Editing a single occurrence](#) process will take place - thus allowing both the [Add](#) and [Edit](#) action to take place together.

By default, when you resize a recurring instance, only the occurrence of the event gets edited and not a whole series.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Width="100%" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code {
DateTime CurrentDate = new DateTime(2020, 1, 31);
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}
```

Virtual Scrolling in Blazor Scheduler Component

To achieve better performance in the Scheduler when loading a large number of resources and events, virtual scrolling support has been added in the timeline views to load a large set of resources and events instantly as you scroll. You can dynamically load large number of resources and events in timeline view of the Scheduler by setting [true](#) to the [AllowVirtualScrolling](#) property within the timeline view-specific settings. The virtual loading of events is possible in Agenda view, by setting [AllowVirtualScrolling](#) property to [true](#) within the agenda view specific settings.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="EventData" Width="100%" Height="650px" @bind-
SelectedDate="@CurrentDate">
<ScheduleGroup EnableCompactView="false"
Resources="@GroupData"></ScheduleGroup>
<ScheduleResources>
<ScheduleResource TItem="ResourceData" TValue="int"
DataSource="@ResourceDatasource" Field="ResourceId" Title="Resource"
Name="Resources" TextField="Text" IdField="Id" ColorField="Color"
AllowMultiple="true"></ScheduleResource>
</ScheduleResources>
<ScheduleEventSettings
DataSource="@AppointmentData"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.TimelineMonth"
AllowVirtualScrolling="true"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 4, 1);
static EventData data = new EventData();
public static List<ResourceData> ResourceDatasource =
GenerateResourceData();
public static List<EventData> AppointmentData = GenerateStaticEvents();
public string[] GroupData { get; set; } = { "Resources" };
static public List<ResourceData> GenerateResourceData()
{
List<ResourceData> resources = new List<ResourceData>(300);
var colors = new string[] { "#ff8787", "#9775fa", "#748ffc", "#3bc9db",
"#69db7c",
"#fdd835", "#748ffc", "#9775fa", "#df5286", "#7fa900",
"#fec200", "#5978ee", "#00bdae", "#ea80fc" };
for (int a = 1; a <= 300; a++)
{
int index = a % colors.Length;
index = index == 0 ? (colors.Length / a) : index;
resources.Add(new ResourceData
{
Id = a,
Text = "Resource " + a,
Color = colors[index]
});
}
return resources;
}
public static List<EventData> GenerateStaticEvents()
{
DateTime date = new DateTime(2020, 4, 1);
List<EventData> data = new List<EventData>(3600);
var id = 1;
for (var i = 0; i < 300; i++)
{
Random random = new Random();
List<int> listNumbers = new List<int>();
int[] randomCollection = new int[24];

```

```

int number;
int max = 30;
for (int a = 0; a < 12; a++)
{
    do
    {
        number = random.Next(max);
    } while (listNumbers.Contains(number));
    listNumbers.Add(number);
    var startDate = date.AddDays(number);
    startDate = startDate.AddMilliseconds((((number % 10) * 10) * (1000 * 60)));
    var endDate = startDate.AddMilliseconds(((1440 + 30) * (1000 * 60)));
    data.Add(new EventData
    {
        Id = id,
        Subject = "Event #" + id,
        StartTime = startDate,
        EndTime = endDate,
        IsAllDay = (id % 10 == 0) ? false : true,
        ResourceId = i + 1
    });
    id++;
}
}
return data;
}
public class EventData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public bool IsAllDay { get; set; }
    public int ResourceId { get; set; }
}
public class ResourceData
{
    public int Id { get; set; }
    public string Text { get; set; }
    public string Color { get; set; }
}
}

```

Virtual scrolling with templates

In Blazor Scheduler, templates can be applied when `AllowVirtualScrolling` property is enabled. In the following code, templates were applied to resources and appointments.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="EventData" Width="100%" Height="650px" @bind-
SelectedDate="@CurrentDate">
<ScheduleTemplates>
<ResourceHeaderTemplate>
<div class='template-wrap'>
<div class="resource-details">

```



```

<div class="resource-name">@(((context as TemplateContext).ResourceData as
ResourceData).Text)</div>
<div class="resource-designation">@(((context as
TemplateContext).ResourceData as ResourceData).Designation)</div>
</div>
</div>
</ResourceHeaderTemplate>
</ScheduleTemplates>
<ScheduleGroup EnableCompactView="false"
Resources="@GroupData"></ScheduleGroup>
<ScheduleResources>
<ScheduleResource TValue="int[]" TItem="ResourceData"
DataSource="@ResourceDatasource" Field="ResourceId" Title="Resource"
Name="Resources" TextField="Text" IdField="Id" ColorField="Color"
AllowMultiple="true"></ScheduleResource>
</ScheduleResources>
<ScheduleEventSettings DataSource="@AppointmentData">
<Template>
<div>Subject: @((context as EventData).Subject)</div>
<div>StartTime: @((context as EventData).StartTime.ToUniversalTime())</div>
<div>EndTime: @((context as EventData).EndTime.ToUniversalTime())</div>
</Template>
</ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.TimelineMonth"
AllowVirtualScrolling="true"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 4, 1);
static EventData data = new EventData();
public static List<ResourceData> ResourceDatasource =
GenerateResourceData();
public static List<EventData> AppointmentData = GenerateStaticEvents();
public string[] GroupData { get; set; } = { "Resources" };
static public List<ResourceData> GenerateResourceData()
{
List<ResourceData> resources = new List<ResourceData>(300);
var colors = new string[] { "#ff8787", "#9775fa", "#748ffc", "#3bc9db",
"#69db7c",
"#fdd835", "#748ffc", "#9775fa", "#df5286", "#7fa900",
"#fec200", "#5978ee", "#00bdae", "#ea80fc" };
var designation = new string[] { "Developer", "Lead", "Product Manager",
"QA", "Newbie" };
for (int a = 1; a <= 300; a++)
{
int index = a % colors.Length;
int designationIndex = a % designation.Length;
index = index == 0 ? (colors.Length / a) : index;
designationIndex = designationIndex == 0 ? (designation.Length / a) :
designationIndex;
resources.Add(new ResourceData
{
Id = a,
Text = "Resource " + a,
Color = colors[index],
Designation = designation[designationIndex]
}
}
}

```

```

});
}
return resources;
}
public static List<EventData> GenerateStaticEvents()
{
    DateTime date = new DateTime(2020, 4, 1);
    List<EventData> data = new List<EventData>(3600);
    var id = 1;
    for (var i = 0; i < 300; i++)
    {
        Random random = new Random();
        List<int> listNumbers = new List<int>();
        int[] randomCollection = new int[24];
        int number;
        int max = 30;
        for (int a = 0; a < 12; a++)
        {
            do
            {
                number = random.Next(max);
            } while (listNumbers.Contains(number));
            listNumbers.Add(number);
            var startDate = date.AddDays(number);
            startDate = startDate.AddMilliseconds((((number % 10) * 10) * (1000 * 60)));
            var endDate = startDate.AddMilliseconds(((1440 + 30) * (1000 * 60)));
            data.Add(new EventData
            {
                Id = id,
                Subject = "Event #" + id,
                StartTime = startDate,
                EndTime = endDate,
                IsAllDay = (id % 10 == 0) ? false : true,
                ResourceId = i + 1
            });
            id++;
        }
    }
    return data;
}
public class EventData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public bool IsAllDay { get; set; }
    public int ResourceId { get; set; }
    public ResourceData ResourceData { get; set; }
}
public class ResourceData
{
    public int Id { get; set; }
    public string Text { get; set; }
    public string Color { get; set; }
    public string Designation { get; set; }
}

```

```
}  

```

For now, the virtual loading of resources and events is available only for timeline views. In the future, we plan to port the same virtual loading on all other applicable Scheduler views.

Editor Window Customization in Blazor Scheduler Component

Scheduler makes use of popups and dialog to display the required notifications, as well as includes an editor window with event fields for making the appointment creation and editing process easier. You can also easily customize the editor window and the fields present in it, and can also apply validations on those fields.

Event editor

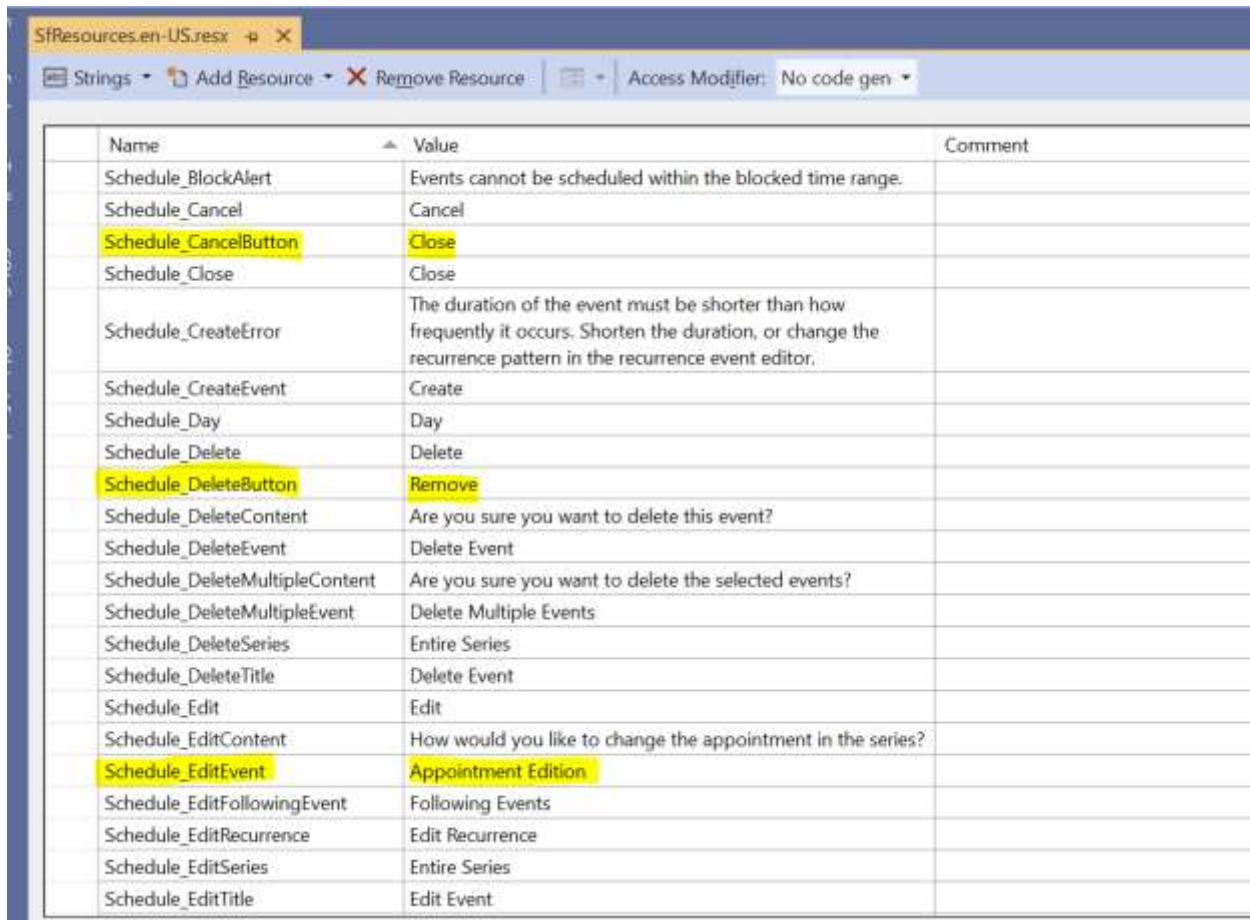
The editor window usually opens on the Scheduler, when a cell or event is double clicked. When a cell is double clicked, the detailed editor window opens in "Add new" mode, whereas when an event is double clicked, the same is opened in an "Edit" mode.

In mobile devices, the detailed editor window can be opened in edit mode by clicking the edit icon on the popup, that opens on single tapping an event. You can also open it in add mode by single tapping a cell, which will display a **+** indication, clicking on it again will open the editor window.

You can also prevent the editor window from opening, by rendering Scheduler in a **Readonly** mode or by doing code customization within the **OnPopupOpen** event.

How to change the editor window header title and text of footer buttons

You can change the header title and the text of buttons displayed at the footer of the editor window by changing the appropriate localized word collection in the resx file of your culture file available in the following directory **Project root folder > Resources > SfResources-en-US.resx** like the below image.



Name	Value	Comment
Schedule_BlockAlert	Events cannot be scheduled within the blocked time range.	
Schedule_Cancel	Cancel	
Schedule_CancelButton	Close	
Schedule_Close	Close	
Schedule_CreateError	The duration of the event must be shorter than how frequently it occurs. Shorten the duration, or change the recurrence pattern in the recurrence event editor.	
Schedule_CreateEvent	Create	
Schedule_Day	Day	
Schedule_Delete	Delete	
Schedule_DeleteButton	Remove	
Schedule_DeleteContent	Are you sure you want to delete this event?	
Schedule_DeleteEvent	Delete Event	
Schedule_DeleteMultipleContent	Are you sure you want to delete the selected events?	
Schedule_DeleteMultipleEvent	Delete Multiple Events	
Schedule_DeleteSeries	Entire Series	
Schedule_DeleteTitle	Delete Event	
Schedule_Edit	Edit	
Schedule_EditContent	How would you like to change the appointment in the series?	
Schedule_EditEvent	Appointment Edition	
Schedule_EditFollowingEvent	Following Events	
Schedule_EditRecurrence	Edit Recurrence	
Schedule_EditSeries	Entire Series	
Schedule_EditTitle	Edit Event	

The editor window opening for creating new event will be displayed as in the following image after changing the localized words.

Add Event
X

Title

Location

Start

End

1/29/20 11:00 AM

1/29/20 11:30 AM

☐ All day

☐ Timezone

Repeat

Never

Description

ADD

CLOSE

How to change the label text of default editor fields

To change the default labels such as Title, Location and other field names in the editor window, make use of the `Title` property available within the field option of `ScheduleEventSettings`.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource">
<ScheduleField>
<FieldSubject Title="Travel Summary"></FieldSubject>
<FieldLocation Title="Source"></FieldLocation>
<FieldDescription Title="Comments"></FieldDescription>
<FieldIsAllDay Title="Full Day"></FieldIsAllDay>
<FieldStartTime Title="Departure Time"></FieldStartTime>
<FieldEndTime Title="Arrival Time"></FieldEndTime>
<FieldStartTimezone Title="Origin"></FieldStartTimezone>
<FieldEndTimezone Title="Destination"></FieldEndTimezone>
</ScheduleField>
</ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
```

```

</SfSchedule>
@code{
    DateTime CurrentDate = new DateTime(2020, 1, 31);
    List<AppointmentData> DataSource = new List<AppointmentData>
    {
        new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
        DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
        0) }
    };
    public class AppointmentData
    {
        public int Id { get; set; }
        public string Subject { get; set; }
        public DateTime StartTime { get; set; }
        public DateTime EndTime { get; set; }
        public bool IsAllDay { get; set; }
        public string RecurrenceRule { get; set; }
        public Nullable<int> RecurrenceID { get; set; }
        public string RecurrenceException { get; set; }
    }
}

```

Edit Event



Travel Summary

Source

Meeting

Departure Time

Arrival Time

1/31/20 9:30 AM



1/31/20 11:00 AM



☐ Full Day ☒ Timezone

Origin

Destination

Asia/Calcutta



Asia/Calcutta



Repeat

Never



Comments

DELETE

SAVE

CANCEL

Field validation

It is possible to validate the required fields of the editor window before submitting it, by adding appropriate validation rules to each field. In the following code example, validation are applied to Subject, Location and Description fields.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource">
<ScheduleField>
<FieldSubject Name="Subject" Validation="@ValidationRules"></FieldSubject>
<FieldLocation Name="Location"
Validation="@LocationValidationRules"></FieldLocation>
<FieldDescription Name="Description"
Validation="@DescriptionValidationRules"></FieldDescription>
<FieldStartTime Name="StartTime"></FieldStartTime>
<FieldEndTime Name="EndTime"></FieldEndTime>
</ScheduleField>
</ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 6);
static Dictionary<string, object> ValidationMessages = new
Dictionary<string, object>() { { "regex", "Special character(s) not allowed
in this field" } };
ValidationRules ValidationRules = new ValidationRules { Required = true };
ValidationRules LocationValidationRules = new ValidationRules { Required =
true, RegexPattern = "^[a-zA-Z0-9- ]*$", Messages = ValidationMessages };
ValidationRules DescriptionValidationRules = new ValidationRules { Required
= true, MinLength = 5, MaxLength = 500 };
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 6, 9, 30, 0) , EndTime = new DateTime(2020, 1, 6, 11, 0,
0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}
```

New Event [X]

Title _____

Location _____

Start **This field is required.** 1/9/20 10:00 AM [Calendar Icon] [Clock Icon]

End **Special character(s) not allowed in this field** 1/9/20 10:30 AM [Calendar Icon] [Clock Icon]

☐ All day ☐ Timezone

Repeat
Never [Dropdown Arrow]

Description _____

This field is required.

SAVE CANCEL

Customizing the default time duration in editor window

In default event editor window, start and end time duration are processed based on the **Interval** value set within the **ScheduleTimeScale** property. By default, **Interval** value is set to 30, and therefore the start and end time duration within the event editor will be in a 30 minutes time difference. This duration value can be changed by changing the **Duration** option within the **OnPopupOpen** event.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEvents TValue="AppointmentData"
OnPopupOpen="@OnPopupOpen"></ScheduleEvents>
<ScheduleEventSettings DataSource="@DataSource">
</ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
```



```

DateTime CurrentDate = new DateTime(2020, 1, 31);
public void OnPopupOpen(PopupOpenEventArgs<AppointmentData> args)
{
    if (args.Type == PopupType.Editor)
    {
        args.Duration = 60;
    }
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
    DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
    0) }
};
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
    public string RecurrenceException { get; set; }
}
}

```

How to prevent the display of editor and quick popups

It is possible to prevent the display of editor and quick popup windows by passing the value `true` to `cancel` option within the `OnPopupOpen` event.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEvents TValue="AppointmentData"
OnPopupOpen="@OnPopupOpen"></ScheduleEvents>
<ScheduleEventSettings DataSource="@DataSource">
</ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
private DateTime CurrentDate = new DateTime(2020, 1, 31);
public void OnPopupOpen(PopupOpenEventArgs<AppointmentData> args)
{
    if (args.Type == PopupType.Editor || args.Type == PopupType.QuickInfo)
    {
        args.Cancel = true;
    }
}
}

```

```

}
List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
    DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
    0) }
};
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
    public string RecurrenceException { get; set; }
}
}

```

In case, in order to prevent only specific popups on Scheduler, check the condition based on the popup type. The types of the popup that can be checked within the `OnPopupOpen` event are as follows.

Type	Description
Editor	For Detailed editor window.
QuickInfo	For Quick popup which opens on cell click.
EditEventInfo	For Quick popup which opens on event click.
ViewEventInfo	For Quick popup which opens on responsive mode.
EventContainer	For more event indicator popup.
RecurrenceAlert	For edit recurrence event alert popup.
DeleteAlert	For delete confirmation popup.
ValidationAlert	For validation alert popup.
RecurrenceValidationAlert	For recurrence validation alert popup.

How to open editor window manually

It is possible to open the editor window manually for a specific time or certain events by using the `OpenEditorAsync` method which allows the `TValue` or `CellClickEventArgs` and `CurrentAction` as parameters.

[Here](#) is the example to open the editor window on a single click.

Customizing event editor using template

The event editor window can be customized by making use of the `EditorTemplate` option. Each field defined within template must use two way binding for the `Value` property of the components used within the template to perform CRUD actions.

To get start quickly on customizing editor window using template, you can check on this video:

{% youtube

"youtube:https://www.youtube.com/watch?v=t0v8rCEP7ps"%}

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Calendars
@using Syncfusion.Blazor.DropDowns
@using Syncfusion.Blazor.Inputs
<SfSchedule TValue="AppointmentData" Width="100%" Height="650px" @bind-
SelectedDate="@CurrentDate">
<ScheduleTemplates>
<EditorTemplate>
<table class="custom-event-editor" width="100%" cellpadding="5">
<tbody>
<tr>
<td class="e-textlabel">Summary</td>
<td colspan="4">
<SfTextBox @bind-Value="@((context as
AppointmentData).Subject)"></SfTextBox>
</td>
</tr>
<tr>
<td class="e-textlabel">Status</td>
<td colspan="4">
<SfDropDownList ID="EventType" DataSource="@StatusData" Placeholder="Choose
status" @bind-Value="@((context as AppointmentData).EventType)">
<DropDownListFieldSettings Value="Id"></DropDownListFieldSettings>
</SfDropDownList>
</td>
</tr>
<tr>
<td class="e-textlabel">From</td>
<td colspan="4">
<SfDateTimePicker @bind-Value="@((context as
AppointmentData).StartTime)"></SfDateTimePicker>
</td>
</tr>
<tr>
<td class="e-textlabel">To</td>
<td colspan="4">
<SfDateTimePicker @bind-Value="@((context as
AppointmentData).EndTime)"></SfDateTimePicker>
</td>
</tr>
<tr>
<td class="e-textlabel">Reason</td>
<td colspan="4">
<SfTextBox Multiline="true" @bind-Value="@((context as
AppointmentData).Description)"></SfTextBox>
</td>
</tr>
</tbody>
</table>
</EditorTemplate>
```

```

</ScheduleTemplates>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 31);
public class DDFields
{
public string Id { get; set; }
public string Text { get; set; }
}
List<DDFields> StatusData = new List<DDFields>() {
new DDFields(){ Id= "New", Text= "New" },
new DDFields(){ Id= "Requested", Text= "Requested" },
new DDFields(){ Id= "Confirmed", Text= "Confirmed" },
};
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
0), EventType = "Confirmed" }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public string EventType { get; set; }
}
}

```

How to add resource options within editor template

The resource field can be added within editor template with the following code example.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Calendars
@using Syncfusion.Blazor.DropDowns
@using Syncfusion.Blazor.Inputs
<SfSchedule TValue="AppointmentData" Width="100%" Height="650px" @bind-
SelectedDate="@CurrentDate">
<ScheduleGroup Resources="@Resources"></ScheduleGroup>
<ScheduleResources>
<ScheduleResource TItem="ResourceData" TValue="int" DataSource="@OwnersData"
Field="OwnerId" Title="Owner" Name="Owners" TextField="OwnerText"
IdField="Id" ColorField="OwnerColor"></ScheduleResource>
</ScheduleResources>

```

```

<ScheduleTemplates>
<EditorTemplate>
<table class="custom-event-editor" width="100%" cellpadding="5">
<tbody>
<tr>
<td class="e-textlabel">Summary</td>
<td colspan="4">
<SfTextBox @bind-Value="@((context as
AppointmentData).Subject)"></SfTextBox>
</td>
</tr>
<tr>
<td class="e-textlabel">Status</td>
<td colspan="4">
<SfDropDownList TValue="int" TItem="ResourceData" ID="OwnerId"
DataSource="@OwnersData" Placeholder="Choose owner" @bind-Value="@((context
as AppointmentData).OwnerId)">
<DropDownListFieldSettings Text="OwnerText"
Value="Id"></DropDownListFieldSettings>
</SfDropDownList>
</td>
</tr>
<tr>
<td class="e-textlabel">From</td>
<td colspan="4">
<SfDateTimePicker @bind-Value="@((context as
AppointmentData).StartTime)"></SfDateTimePicker>
</td>
</tr>
<tr>
<td class="e-textlabel">To</td>
<td colspan="4">
<SfDateTimePicker @bind-Value="@((context as
AppointmentData).EndTime)"></SfDateTimePicker>
</td>
</tr>
<tr>
<td class="e-textlabel">Reason</td>
<td colspan="4">
<SfTextBox Multiline="true" @bind-Value="@((context as
AppointmentData).Description)"></SfTextBox>
</td>
</tr>
</tbody>
</table>
</EditorTemplate>
</ScheduleTemplates>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfScheduler>
@code{

```

```

DateTime CurrentDate = new DateTime(2020, 1, 31);
public string[] Resources { get; set; } = { "Owners" };
public List<ResourceData> OwnersData { get; set; } = new List<ResourceData>
{
    new ResourceData { OwnerText = "Nancy", Id = 1, OwnerColor = "#ffaa00" },
    new ResourceData { OwnerText = "Steven", Id = 2, OwnerColor = "#f8a398" },
    new ResourceData { OwnerText = "Michael", Id = 3, OwnerColor = "#7499e1" }
};
List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
    DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
    0), OwnerId = 1 }
};
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public int OwnerId { get; set; }
}
public class ResourceData
{
    public int Id { get; set; }
    public string OwnerText { get; set; }
    public string OwnerColor { get; set; }
}
}

```

EditorTemplate is not applicable when we set **AllowMutiple** as true without enabling **AllowGroupEdit**, so in that case use custom editor window.

How to add recurrence options within editor template

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Calendars
@using Syncfusion.Blazor.DropDowns
@using Syncfusion.Blazor.Inputs
<SfSchedule TValue="AppointmentData" Width="100%" Height="650px" @bind-
SelectedDate="@CurrentDate">
    <ScheduleTemplates>
        <EditorTemplate>
            <table class="custom-event-editor" width="100%" cellpadding="5">
                <tbody>
                    <tr>
                        <td class="e-textlabel">Summary</td>
                        <td colspan="4">
                            <SfTextBox @bind-Value="@((context as
                            AppointmentData).Subject)"></SfTextBox>
                        </td>
                    </tr>
                    <tr>
                        <td class="e-textlabel">Status</td>

```

```

<td colspan="4">
<SfDropDownList ID="EventType" DataSource="@StatusData" Placeholder="Choose
status" @bind-Value="@((context as AppointmentData).EventType)">
<DropDownListFieldSettings Value="Id"></DropDownListFieldSettings>
</SfDropDownList>
</td>
</tr>
<tr>
<td class="e-textlabel">From</td>
<td colspan="4">
<SfDateTimePicker @bind-Value="@((context as
AppointmentData).StartTime)"></SfDateTimePicker>
</td>
</tr>
<tr>
<td class="e-textlabel">To</td>
<td colspan="4">
<SfDateTimePicker @bind-Value="@((context as
AppointmentData).EndTime)"></SfDateTimePicker>
</td>
</tr>
<tr>
<td class="e-textlabel">Reason</td>
<td colspan="4">
<SfTextBox Multiline="true" @bind-Value="@((context as
AppointmentData).Description)"></SfTextBox>
</td>
</tr>
<tr>
<td class="e-textlabel">Recurrence</td>
<td colspan="4">
<SfRecurrenceEditor @bind-Value="@((context as
AppointmentData).RecurrenceRule)"></SfRecurrenceEditor>
</td>
</tr>
</tbody>
</table>
</EditorTemplate>
</ScheduleTemplates>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
private DateTime CurrentDate = new DateTime(2020, 1, 31);
public class DDFields
{
public string Id { get; set; }
public string Text { get; set; }
}
List<DDFields> StatusData = new List<DDFields>()
{

```

```

new DDFields(){ Id= "New", Text= "New" },
new DDFields(){ Id= "Requested", Text= "Requested" },
new DDFields(){ Id= "Confirmed", Text= "Confirmed" },
};
List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
    DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
    0), EventType = "Confirmed" }
};
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
    public string RecurrenceException { get; set; }
    public string Description { get; set; }
    public string EventType { get; set; }
}
}

```

Apply validations on editor template fields

In the following code example, validation has been added to the **EventType** field by importing **DataAnnotations** namespace and that field is set as **Required** and displays the validation message for this field by using the **ValidationMessage** tag.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Calendars
@using Syncfusion.Blazor.DropDowns
@using Syncfusion.Blazor.Inputs
@using System.ComponentModel.DataAnnotations
<SfSchedule TValue="AppointmentData" Width="100%" Height="650px" @bind-
SelectedDate="@CurrentDate">
<ScheduleTemplates>
<EditorTemplate>
<table class="custom-event-editor" width="100%" cellpadding="5">
<tbody>
<tr>
<td class="e-textlabel">Summary</td>
<td colspan="4">
<SfTextBox @bind-Value="@((context as
AppointmentData).Subject)"></SfTextBox>
</td>
</tr>
<tr>
<td class="e-textlabel">Status</td>
<td colspan="4">
<SfDropDownList ID="EventType" DataSource="@StatusData" Placeholder="Choose
status" @bind-Value="@((context as AppointmentData).EventType)">

```



```

<DropDownListFieldSettings Value="Id"></DropDownListFieldSettings>
</SfDropDownList>
<ValidationMessage For="()=>((context as AppointmentData).EventType)"/>
</td>
</tr>
<tr>
<td class="e-textlabel">From</td>
<td colspan="4">
<SfDateTimePicker @bind-Value="@((context as
AppointmentData).StartTime)"></SfDateTimePicker>
</td>
</tr>
<tr>
<td class="e-textlabel">To</td>
<td colspan="4">
<SfDateTimePicker @bind-Value="@((context as
AppointmentData).EndTime)"></SfDateTimePicker>
</td>
</tr>
<tr>
<td class="e-textlabel">Reason</td>
<td colspan="4">
<SfTextBox Multiline="true" @bind-Value="@((context as
AppointmentData).Description)"></SfTextBox>
</td>
</tr>
<tr>
<td class="e-textlabel">Recurrence</td>
<td colspan="4">
<SfRecurrenceEditor @bind-Value="@((context as
AppointmentData).RecurrenceRule)"></SfRecurrenceEditor>
</td>
</tr>
</tbody>
</table>
</EditorTemplate>
</ScheduleTemplates>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
private DateTime CurrentDate = new DateTime(2020, 1, 31);
public class DDFields
{
public string Id { get; set; }
public string Text { get; set; }
}
List<DDFields> StatusData = new List<DDFields>() {
new DDFields(){ Id= "New", Text= "New" },
new DDFields(){ Id= "Requested", Text= "Requested" },
new DDFields(){ Id= "Confirmed", Text= "Confirmed" },

```

```

};
Dictionary<string, object> StartName = new Dictionary<string, object>()
{
    {"data-name", "StartTime"},
};
List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
    DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
    0), EventType = "Confirmed" }
};
public class AppointmentData
{
    [Required]
    public string EventType { get; set; }
    public int Id { get; set; }
    public string Subject { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
    public string RecurrenceException { get; set; }
    public string Description { get; set; }
}
}

```

Quick popups

The quick info popups are the ones that gets opened, when a cell or appointment is single clicked on the desktop mode. On single clicking a cell, you can simply provide a subject and save it. Also, while single clicking on an event, a popup will be displayed where you can get the overview of the event information. You can also edit or delete those events through the options available in it.

By default, these popups are displayed over cells and appointments of Scheduler and to disable this action, set `false` to `ShowQuickInfo` property.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate" ShowQuickInfo="false">
<ScheduleEventSettings DataSource="@DataSource">
</ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 31);
List<AppointmentData> DataSource = new List<AppointmentData>
{

```

```

new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public Nullable<int> RecurrenceID { get; set; }
public string RecurrenceException { get; set; }
}
}

```

The quick popup that opens while single clicking on the cells are not applicable on mobile devices.

How to change the watermark text of quick popup subject

By default, **Add Title** text is displayed on the subject field of quick popup. To change the default watermark text, you can change the value of the appropriate localized word collection in the resx file of your culture file.

Customizing quick popups

The look and feel of the built-in quick popup window, which opens when single clicked on the cells or appointments can be customized by making use of the **ScheduleQuickInfoTemplates** option of the Scheduler. There are 3 sub-options available to customize them easily,

- Header - Accepts the template design that customizes the header part of the quick popup.
- Content - Accepts the template design that customizes the content part of the quick popup.
- Footer - Accepts the template design that customizes the footer part of the quick popup.

The quick popup accepts the template that customizes both the cell click quick popup and event click quick popup or only cell click quick popup or event click quick popup by making use of **TemplateType** option which is **TemplateType.Both** by default and also accepts **TemplateType.Cell** or **TemplateType.Event** value.

Customizing quick popup on cell

The quick popup accepts the template that customizes quick popup only on cell by giving **TemplateType.Cell** to the **TemplateType** option in **ScheduleQuickInfoTemplates**.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Inputs
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.DropDowns
@using System.Globalization
<SfSchedule TValue="AppointmentData" @ref="ScheduleRef" CssClass="quick-info"
Width="100%" Height="650px" SelectedDate="@ (new DateTime(2020, 1, 9)) ">
<ScheduleQuickInfoTemplates TemplateType="TemplateType.Cell">
<HeaderTemplate>

```

```

<div class="quick-info-header">
<div class="quick-info-header-content" style="align-items: center; color:
#919191;">
<div class="quick-info-title">
Add Appointment
</div>
<div class="duration-text">@(GetEventDetails((context as
AppointmentData)))</div>
</div>
</div>
</HeaderTemplate>
<ContentTemplate>
<div class="e-cell-content">
<div class="content-area">
<SfTextBox @ref="SubjectRef" Value="@((context as AppointmentData).Subject)"
Placeholder="Title"></SfTextBox>
</div>
<div class="content-area">
<SfDropDownList @ref="EventTypeRef" TValue="int" TItem="RoomsData" Index="0"
DataSource="@ResourceData" Placeholder="Choose Type">
<DropDownListFieldSettings Text="Name"
Value="Id"></DropDownListFieldSettings>
</SfDropDownList>
</div>
<div class="content-area">
<SfTextBox @ref="DescriptionRef" Value="@((context as
AppointmentData).Description)" Placeholder="Notes"></SfTextBox>
</div>
</div>
</ContentTemplate>
<FooterTemplate>
<div class="cell-footer">
<SfButton Content="More Details" OnClick="@ (e => OnMoreDetailsClick(e,
context as AppointmentData)) "></SfButton>
<SfButton Content="Add" IsPrimary="true" OnClick="@ (e => OnAdd(e, context as
AppointmentData)) "></SfButton>
</div>
</FooterTemplate>
</ScheduleQuickInfoTemplates>
<ScheduleViews>
<ScheduleView Option="View.Week"></ScheduleView>
</ScheduleViews>
<ScheduleResources>
<ScheduleResource TValue="int" TItem="RoomsData" DataSource="@ResourceData"
Field="RoomId" Title="Room Type" Name="MeetingRoom" TextField="Name"
IdField="Id" ColorField="Color" AllowMultiple="false"></ScheduleResource>
</ScheduleResources>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
</SfSchedule>
@code{
SfSchedule<AppointmentData> SheduleRef;
SfDropDownList<int, RoomsData> EventTypeRef;
SfTextBox SubjectRef;
SfTextBox DescriptionRef;
private string GetEventDetails(AppointmentData data)
{

```

```

return data.StartTime.ToString("dddd dd, MMMM yyyy",
CultureInfo.InvariantCulture) + " (" + data.StartTime.ToString("hh:mm tt",
CultureInfo.InvariantCulture) + "-" + data.EndTime.ToString("hh:mm tt",
CultureInfo.InvariantCulture) + ")";
}
private async void OnMoreDetailsClick(MouseEventArgs args, AppointmentData
data)
{
await SheduleRef.CloseQuickInfoPopupAsync();
AppointmentData eventData = new AppointmentData
{
Id = new Random().Next(1000),
Subject = SubjectRef.Value ?? "",
StartTime = data.StartTime,
EndTime = data.EndTime,
Location = data.Location,
Description = DescriptionRef.Value ?? "",
IsAllDay = data.IsAllDay,
RoomId = EventTypeRef.Value,
RecurrenceException = data.RecurrenceException,
RecurrenceID = data.RecurrenceID,
RecurrenceRule = data.RecurrenceRule
};
await SheduleRef.OpenEditorAsync(eventData, CurrentAction.Add);
}
private async Task OnAdd(MouseEventArgs args, AppointmentData data)
{
await SheduleRef.CloseQuickInfoPopupAsync();
AppointmentData cloneData = new AppointmentData
{
Id = new Random().Next(1000),
Subject = SubjectRef.Value ?? "",
Description = DescriptionRef.Value ?? "",
StartTime = data.StartTime,
EndTime = data.EndTime,
RoomId = EventTypeRef.Value,
Location = data.Location,
IsAllDay = data.IsAllDay,
RecurrenceException = data.RecurrenceException,
RecurrenceID = data.RecurrenceID,
RecurrenceRule = data.RecurrenceRule
};
await SheduleRef.AddEventAsync(cloneData);
}
public List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Board Meeting", Description =
"Meeting to discuss business goal of 2020.", StartTime = new DateTime(2020,
1, 5, 9, 30, 0), EndTime = new DateTime(2020, 1, 5, 11, 0, 0), RoomId = 1},
new AppointmentData { Id = 2, Subject = "Training session on JSP",
Description = "Knowledge sharing on JSP topics.", StartTime = new
DateTime(2020, 1, 7, 9, 30, 0), EndTime = new DateTime(2020, 1, 7, 11, 0,
0), RoomId = 2}
};
private List<RoomsData> ResourceData { get; set; } = new List<RoomsData> {
new RoomsData { Name = "Jammy", Id = 1, Color = "#ea7a57", Capacity = 20,
Type = "Conference" },

```

```

new RoomsData { Name = "Tweety", Id = 2, Color = "#7fa900", Capacity = 7,
Type = "Cabin" },
new RoomsData { Name = "Nestle", Id = 3, Color = "#5978ee", Capacity = 5,
Type = "Cabin" }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public string Description { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public Nullable<bool> IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public Nullable<int> RecurrenceID { get; set; }
public string RecurrenceException { get; set; }
public int RoomId { get; set; }
}
public class RoomsData
{
public string Name { get; set; }
public int Id { get; set; }
public int Capacity { get; set; }
public string Color { get; set; }
public string Type { get; set; }
}
}
<style>
/*HeaderStyles*/
.quick-info-header {
background-color: white;
padding: 8px 18px;
}
.quick-info-header-content {
justify-content: flex-end;
display: flex;
flex-direction: column;
padding: 5px 10px 5px;
}
.quick-info-title {
font-weight: 500;
font-size: 16px;
letter-spacing: 0.48px;
height: 22px;
}
.duration-text {
font-size: 11px;
letter-spacing: 0.33px;
height: 14px;
}
/*ContentStyles*/
.content-area {
padding: 10px;
width: 100%;
}
.meeting-type-wrap, .meeting-subject-wrap, .notes-wrap {

```

```

font-size: 11px;
color: #666;
letter-spacing: 0.33px;
height: 24px;
padding: 5px;
}
.quick-info .e-popup-content {
}
/*FooterStyles*/
.cell-footer.e-btn {
background-color: #ffffff;
border-color: #878787;
color: #878787;
}
.cell-footer {
padding-top: 10px;
}
.e-quick-popup-wrapper .e-cell-popup .e-popup-content {
padding: 0 14px;
}
</style>

```

Customizing quick popup on event

The quick popup accepts the template that customizes quick popup only on event by giving `TemplateType.Event` to the `TemplateType` option in `ScheduleQuickInfoTemplates`.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Buttons
@using System.Globalization
<SfSchedule TValue="AppointmentData" @ref="SheduleRef" CssClass="quick-info"
Width="100%" Height="650px" SelectedDate="@ (new DateTime(2020, 1, 9))">
<ScheduleQuickInfoTemplates TemplateType="TemplateType.Event">
<HeaderTemplate>
<div class="quick-info-header">
<div class="quick-info-header-content" style="@ ("background:" +
this.ResourceData.Where(item => item.Id.Equals((context as
AppointmentData).RoomId)).FirstOrDefault().Color + "; color: #FFFFFF;")">
<div class="quick-info-title">
Appointment Details
</div>
<div class="duration-text">@(GetEventDetails((context as
AppointmentData)))</div>
</div>
</div>
</HeaderTemplate>
<ContentTemplate>
@{
AppointmentData Data = context as AppointmentData;
<div class="event-content">
<div class="meeting-type-wrap">
<label>Subject</label>:
<span>@(Data.Subject)</span>
</div>
<div class="meeting-subject-wrap">

```

```

<label>Type</label>:
<span>@((Data.RoomId != 0) ? ResourceData.Where(item =>
item.Id.Equals(Data.RoomId)).FirstOrDefault().Name : "")</span>
</div>
<div class="notes-wrap">
<label>Notes</label>:
<span>@(Data.Description)</span>
</div>
</div>
}
</ContentTemplate>
<FooterTemplate>
<div class="event-footer">
<SfButton IsPrimary="true" Content="More Details" OnClick="@ (e =>
OnMoreDetailsClick(e, context as AppointmentData)) "></SfButton>
</div>
</FooterTemplate>
</ScheduleQuickInfoTemplates>
<ScheduleViews>
<ScheduleView Option="View.Week"></ScheduleView>
</ScheduleViews>
<ScheduleResources>
<ScheduleResource TValue="int" TItem="RoomsData" DataSource="@ResourceData"
Field="RoomId" Title="Room Type" Name="MeetingRoom" TextField="Name"
IdField="Id" ColorField="Color" AllowMultiple="false"></ScheduleResource>
</ScheduleResources>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
</SfSchedule>
@code{
SfSchedule<AppointmentData> SheduleRef;
private string GetEventDetails(AppointmentData data)
{
return data.StartTime.ToString("dddd dd, MMMM yyyy",
CultureInfo.InvariantCulture) + " (" + data.StartTime.ToString("hh:mm tt",
CultureInfo.InvariantCulture) + "-" + data.EndTime.ToString("hh:mm tt",
CultureInfo.InvariantCulture) + ")";
}
private async void OnMoreDetailsClick(MouseEventArgs args, AppointmentData
data)
{
await SheduleRef.CloseQuickInfoPopupAsync();
AppointmentData eventData = new AppointmentData
{
Id = data.Id,
Subject = data.Subject,
Location = data.Location,
Description = data.Description,
StartTime = data.StartTime,
EndTime = data.EndTime,
IsAllDay = data.IsAllDay,
RoomId = data.RoomId,
RecurrenceException = data.RecurrenceException,
RecurrenceID = data.RecurrenceID,
RecurrenceRule = data.RecurrenceRule
};
await SheduleRef.OpenEditorAsync(eventData, CurrentAction.Save);
}
}

```



```

public List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData { Id = 1, Subject = "Board Meeting", Description =
    "Meeting to discuss business goal of 2020.", StartTime = new DateTime(2020,
    1, 5, 9, 30, 0), EndTime = new DateTime(2020, 1, 5, 11, 0, 0), RoomId = 1},
    new AppointmentData { Id = 2, Subject = "Training session on JSP",
    Description = "Knowledge sharing on JSP topics.", StartTime = new
    DateTime(2020, 1, 7, 9, 30, 0), EndTime = new DateTime(2020, 1, 7, 11, 0,
    0), RoomId = 2}
};
private List<RoomsData> ResourceData { get; set; } = new List<RoomsData> {
    new RoomsData { Name = "Jammy", Id = 1, Color = "#ea7a57", Capacity = 20,
    Type = "Conference" },
    new RoomsData { Name = "Tweety", Id = 2, Color = "#7fa900", Capacity = 7,
    Type = "Cabin" },
    new RoomsData { Name = "Nestle", Id = 3, Color = "#5978ee", Capacity = 5,
    Type = "Cabin" }
};
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public string Description { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public Nullable<bool> IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
    public string RecurrenceException { get; set; }
    public int RoomId { get; set; }
}
public class RoomsData
{
    public string Name { get; set; }
    public int Id { get; set; }
    public int Capacity { get; set; }
    public string Color { get; set; }
    public string Type { get; set; }
}
<style>
/*HeaderStyles*/
.quick-info-header {
background-color: white;
padding: 8px 18px;
}
.quick-info-header-content {
justify-content: flex-end;
display: flex;
flex-direction: column;
padding: 5px 10px 5px;
}
.quick-info-title {
font-weight: 500;
font-size: 16px;
letter-spacing: 0.48px;

```

```

height: 22px;
}
.duration-text {
font-size: 11px;
letter-spacing: 0.33px;
height: 14px;
}
/*ContentStyles*/
.content-area {
padding: 10px;
width: 100%;
}
.event-content {
height: 90px;
display: flex;
flex-direction: column;
justify-content: center;
padding: 0 15px;
}
.meeting-type-wrap, .meeting-subject-wrap, .notes-wrap {
font-size: 11px;
color: #666;
letter-spacing: 0.33px;
height: 24px;
padding: 5px;
}
.event-content div label {
display: inline-block;
min-width: 45px;
color: #666;
}
.event-content div span {
font-size: 11px;
color: #151515;
letter-spacing: 0.33px;
line-height: 14px;
padding-left: 8px;
}
.quick-info .e-popup-content {
}
</style>

```

Customizing the different combinations for cell and event quick popups

The quick popup accepts the template that customizes only event click quick popup by giving **TemplateType.Both** to the **TemplateType** option in **ScheduleQuickInfoTemplates**.

You can also do different customization for quick popup on cell and event by checking the **ElementType** option within the event. You can check **ElementType** with any of the following.

Element type	Description
----- -----	
cell	customizes cell popup.
event	customized event popup.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Inputs
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.DropDowns
@using System.Globalization

<SfSchedule TValue="AppointmentData" @ref="ScheduleRef" CssClass="quick-info"
Width="100%" Height="650px" SelectedDate="@ (new DateTime(2020, 1, 9))">
<ScheduleQuickInfoTemplates>
<HeaderTemplate>
<div class="quick-info-header">
<div class="quick-info-header-content" style="@ (GetHeaderStyles((context as
AppointmentData)))">
<div class="quick-info-title">
@((context as AppointmentData).ElementType == "cell" ? "Add Appointment" :
"Appointment Details")
</div>
<div class="duration-text">@(GetEventDetails((context as
AppointmentData)))</div>
</div>
</div>
</HeaderTemplate>
<ContentTemplate>
@if ((context as AppointmentData).ElementType == "cell")
{
<div class="e-cell-content">
<div class="content-area">
<SfTextBox @ref="SubjectRef" Value="@((context as AppointmentData).Subject)"
Placeholder="Title"></SfTextBox>
</div>
<div class="content-area">
<SfDropDownList @ref="EventTypeRef" TValue="int" TItem="RoomsData" Index="0"
DataSource="@ResourceData" Placeholder="Choose Type">
<DropDownListFieldSettings Text="Name"
Value="Id"></DropDownListFieldSettings>
</SfDropDownList>
</div>
<div class="content-area">
<SfTextBox @ref="DescriptionRef" Value="@((context as
AppointmentData).Description)" Placeholder="Notes"></SfTextBox>
</div>
</div>
}
else
{
AppointmentData Data = context as AppointmentData;
<div class="event-content">
<div class="meeting-type-wrap">
<label>Subject</label>:
<span>@(Data.Subject)</span>
</div>
<div class="meeting-subject-wrap">
<label>Type</label>:
<span>@((Data.RoomId != 0) ? ResourceData.Where(item =>
item.Id.Equals(Data.RoomId)).FirstOrDefault().Name : "")</span>
</div>
</div>

```

```

<div class="notes-wrap">
<label>Notes</label>:
<span>@(Data.Description)</span>
</div>
</div>
}
</ContentTemplate>
<FooterTemplate>
@if ((context as AppointmentData).ElementType == "cell")
{
<div class="cell-footer">
<SfButton Content="More Details" OnClick="@ (e => OnMoreDetailsClick(e,
(context as AppointmentData), false))"></SfButton>
<SfButton Content="Add" IsPrimary="true" OnClick="@ (e => OnAdd(e, (context
as AppointmentData)))"></SfButton>
</div>
}
else
{
<div class="event-footer">
<SfButton IsPrimary="true" Content="More Details" OnClick="@ (e =>
OnMoreDetailsClick(e, (context as AppointmentData), true))"></SfButton>
</div>
}
</FooterTemplate>
</ScheduleQuickInfoTemplates>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
</ScheduleViews>
<ScheduleResources>
<ScheduleResource TValue="int" TItem="RoomsData" DataSource="@ResourceData"
Field="RoomId" Title="Room Type" Name="MeetingRoom" TextField="Name"
IdField="Id" ColorField="Color" AllowMultiple="false"></ScheduleResource>
</ScheduleResources>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
</SfSchedule>
@code{
SfSchedule<AppointmentData> SheduleRef;
SfDropDownList<int, RoomsData> EventTypeRef;
SfTextBox SubjectRef;
SfTextBox DescriptionRef;
private string GetEventDetails(AppointmentData data)
{
return data.StartTime.ToString("dddd dd, MMMM yyyy",
CultureInfo.InvariantCulture) + " (" + data.StartTime.ToString("hh:mm tt",
CultureInfo.InvariantCulture) + "-" + data.EndTime.ToString("hh:mm tt",
CultureInfo.InvariantCulture) + ")";
}
private string GetHeaderStyles(AppointmentData data)
{
if (data.ElementType == "cell")
{
//CellClick Header Styles
return "align-items: center ; color: #919191;";
}
}

```

```
else
{
//EventClick Header Styles
return "background:" + this.ResourceData.Where(item =>
item.Id.Equals(data.RoomId)).FirstOrDefault().Color + "; color: #FFFFFF;";
}
}
private async void OnMoreDetailsClick(MouseEventArgs args, AppointmentData
data, bool isEventData)
{
await SheduleRef.CloseQuickInfoPopupAsync();
if (isEventData == false)
{
AppointmentData eventData = new AppointmentData
{
Id = new Random().Next(1000),
Subject = SubjectRef.Value ?? "",
StartTime = data.StartTime,
EndTime = data.EndTime,
Location = data.Location,
Description = DescriptionRef.Value ?? "",
IsAllDay = data.IsAllDay,
RoomId = EventTypeRef.Value,
RecurrenceException = data.RecurrenceException,
RecurrenceID = data.RecurrenceID,
RecurrenceRule = data.RecurrenceRule
};
await SheduleRef.OpenEditorAsync(eventData, CurrentAction.Add);
}
else
{
AppointmentData eventData = new AppointmentData
{
Id = data.Id,
Subject = data.Subject,
Location = data.Location,
Description = data.Description,
StartTime = data.StartTime,
EndTime = data.EndTime,
IsAllDay = data.IsAllDay,
RoomId = data.RoomId,
RecurrenceException = data.RecurrenceException,
RecurrenceID = data.RecurrenceID,
RecurrenceRule = data.RecurrenceRule
};
await SheduleRef.OpenEditorAsync(eventData, CurrentAction.Save);
}
}
private async Task OnAdd(MouseEventArgs args, AppointmentData data)
{
await SheduleRef.CloseQuickInfoPopupAsync();
AppointmentData cloneData = new AppointmentData
{
Id = new Random().Next(1000),
Subject = SubjectRef.Value ?? "",
Description = DescriptionRef.Value ?? "",
StartTime = data.StartTime,
```

```

EndTime = data.EndTime,
RoomId = EventTypeRef.Value,
Location = data.Location,
IsAllDay = data.IsAllDay,
RecurrenceException = data.RecurrenceException,
RecurrenceID = data.RecurrenceID,
RecurrenceRule = data.RecurrenceRule
};
await SheduleRef.AddEventAsync(cloneData);
}
public List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData { Id = 1, Subject = "Board Meeting", Description =
    "Meeting to discuss business goal of 2020.", StartTime = new DateTime(2020,
    1, 5, 9, 30, 0), EndTime = new DateTime(2020, 1, 5, 11, 0, 0), RoomId = 1},
    new AppointmentData { Id = 2, Subject = "Training session on JSP",
    Description = "Knowledge sharing on JSP topics.", StartTime = new
    DateTime(2020, 1, 7, 9, 30, 0), EndTime = new DateTime(2020, 1, 7, 11, 0,
    0), RoomId = 2}
};
private List<RoomsData> ResourceData { get; set; } = new List<RoomsData> {
    new RoomsData { Name = "Jammy", Id = 1, Color = "#ea7a57", Capacity = 20,
    Type = "Conference" },
    new RoomsData { Name = "Tweety", Id = 2, Color = "#7fa900", Capacity = 7,
    Type = "Cabin" },
    new RoomsData { Name = "Nestle", Id = 3, Color = "#5978ee", Capacity = 5,
    Type = "Cabin" }
};
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public string Description { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public Nullable<bool> IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
    public string RecurrenceException { get; set; }
    public int RoomId { get; set; }
    public virtual string ElementType { get; set; }
}
public class RoomsData
{
    public string Name { get; set; }
    public int Id { get; set; }
    public int Capacity { get; set; }
    public string Color { get; set; }
    public string Type { get; set; }
}
<style>
/*HeaderStyles*/
.quick-info-header {
background-color: white;
padding: 8px 18px;

```

```
}
.quick-info-header-content {
justify-content: flex-end;
display: flex;
flex-direction: column;
padding: 5px 10px 5px;
}
.quick-info-title {
font-weight: 500;
font-size: 16px;
letter-spacing: 0.48px;
height: 22px;
}
.duration-text {
font-size: 11px;
letter-spacing: 0.33px;
height: 14px;
}
/*ContentStyles*/
.content-area {
padding: 10px;
width: 100%;
}
.event-content {
height: 90px;
display: flex;
flex-direction: column;
justify-content: center;
padding: 0 15px;
}
.meeting-type-wrap, .meeting-subject-wrap, .notes-wrap {
font-size: 11px;
color: #666;
letter-spacing: 0.33px;
height: 24px;
padding: 5px;
}
.event-content div label {
display: inline-block;
min-width: 45px;
color: #666;
}
.event-content div span {
font-size: 11px;
color: #151515;
letter-spacing: 0.33px;
line-height: 14px;
padding-left: 8px;
}
.quick-info .e-popup-content {
}
/*FooterStyles*/
.cell-footer.e-btn {
background-color: #ffffff;
border-color: #878787;
color: #878787;
}
```

```

.cell-footer {
padding-top: 10px;
}
.e-quick-popup-wrapper .e-cell-popup .e-popup-content {
padding: 0 14px;
}
</style>

```

The top screenshot displays a calendar view for the week of January 12-18, 2020. The interface includes navigation arrows, a date range selector, and view toggles for Today, Day, Week (selected), and Month. The calendar grid shows time slots from 9:00 AM to 12:00 PM. A modal dialog titled 'Add Appointment' is open, showing the date and time of the appointment (Monday 13, January 2020 (09:30 AM-10:00 AM)) and fields for Title, Jammy (selected), and Notes. Buttons for 'More Details' and 'Add' are at the bottom.

The bottom screenshot displays a calendar view for the week of January 05-11, 2020. It features the same navigation and view toggles. An appointment titled 'Board Meeting' is scheduled for Monday, January 6, from 9:30 AM to 11:00 AM. A modal dialog titled 'Appointment Details' is open, showing the subject (Board Meeting), type (Jammy), and notes (Meeting to discuss business goal of 2020). A 'More Details' button is at the bottom.

How to enable/disable quick popup on selection end

By default, the `QuickInfoOnSelectionEnd` property is set to `false` to prevent the quick popup at the end of multiple cell selection. The quick popup will be shown at the selection end by setting `true` value to this property.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px"
QuickInfoOnSelectionEnd="true" @bind-SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 2, 11);
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Paris", StartTime = new
DateTime(2020, 2, 11, 10, 0, 0) , EndTime = new DateTime(2020, 2, 11, 12, 0,
0) },
new AppointmentData { Id = 2, Subject = "Germany", StartTime = new
DateTime(2020, 2, 13, 10, 0, 0) , EndTime = new DateTime(2020, 2, 13, 12, 0,
0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}
```

How to enable/disable the multiple days selection

By default, the Scheduler allows the user to select multiple days. We can prevent this action by setting `false` to `AllowMultiRowSelection` property whereas its default value is `true`.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px"
AllowMultiRowSelection="false" @bind-SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
```

```

<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
    DateTime CurrentDate = new DateTime(2020, 2, 11);
    List<AppointmentData> DataSource = new List<AppointmentData>
    {
        new AppointmentData { Id = 1, Subject = "Paris", StartTime = new
        DateTime(2020, 2, 11, 10, 0, 0) , EndTime = new DateTime(2020, 2, 11, 12, 0,
        0) },
        new AppointmentData { Id = 2, Subject = "Germany", StartTime = new
        DateTime(2020, 2, 13, 10, 0, 0) , EndTime = new DateTime(2020, 2, 13, 12, 0,
        0) }
    };
    public class AppointmentData
    {
        public int Id { get; set; }
        public string Subject { get; set; }
        public string Location { get; set; }
        public DateTime StartTime { get; set; }
        public DateTime EndTime { get; set; }
        public string Description { get; set; }
        public bool IsAllDay { get; set; }
        public string RecurrenceRule { get; set; }
        public string RecurrenceException { get; set; }
        public Nullable<int> RecurrenceID { get; set; }
    }
}

```

How to close quick info popup manually

The quick info popup can be closed in scheduler by using the `CloseQuickInfoPopupAsync` public method.

ASPX-CS

```

@using Syncfusion.Blazor.Scheduler
@using Syncfusion.Blazor.Buttons
<SfButton Content="Close popup" OnClick="@e => OnBtnClick()"></SfButton>
<SfSchedule TValue="AppointmentData" @ref="ScheduleObj" Width="100%"
Height="750px" @bind-SelectedDate="@SelectedDate">
    <ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
    <ScheduleViews>
        <ScheduleView Option="View.Day"></ScheduleView>
        <ScheduleView Option="View.Week"></ScheduleView>
        <ScheduleView Option="View.WorkWeek"></ScheduleView>
        <ScheduleView Option="View.Month"></ScheduleView>
        <ScheduleView Option="View.Agenda"></ScheduleView>
    </ScheduleViews>
</SfSchedule>
@code{
    SfSchedule<AppointmentData> ScheduleObj;
    private DateTime SelectedDate = new DateTime(2020, 4, 1);
}

```

```
private async Task OnBtnClick()
{
    await ScheduleObj.CloseQuickInfoPopupAsync();
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
    DateTime(2020, 4, 1, 9, 30, 0) , EndTime = new DateTime(2020, 4, 1, 11, 0,
    0), EventType = "Confirmed" }
};
public class AppointmentData
{
    public string EventType { get; set; }
    public int Id { get; set; }
    public string Subject { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
    public string RecurrenceException { get; set; }
    public string Description { get; set; }
}
}
```

More events indicator and popup

When the number of appointments count that lies on a particular time range * default appointment height exceeds the default height of a cell in month view and all other timeline views, a + more text indicator will be displayed at the bottom of those cells. This indicator denotes that the cell contains few more appointments in it and clicking on that will display a popup displaying all the appointments present on that day.

To disable this option of showing popup with all hidden appointments, while clicking on the text indicator, you can do code customization within the `OnPopupOpen` event.

The same indicator is displayed on all-day row in calendar views such as day, week and work week views alone, when the number of appointment count present in a cell exceeds three. Clicking on the text indicator here will not open a popup, but will allow the expand/collapse option for viewing the remaining appointments present in the all-day row.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEvents TValue="AppointmentData"
OnPopupOpen="OnPopupOpen"></ScheduleEvents>
<ScheduleEventSettings DataSource="@DataSource">
</ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Month" MaxEventsPerRow="2"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
    DateTime CurrentDate = new DateTime(2020, 1, 9);
}
```

```

public void OnPopupOpen(PopupOpenEventArgs<AppointmentData> args)
{
    if (args.Type == PopupType.EventContainer)
    {
        args.Cancel = true;
    }
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData { Id = 1, Subject = "Meeting-1", StartTime = new
    DateTime(2020, 1, 9, 9, 30, 0) , EndTime = new DateTime(2020, 1, 9, 11, 0,
    0) },
    new AppointmentData { Id = 2, Subject = "Meeting-2", StartTime = new
    DateTime(2020, 1, 9, 9, 30, 0) , EndTime = new DateTime(2020, 1, 9, 11, 0,
    0) },
    new AppointmentData { Id = 3, Subject = "Meeting-3", StartTime = new
    DateTime(2020, 1, 9, 9, 30, 0) , EndTime = new DateTime(2020, 1, 9, 11, 0,
    0) }
};
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
    public string RecurrenceException { get; set; }
}
}

```

How to prevent the display of popup when clicking on the more text indicator

It is possible to prevent the display of popup window by passing the value `true` to `Cancel` option within the `MoreEventsClicked` event.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" @ref="ScheduleObj" Width="100%"
Height="750px" @bind-SelectedDate="@SelectedDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Month"></ScheduleView>
</ScheduleViews>
<ScheduleEvents TValue="AppointmentData"
MoreEventsClicked="OnMoreEventsClicked"></ScheduleEvents>
</SfSchedule>
@code{
SfSchedule<AppointmentData> ScheduleObj;
private DateTime SelectedDate = new DateTime(2020, 1, 31);
private void OnMoreEventsClicked(MoreEventsClickArgs args)
{
    args.Cancel = true;
}
List<AppointmentData> DataSource = new List<AppointmentData>

```

```

{
    new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
    DateTime(2020, 1, 15, 9, 30, 0) , EndTime = new DateTime(2020, 1, 15, 11, 0,
    0), EventType = "Confirmed" },
    new AppointmentData { Id = 1, Subject = "Scrum Meeting", StartTime = new
    DateTime(2020, 1, 15, 10, 30, 0) , EndTime = new DateTime(2020, 1, 15, 11,
    0, 0), EventType = "Confirmed" }
};
public class AppointmentData
{
    public string EventType { get; set; }
    public int Id { get; set; }
    public string Subject { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
    public string RecurrenceException { get; set; }
    public string Description { get; set; }
}
}

```

Timezone in Blazor Scheduler

The Scheduler renders events based on current system time zone of server in server side application and in WASM application the events are rendered based on UTC timezone by default.

- You can change the timezone of the scheduler by setting [Timezone](#) property. For more information, please refer to the [section](#).
- You can also set timezone for each appointment (events) through [StartTimezone](#) and [EndTimezone](#) properties which can be defined as separate fields within the event fields collection. For more information, please refer to the [section](#).
- You can also set the timezone for both the scheduler [Timezone](#) property and as well as the event's [StartTimezone](#) and [EndTimezone](#) properties. For more information, please refer to the [section](#).

NOTE

- The given value for the Timezone property for both the Scheduler and the appointments should be in the [IANA](#) format.
- The WASM application has supported the limited [time zones](#) in .Net5. But in .Net6, it supported all the [time zones](#).

Create appointments in different time zones

You can create appointments at different time zones using the [StartTimezone](#) and [EndTimezone](#) properties. An appointment's start time and end time are calculated based on the given time zone information.

In the following code example, the appointments time zone is Europe Time (UTC+03:00), and the application is running in UTC time zone (here the Blazor Server application is hosted in UTC time zone

and the Blazor WASM application's default time zone is UTC). In this scenario, the appointment will be displayed at 6 AM.

CSHARP

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Width="100%" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 3, 10);
List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData {
        Id = 1,
        Subject = "Meeting",
        StartTime = new DateTime(2020, 3, 9, 9, 0, 0) ,
        EndTime = new DateTime(2020, 3, 9, 11, 0, 0),
        StartTimezone = "Europe/Moscow",
        EndTimezone = "Europe/Moscow"
    }
};
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
    public string StartTimezone { get; set; }
    public string EndTimezone { get; set; }
}
```

NOTE

- If the recurring appointment is converted to another time zone, then the whole sequence will be recalculated according to the new time zone information.
- If an all-day appointment is created, it's start time and end time will be set to 12 A.M. and 12 A.M. by default, so time zone is not applicable for all-day appointments.
- Scheduler supports daylight saving time.

- The time zone support is applicable for custom appointments too, so map the corresponding property.
- Use [TimeZone](#) for custom appointments by mapping the [StartTimeZone](#) and [EndTimeZone](#) custom properties.

Display appointments based on Scheduler time zone

Set the specific time zone to schedule using the [TimeZone](#) property of scheduler. On this scenario, the appointments will be displayed in UTC time when the [StartTimeZone](#) and [EndTimeZone](#) properties are set to null. The appointments will be displayed in UTC time based on the given scheduler time zone.

In the following code example, the Scheduler time zone is Europe Time (UTC+03:00), and the application is running in UTC time zone (here the Blazor Server application is hosted in UTC time zone and the Blazor WASM application's default time zone is UTC). In this scenario, the appointment will be displayed at 12 PM.

CSHARP

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Width="100%" Height="550px" @bind-
SelectedDate="@CurrentDate" Timezone="Europe/Moscow">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 3, 10);
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 3, 9, 9, 0, 0) , EndTime = new DateTime(2020, 3, 9, 11, 0, 0)
}
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
public string StartTimeZone { get; set; }
public string EndTimezone { get; set; }
}
}
```

Display Appointments based on client's time zone

Display the appointments based on the client's local time zone in scheduler. It can be achieved by getting browser's timezone and set it's value to the scheduler time zone and appointment's time zone.

For example, consider a scenario that you are in North Carolina (America/New_York) and you want to set up a meeting at 10 A.M. on North Carolina time. You have colleagues in London and Chennai, and they also need to participate. The time for this meeting will be 3 P.M. (15:00) in London and 8.30 P.M. in Chennai. When each view your Scheduler, you need to see the appointment displayed relative to your local time zones.

CSHARP

```
@using Syncfusion.Blazor.Schedule
@inject IJSRuntime JSRuntime
<SfSchedule TValue="AppointmentData" Width="100%" Height="550px" @bind-
SelectedDate="@CurrentDate" Timezone="@TimezoneValue">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
public DateTime CurrentDate { get; set; } = new DateTime(2021, 1, 10);
string TimezoneValue;
protected override async Task OnAfterRenderAsync(bool firstRender)
{
if (firstRender)
{
TimezoneValue = await JSRuntime.InvokeAsync<string>("eval", "(function(){try
{ return '+' + Intl.DateTimeFormat().resolvedOptions().timeZone; } catch(e) {}
return 'UTC';})();");
StateHasChanged();
}
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2021, 1, 11, 10, 0, 0) , EndTime = new DateTime(2021, 1, 11, 11,
0, 0), StartTimezone = "America/New_York",
EndTimezone = "America/New_York" }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
}
```



```
public Nullable<int> RecurrenceID { get; set; }
public string StartTimezone { get; set; }
public string EndTimezone { get; set; }
}
```

Display appointments at same time everywhere regardless of client's time zone

Display the appointments at the same time everywhere without considering the time zone while setting the [Timezone](#) property of the scheduler, the [StartTimezone](#) and [EndTimezone](#) properties to null. The appointments will be displayed based on the given [StartTime](#) and [EndTime](#) of appointment everywhere without considering the time zone.

Updating StartTime and EndTime after drag and drop appointment based on time zone

After rescheduling an appointment using drag and drop, appointment's start and end time value will be updated based on scheduler time zone and appointment's time zone.

For an example, consider the local time zone is India Standard Time, if you drag an appointment from 9 AM and drop this on 1 PM and the scheduler's [Timezone](#) is not set and the appointment's [StartTimezone](#) and [EndTimezone](#) has set as AUS Central Standard Time (Australia/Darwin) then appointment's start time and end time value will be converted from Local time zone to appointment time zone and the appointment's start time will be saved at 5 PM.

If you set scheduler's [TimeZone](#) as AUS Central Standard Time (Australia/Darwin) and the appointment's [StartTimeZone](#) and [EndTimeZone](#) as Central Standard Time (America/Mexico_City) then the appointment's start time and end time value will be converted from scheduler's time zone to appointment time zone and the appointment's start time will be saved at 8.30 PM.

If you set scheduler's [TimeZone](#) as AUS Central Standard Time (Australia/Darwin) and appointment's time zone is not set then the appointment's start time and end time value will be converted from scheduler time zone to UTC time zone and the appointment's start time will be saved at 9 AM.

Views in Blazor Scheduler Component

The Scheduler includes wide variety of view modes with unique configuration options for each view. The available view modes are Day, Week, Work Week, Month, Agenda, Month Agenda, Timeline Day, Timeline Week, Timeline Work Week and Timeline Month, out of which the [Week](#) view is set as active.

To navigate between different views and dates, the navigation options are available at the Scheduler header bar. The active view option is usually highlighted by default. The date range of the active view will also be displayed at the left corner of the header bar, clicking on which will open a calendar popup for the ease of desired date selection.

By default, Scheduler displays the calendar views such as day, week, work week, month and agenda.

Setting specific view on scheduler

As the Scheduler displays [Week](#) view by default, therefore to change the active view, set [CurrentView](#) property with the desired view name. The applicable view names that the Scheduler accepts are as follows,

- Day
- Week
- WorkWeek
- Month

- Agenda
- MonthAgenda
- TimelineDay
- TimelineWeek
- TimelineWorkWeek
- TimelineMonth
- TimelineYear
- Year

It is possible to display only the desired views on the Scheduler using the **Views** property.

To get start quickly on customizing individual views of scheduler, you can check on this video:

{% youtube

"youtube:https://www.youtube.com/watch?v=bBhn7YHje8k"%}

In the following example, the Scheduler displays 2 views namely, Week and TimelineDay.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
CurrentView="@CurrentView">
  <ScheduleViews>
    <ScheduleView Option="View.Week"></ScheduleView>
    <ScheduleView Option="View.TimelineDay" MaxEventsPerRow="10"></ScheduleView>
  </ScheduleViews>
</SfSchedule>

@code{
View CurrentView = View.Week;
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
}
}
```

To configure Scheduler with different configurations on each view, refer the following code example.

Here, the Week view displays the dates in **dd-MM-yyyy** format whereas the Month view hides the weekend days and also displays it in readonly mode.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
CurrentView="@CurrentView">
  <ScheduleViews>
```

```

<ScheduleView Option="View.Week" DateFormat="dd-MMM-yyyy"></ScheduleView>
<ScheduleView Option="View.Month" MaxEventsPerRow="2" Readonly="true"
ShowWeekend="false"></ScheduleView>
</ScheduleViews>
</SfScheduler>
@code{
View CurrentView = View.Week;
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

View specific configuration

There are scenarios where each view may need to have different configurations. For such cases, you can define the applicable scheduler properties within the **Views** Property for each view option as depicted in the following examples. The fields available to be used within each view options are as follows.

Property	Type	Description	Applicable views
Option	View	It accepts the Scheduler view name, based on which its related properties is defined. The view names can be Day, Week and so on.	All views.
IsSelected	bool	It acts similar to the CurrentView property and defines the active view of the Scheduler.	All views.
DateFormat	string	By default, Scheduler follows the date format as per the default culture assigned to it. When it is defined under specific view, only those assigned views follows this date format.	All views.
Readonly	bool	When set to true, prevents the CRUD actions on the respective view under where it is defined.	All views.
ResourceHeaderTemplate	string	The template option which is used to customize the resource header cells on the Scheduler. It gets applied only on the views, wherever it is defined.	All views.
DateHeaderTemplate	string	The template option which is used to customize the date header cells and is applied only on the views, wherever it is defined.	All views.
EventTemplate	string	The template option to customize the events background. It will get applied to the events of the view to which it is currently being defined.	All views.
ShowWeekend	bool	When set to false, it hides the weekend days of a week from the views on which it is defined.	All views.

| **Group** | **GroupModel** | Allows to set different resource grouping options on all available Scheduler view modes. | All views. |

| **CellTemplate** | string | The template option to customize the work cells of the Scheduler and is applied only on the views, on which it is defined. | Applicable on all views except Agenda view. |

| **WorkDays** | int[] | It is used to set the working days on the Scheduler views. | Applicable on all views except Agenda view. |

| **DisplayName** | string | When a particular view is customized to display with different intervals, this property allows the user to set different display name for each of the views. | Applicable on all views except Agenda and Month Agenda. |

| **Interval** | int | It allows to customize the default Scheduler views with different set of days, weeks, work weeks or months on the applicable view type. | Applicable on all views except Agenda and Month Agenda. |

| **StartHour** | string | It is used to specify the start hour, from which the Scheduler should be displayed. It accepts the time string in a short skeleton format and also, hides the time beyond the specified start time. | Applicable on Day, Week, Work Week, Timeline Day, Timeline Week and Timeline Work Week views. |

| **EndHour** | string | It is used to specify the end hour, at which the Scheduler ends. It accepts the time string in a short skeleton format. | Applicable on Day, Week, Work Week, Timeline Day, Timeline Week, and Timeline Work Week views. |

| **TimeScale** | **TimeScaleModel** | Allows to set different timescale configuration on each applicable view modes. | Applicable on Day, Week, Work Week, Timeline Day, Timeline Week, and Timeline Work Week views. |

| **ShowWeekNumber** | bool | When set to **true**, shows the week number on the respective weeks. | Applicable on Day, Week, Work Week, and Month views. |

| **AllowVirtualScrolling** | bool | It is used to enable or disable the virtual scrolling functionality. | Applicable on Agenda and Timeline views. |

| **HeaderRows** | **HeaderRowsModel** | Allows defining the custom header rows on timeline views of the Scheduler to display the year, month, week, date and hour label as an individual row. | Applicable only on all timeline views. |

Day view

Usually a day view displays a single day with all its related appointments. It is possible to customize the day view to display more number of days by extending the **Views** property with **Interval** option. You can also define any of the above defined properties within the **ScheduleView** tag helper as depicted in the following code example.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px">
  <ScheduleViews>
    <ScheduleView Option="View.Day" Interval="3"></ScheduleView>
  </ScheduleViews>
</SfSchedule>
@code{
```

```
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
}
}
```

All the above defined properties can be accessed within Day view except **AllowVirtualScrolling** and **HeaderRows**.

Week view

The Week view displays a count of 7 days (from Sunday to Saturday) with all its related appointments. The first day of the week can be changed using the **FirstDayOfWeek** which accepts the integer (Sunday=0, Monday=1, Tuesday=2 and so on) value. You can navigate to a particular date in day view from the week view by clicking on the appropriate dates on the date header bar.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px">
    <ScheduleViews>
        <ScheduleView Option="View.Day"></ScheduleView>
        <ScheduleView Option="View.Week" Interval="3" DisplayName="3
        Weeks"></ScheduleView>
    </ScheduleViews>
</SfSchedule>
@code{
    public class AppointmentData
    {
        public int Id { get; set; }
        public string Subject { get; set; }
        public string Location { get; set; }
        public DateTime StartTime { get; set; }
        public DateTime EndTime { get; set; }
        public string Description { get; set; }
        public bool IsAllDay { get; set; }
        public string RecurrenceRule { get; set; }
        public string RecurrenceException { get; set; }
        public Nullable<int> RecurrenceID { get; set; }
    }
}
```

All the above defined properties in the table can be accessed within Week and Work week views except **AllowVirtualScrolling** and **HeaderRows**.

Work Week view

The Work week view displays only the working days of a week (count of 5 days) and its associated appointments. It is possible to customize the working days on the work week view by using the **WorkDays** property which accepts an array of integer values (such as Sunday=0, Monday=1, Tuesday=2 and so on). By default, it displays from Monday to Friday (5 days). You can also navigate to a particular date in the day view from the work week view by clicking on the appropriate dates in the date header bar.

The following code example depicts how to change the start and end hours only on the **Work Week** view of the Scheduler.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px">
  <ScheduleViews>
    <ScheduleView Option="View.WorkWeek" StartHour="08:00"
    EndHour="13:00"></ScheduleView>
  </ScheduleViews>
</SfSchedule>

@code{
  public class AppointmentData
  {
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
  }
}
```

The Week, Work week and Day views can display the all-day row appointments in a separate all-day row with an expand or collapse option to view it.

Month view

A Month view displays the entire days of a particular month and all its related appointments. You can navigate to a particular date in the day view by clicking on the appropriate date text on the month cells.

By default, when you try to create an appointment through Month view, it is considered to be created for an entire day. You can explicitly change this behavior by unchecking the **All-day** option from editor window, so that it defaults to the start time duration as 9.00 AM and end time as 9.30 AM.

By default, in month view, you can view single appointment on each day cell. If you have more than one appointment in a day, the **+ more** text indicator will be available on that cell, clicking on which will allow you to view the hidden appointments of a day. You can decide how many appointments can render on a day based on your Scheduler and Month cell height using **MaxEventsPerRow** property within **ScheduleView** whereas its default value is 1. The following code example depicts how to change the working days only on the **Month** view of the Scheduler.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px">
<ScheduleViews>
<ScheduleView Option="View.Month" MaxEventsPerRow="2"
WorkDays="@WorkingDays"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
public int[] WorkingDays { get; set; } = { 1, 3, 5 };
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

Agenda view

The Agenda view lists out the appointments in a grid-like view for the next 7 days by default from the current date. The count of the days can be changed using the API `AgendaDaysCount`. It allows virtual scrolling of dates by enabling the `AllowVirtualScrolling` property. Also, you can enable or disable the display of days on Scheduler that has no appointments by setting true or false to the `HideEmptyAgendaDays` property.

The following code example depicts how to display events of four days in Agenda view.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" AgendaDaysCount="4">
<ScheduleViews>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

```
}
}
```

Month Agenda view

A Month-Agenda view shows a month calendar, where clicking on a particular day will display the appointments present on that date below the calendar. The day with appointments are differentiated with a circular dot below the date of the calendar.

The following code example shows how to hide the weekend days on `MonthAgenda` view as well as the working days list is modified on Month Agenda view alone.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px">
  <ScheduleViews>
    <ScheduleView Option="View.MonthAgenda" ShowWeekend="false"
      WorkDays="@WorkingDays"></ScheduleView>
  </ScheduleViews>
</SfSchedule>

@code{
public int[] WorkingDays { get; set; } = { 0, 3, 6 };
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
}
}
```

Timeline views – Day, Week, Work Week

Similar to the vertical day, week and work week views, the respective view shows all its appointments where the time slots are displayed horizontally. By default, the cell height adjusts as per the height set to Scheduler and you can view single appointment on each cell. If you have more than one appointment, the `+ more` text indicator will be available on the bottom of that cell, clicking on which allows you to view the hidden appointments of a day. You can decide how many appointments can render on a cell based on your Scheduler and work cell height using `MaxEventsPerRow` property within `ScheduleView` whereas its default value is `1`.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px">
  <ScheduleViews>
    <ScheduleView Option="View.TimelineDay" StartHour="08:00" EndHour="12:00"
      ShowWeekend="false" MaxEventsPerRow="10"></ScheduleView>
  </ScheduleViews>
</SfSchedule>
```



```

<ScheduleView Option="View.TimelineWeek" IsSelected="true"
MaxEventsPerRow="10"></ScheduleView>
<ScheduleView Option="View.TimelineWorkWeek" StartHour="13:00"
EndHour="20:00" WorkDays="@WorkingDays" MaxEventsPerRow="10"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
public int[] WorkingDays { get; set; } = { 1, 3, 6 };
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

Clicking on the dates in the date header bar of Timeline day, Timeline week and Timeline work week will allow to navigate to the Agenda view.

Timeline Month view

A Timeline Month view displays the current month days along with its appointments.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px">
<ScheduleViews>
<ScheduleView Option="View.TimelineDay" StartHour="08:00" EndHour="12:00"
MaxEventsPerRow="10"></ScheduleView>
<ScheduleView Option="View.TimelineMonth" IsSelected="true"
ShowWeekend="false" MaxEventsPerRow="10"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

Clicking on the dates in the date header bar of Timeline month allows to navigate to the Timeline day view.

Timeline Year view

A Timeline Year view displays the complete year along with its appointments.

By default, the timeline year view orientation is set to Horizontal view. In this following code example, the timeline year view is set with vertical orientation.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Width="100%" Height="550px" @bind-
SelectedDate="@CurrentDate">
  <ScheduleViews>
    <ScheduleView Option="View.TimelineYear" Orientation="Orientation.Vertical"
    DisplayName="Horizontal Year"></ScheduleView>
  </ScheduleViews>
  <ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 3, 10);
List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
    DateTime(2020, 3, 4, 0, 0, 0) , EndTime = new DateTime(2020, 3, 5, 0, 0, 0)
    },
    new AppointmentData { Id = 2, Subject = "Conference", StartTime = new
    DateTime(2020, 5, 1, 9, 30, 0) , EndTime = new DateTime(2020, 5, 1, 12, 0,
    0) },
    new AppointmentData { Id = 3, Subject = "Seminar", StartTime = new
    DateTime(2020, 1, 2, 9, 30, 0) , EndTime = new DateTime(2020, 1, 2, 12, 0,
    0) }
};
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
}
}
```

Setting the first month of timeline year

By default, months in timeline year view displayed from January to December. User can customize this default behavior with the help of scheduler [FirstMonthOfYear](#) property. This property allows user to set the first month of the timeline year on Scheduler. User can set first month of timeline year by passing

integer value to the `FirstMonthOfYear` property, whereby 1 is always denoted as January, 2 as February and so on. This property applicable only in timeline year views.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="650px" @bind-
SelectedDate="@CurrentDate" FirstMonthOfYear="4">
  <ScheduleViews>
    <ScheduleView Option="View.TimelineYear"
Orientation="Orientation.Horizontal" DisplayName="Horizontal Timeline Year"
IsSelected="true"></ScheduleView>
    <ScheduleView Option="View.TimelineYear" Orientation="Orientation.Vertical"
DisplayName="Vertical Timeline Year"></ScheduleView>
  <ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
</ScheduleViews>
</SfSchedule>

@code{
DateTime CurrentDate = new DateTime(2021, 4, 3);
List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData { Id = 1, Subject = "Paris", StartTime = new
DateTime(2021, 5, 13, 10, 0, 0) , EndTime = new DateTime(2021, 5, 13, 12, 0,
0) },
    new AppointmentData { Id = 2, Subject = "Germany", StartTime = new
DateTime(2021, 5, 15, 10, 0, 0) , EndTime = new DateTime(2021, 5, 15, 12, 0,
0) }
};
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
}
}
```

Year view

The Year view shows a year calendar, where clicking on a particular day will display the appointments present on that date below the calendar. The day with appointments are differentiated with a circular dot below the date of the calendar.

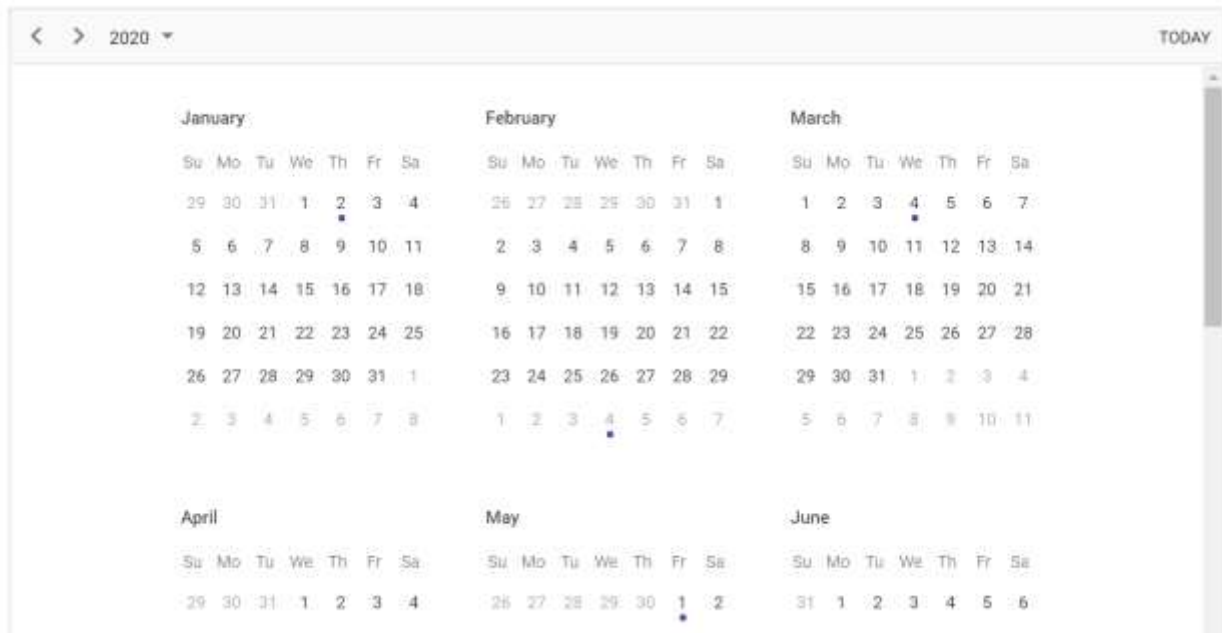
ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Width="100%" Height="550px" @bind-
SelectedDate="@CurrentDate">
  <ScheduleViews>
    <ScheduleView Option="View.Year"></ScheduleView>
  </ScheduleViews>
</SfSchedule>
```

```

<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 3, 10);
List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
    DateTime(2020, 3, 4, 0, 0, 0) , EndTime = new DateTime(2020, 3, 5, 0, 0, 0)
    },
    new AppointmentData { Id = 2, Subject = "Conference", StartTime = new
    DateTime(2020, 5, 1, 9, 30, 0) , EndTime = new DateTime(2020, 5, 1, 12, 0,
    0) },
    new AppointmentData { Id = 3, Subject = "Seminar", StartTime = new
    DateTime(2020, 1, 2, 9, 30, 0) , EndTime = new DateTime(2020, 1, 2, 12, 0,
    0) }
    };
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
}
}

```



Extending view intervals

It is possible to customize the display of default number of days on different Scheduler view modes. For example, a day view can be extended to display 4 days by setting the `Interval` option as 4 for the `Day`

option within the `ScheduleView` as depicted in the following code example. In the same way, you can also display 3 weeks by setting interval 3 for the `Week` option.

You can provide the alternative display name for such customized views on the Scheduler header bar, by setting the appropriate `DisplayName` property.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px">
  <ScheduleViews>
    <ScheduleView Option="View.Day" Interval="4" DisplayName="4
    Days"></ScheduleView>
    <ScheduleView Option="View.Week" IsSelected="true" Interval="3"
    DisplayName="3 Weeks"></ScheduleView>
  </ScheduleViews>
</SfSchedule>
@code{
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
}
}
```

The view intervals can be extended on all the Scheduler view modes except Agenda and Month-Agenda views.

Resources and Grouping in Blazor Scheduler Component

[Resources and grouping](#) support allows the Scheduler to be shared by multiple resources. Also, the appointments of each resource are displayed under relevant resources. Each resource in the Scheduler is arranged in a column or row wise order, with individual spacing to display all its respective appointments on a single page. It also supports the multiple levels of grouping of resources, thus enabling the categorization of resources in a hierarchical structure and shows it either in expandable groups (Timeline views) or else vertical hierarchy one after the other (Calendar views).

It is also possible to assign one or more resources to the same appointment, by allowing multiple selection of resource options available in the event editor window.

The Scheduler groups the resources based on different criteria. It includes grouping appointments based on resources, grouping resources based on dates, and timeline scheduling. Also, the data for resources bind with Scheduler either as a local JSON collection or URL, retrieving data from remote data services.

Resource fields

The default options available within the `Resources` collection are as follows,

Field name	Type	Description
------------	------	-------------

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

| **Field** | string | A value that binds to the resource field of event object. |

| **Title** | string | It holds the title of the resource field to be displayed on the event editor window. |

| **Name** | string | A unique resource name used for differentiating various resource objects while grouping. |

| **AllowMultiple** | bool | When set to **true**, allows multiple selection of resource names, thus creating multiple instances of same appointment for the selected resources. |

| **DataSource** | Object | Assigns the resource **DataSource**, where data can be passed either as an array of JavaScript objects, or else can create an instance of [DataManager](#) in case of processing remote data and can be assigned to the **DataSource** property. With the remote data assigned to **DataSource**, check the available [Adaptors](#) to customize the data processing. |

| **Query** | query | Defines the external [Query](#) that will be executed along with the data processing. |

| **IdField** | string/int/Guid | Binds the resource ID field name from the resources **DataSource**. |

| **TextField** | string | Binds the text field name from the resources **DataSource**. It usually holds the resource names. |

| **GroupIDField** | string | Binds the group ID field name from the resource **DataSource**. It usually holds the value of resource IDs of parent level resources. |

| **ColorField** | string | Binds the color field name from the resource **DataSource**. The color value mapped in this field will be applied to the events of resources. |

| **StartHourField** | string | Binds the start hour field name from the resource **DataSource**. It allows to provide different work start hour for the resources. |

| **EndHourField** | string | Binds the end hour field name from the resource **DataSource**. It allows to provide different work end hour for the resources. |

| **WorkDaysField** | string | Binds the work days field name from the resources **DataSource**. It allows to provide different working days collection for the resources. |

| **CssClassField** | string | Binds the custom CSS class field name from the resources **DataSource**. It maps the CSS class written for the specific resources and applies it to the events of those resources. |

Resource data binding

The data for resources can bind with Scheduler either as list of object collection or a service URL, retrieving resource data from remote data services. The **TItem** in the **ScheduleResource** holds the generic class model of resource **DataSource** and **TValue** holds the generic type of resource id which need to be in array when **AllowMultiple** is set to true.

The following code example depicts how to bind the list of object collection to the **DataSource** of Resource collection.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="650px" @bind-
SelectedDate="@CurrentDate">
```

```

<ScheduleResources>
<ScheduleResource TItem="ResourceData" TValue="int" DataSource="@OwnerData"
Field="OwnerId" Title="Owner" Name="Owner" TextField="Text" IdField="Id"
ColorField="Color"></ScheduleResource>
</ScheduleResources>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 4, 4);
public List<ResourceData> OwnerData { get; set; } = new List<ResourceData> {
new ResourceData{ Text = "Nancy", Id= 1, Color = "#df5286" },
new ResourceData{ Text = "Steven", Id= 2, Color = "#7fa900" },
new ResourceData{ Text = "Robert", Id= 3, Color = "#ea7a57" }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
public int OwnerId { get; set; }
}
public class ResourceData
{
public int Id { get; set; }
public string Text { get; set; }
public string Color { get; set; }
}
}

```

Binding ExpandoObject

Scheduler is a generic component which is strongly bound to a model type. There are cases when the model type is unknown during compile time. In such cases data can be bound to the scheduler as list of **ExpandoObject**.

ExpandoObject can be bound to the **DataSource** option of the scheduler within the **ScheduleResource** tag. Scheduler can also perform all kind of supported data operations and editing in **ExpandoObject**.

C# SHARP

```

@using System.Dynamic
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">

```

```

<ScheduleGroup Resources="@Resources"></ScheduleGroup>
<ScheduleResources>
<ScheduleResource TItem="ExpandoObject" TValue="int"
DataSource="@ResourceCollection" Field="OwnerId" Title="Owner" Name="Owners"
TextField="OwnerText" IdField="Id" ColorField="OwnerColor"
AllowMultiple="false"></ScheduleResource>
</ScheduleResources>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfScheduler>
@code {
DateTime CurrentDate = new DateTime(2020, 3, 9);
public string[] Resources { get; set; } = { "Owners" };
public List<ExpandoObject> ResourceCollection = new List<ExpandoObject>() {
};
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 3, 9, 9, 0, 0) , EndTime = new DateTime(2020, 3, 9, 11, 0,
0), OwnerId = 1 }
};
protected override void OnInitialized()
{
var colors = new string[] { "#ff8787", "#9775fa", "#748ffc" };
for (int a = 1; a <= 3; a++)
{
dynamic d = new ExpandoObject();
d.Id = a;
d.OwnerText = "Resource" + a;
d.OwnerColor = colors[a - 1];
ResourceCollection.Add(d);
}
}
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
public string StartTimezone { get; set; }
public string EndTimezone { get; set; }
public int OwnerId { get; set; }
}
}

```


Binding DynamicObject

Scheduler is a generic component which is strongly bound to a model type. There are cases when the model type is unknown during compile time. In such cases data can be bound to the scheduler as list of **DynamicObject**.

DynamicObject can be bound to the **DataSource** option of the scheduler within the **ScheduleResource** tag. Scheduler can also perform all kinds of supported data operations and editing in **DynamicObject**.

The [GetDynamicMemberNames](#) method of **DynamicObject** class must be overridden and return the property names to perform data operation and editing while using **DynamicObject**.

CSHARP

```
@using System.Dynamic
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleGroup Resources="@Resources"></ScheduleGroup>
<ScheduleResources>
<ScheduleResource TItem="DynamicDictionary" TValue="int"
DataSource="@ResourceCollection" Field="OwnerId" Title="Owner" Name="Owners"
TextField="OwnerText" IdField="Id" ColorField="OwnerColor"
AllowMultiple="false"></ScheduleResource>
</ScheduleResources>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code {
DateTime CurrentDate = new DateTime(2020, 3, 10);
public string[] Resources { get; set; } = { "Owners" };
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 3, 9, 9, 0, 0) , EndTime = new DateTime(2020, 3, 9, 11, 0,
0), OwnerId = 1 }
};
public List<DynamicDictionary> ResourceCollection = new
List<DynamicDictionary>() { };
protected override void OnInitialized()
{
var colors = new string[] { "#ff8787", "#9775fa", "#748ffc" };
for (int a = 1; a <= 3; a++)
{
dynamic d = new DynamicDictionary();
d.Id = a;
d.OwnerText = "Resource" + a;
d.OwnerColor = colors[a - 1];
ResourceCollection.Add(d);
}
```

```

}
}
public class DynamicDictionary : System.Dynamic.DynamicObject
{
    Dictionary<string, object> dictionary = new Dictionary<string, object>();
    public override bool TryGetMember(GetMemberBinder binder, out object result)
    {
        string name = binder.Name;
        return dictionary.TryGetValue(name, out result);
    }
    public override bool TrySetMember(SetMemberBinder binder, object value)
    {
        dictionary[binder.Name] = value;
        return true;
    }
    public override System.Collections.Generic.IEnumerable<string>
        GetDynamicMemberNames()
    {
        return this.dictionary?.Keys;
    }
}
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
    public int OwnerId { get; set; }
}
}

```

Binding ObservableCollection

This [ObservableCollection](#) (dynamic data collection) provides notifications when items are added, removed and moved. The implement [INotifyCollectionChanged](#) notifies when dynamic changes of add,remove, move and clear the collection. The implement [INotifyPropertyChanged](#) notifies when property value has changed in client side.

Here, ResourceData class implements the interface of **INotifyPropertyChanged** and it raises the event when RoomText and OwnerText property value was changed.

CSHARP

```

@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Buttons
@using System.Collections.ObjectModel;
@using System.ComponentModel;
<SfButton @onclick="AddRecord">Add Data</SfButton>
<SfButton @onclick="DeleteRecord">Delete Data</SfButton>
<SfButton @onclick="UpdateRecord">Update Data</SfButton>

```

```

<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleGroup Resources="@Resources"></ScheduleGroup>
<ScheduleResources>
<ScheduleResource TItem="ResourceData" TValue="int"
DataSource="@ObservableRoomData" Field="RoomId" Title="Room" Name="Rooms"
TextField="RoomText" IdField="Id" ColorField="RoomColor"
AllowMultiple="false"></ScheduleResource>
<ScheduleResource TItem="ResourceData" TValue="int[]"
DataSource="@ObservableOwnersData" Field="OwnerId" Title="Owner"
Name="Owners" TextField="OwnerText" IdField="Id" GroupIDField="OwnerGroupId"
ColorField="OwnerColor" AllowMultiple="true"></ScheduleResource>
</ScheduleResources>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 31);
int roomId = 2;
int ownerId = 3;
public string[] Resources { get; set; } = { "Rooms", "Owners" };
public ObservableCollection<ResourceData> ObservableRoomData { get; set; }
public ObservableCollection<ResourceData> ObservableOwnersData { get; set; }
protected override void OnInitialized()
{
ObservableRoomData = new ObservableCollection<ResourceData>(GetRoomData());
ObservableOwnersData = new
ObservableCollection<ResourceData>(GetOwnersData());
}
private static List<ResourceData> GetRoomData()
{
List<ResourceData> roomData = new List<ResourceData>
{
new ResourceData{ RoomText = "ROOM 1", Id = 1, RoomColor = "#cb6bb2" },
new ResourceData{ RoomText = "ROOM 2", Id = 2, RoomColor = "#56ca85" }
};
return roomData;
}
private static List<ResourceData> GetOwnersData()
{
List<ResourceData> ownersData = new List<ResourceData>
{
new ResourceData{ OwnerText = "Nancy", Id = 1, OwnerGroupId = 1, OwnerColor
= "#ffaa00" },
new ResourceData{ OwnerText = "Steven", Id = 2, OwnerGroupId = 2, OwnerColor
= "#f8a398" },
new ResourceData{ OwnerText = "Michael", Id = 3, OwnerGroupId = 1,
OwnerColor = "#7499e1" }
};
return ownersData;
}
}

```

```

List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
    DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
    0), OwnerId = 1, RoomId = 1 }
};
public void AddRecord()
{
    ownerId++;
    ObservableOwnersData.Add(new ResourceData() { OwnerText = "Nancy", Id =
    ownerId, OwnerGroupId = 1, OwnerColor = "#ffaa00" });
}
public void DeleteRecord()
{
    if (ObservableOwnersData.Count() != 0)
    {
        ObservableOwnersData.Remove(ObservableOwnersData.First());
    }
}
public void UpdateRecord()
{
    if (ObservableOwnersData.Count() != 0)
    {
        var data = ObservableOwnersData.First();
        data.OwnerText = "Updated Name";
    }
}
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
    public int OwnerId { get; set; }
    public int RoomId { get; set; }
}
public class ResourceData : INotifyPropertyChanged
{
    public int Id { get; set; }
    private string roomText { get; set; }
    public string RoomText
    {
        get { return roomText; }
        set
        {
            this.roomText = value;
            NotifyPropertyChanged("RoomText");
        }
    }
    public string RoomColor { get; set; }
    private string ownerText { get; set; }
}

```

```

public string OwnerText
{
    get { return ownerText; }
    set
    {
        this.ownerText = value;
        NotifyPropertyChanged("OwnerText");
    }
}

public string OwnerColor { get; set; }
public int OwnerGroupId { get; set; }
public event PropertyChangedEventHandler PropertyChanged;
private void NotifyPropertyChanged(string propertyName)
{
    var handler = PropertyChanged;
    if (handler != null)
    {
        handler(this, new PropertyChangedEventArgs(propertyName));
    }
}
}

```

Scheduler with multiple resources

It is possible to display the Scheduler in default mode without visually showcasing all the resources in it, but allowing to assign the required resources to the appointments through the event editor resource options.

The appointments belonging to the different resources will be displayed altogether on the default Scheduler, which will be differentiated based on the resource color assigned in the **Resources** (depicting to which resource that particular appointment belongs) collection.

To get start quickly about multiple resource on scheduler, you can check on this video:

{% youtube

"youtube:https://www.youtube.com/watch?v=ZJU73bqeoC0"%}

Example: To display default Scheduler with multiple resource options in the event editor, ignore the group option and simply define the `ScheduleResource` with all its internal options.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
    <ScheduleResources>
        <ScheduleResource TItem="ResourceData" TValue="int[]"
DataSource="@OwnersData" Field="OwnerId" Title="Owner" Name="Owners"
TextField="OwnerText" IdField="Id" ColorField="OwnerColor"
AllowMultiple="true"></ScheduleResource>
    </ScheduleResources>
    <ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
    <ScheduleViews>
        <ScheduleView Option="View.Day"></ScheduleView>
        <ScheduleView Option="View.Week"></ScheduleView>
    </ScheduleViews>
</SfSchedule>

```

```

<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
    DateTime CurrentDate = new DateTime(2020, 1, 31);
    List<AppointmentData> DataSource = new List<AppointmentData>
    {
        new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
        DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
        0), OwnerId = 1 }
    };
    public List<ResourceData> OwnersData { get; set; } = new List<ResourceData>
    {
        new ResourceData{ OwnerText = "Nancy", Id = 1, OwnerColor = "#ffaa00" },
        new ResourceData{ OwnerText = "Steven", Id = 2, OwnerColor = "#f8a398" },
        new ResourceData{ OwnerText = "Michael", Id = 3, OwnerColor = "#7499e1" }
    };
    public class AppointmentData
    {
        public int Id { get; set; }
        public string Subject { get; set; }
        public string Location { get; set; }
        public DateTime StartTime { get; set; }
        public DateTime EndTime { get; set; }
        public string Description { get; set; }
        public bool IsAllDay { get; set; }
        public string RecurrenceRule { get; set; }
        public string RecurrenceException { get; set; }
        public Nullable<int> RecurrenceID { get; set; }
        public int OwnerId { get; set; }
    }
    public class ResourceData
    {
        public int Id { get; set; }
        public string OwnerText { get; set; }
        public string OwnerColor { get; set; }
    }
}

```

Setting **AllowMultiple** to **true** in the above code example allows to select multiple resources from the event editor and also creates multiple copies of the same appointment in the Scheduler for each resources while rendering.

Resource grouping

Resource grouping support allows the Scheduler to group the resources in a hierarchical structure both as an expandable groups (Timeline views) and as vertical hierarchy displaying resources one after the other (Resources view).

To get start quickly about grouping multiple resource on scheduler, you can check on this video:

{% youtube

"youtube:https://www.youtube.com/watch?v=70qD6wycxAk"%}

Scheduler supports both single and multiple levels of resource grouping that can be customized both in timeline and vertical Scheduler views.

Vertical resource view

The following code example displays how the multiple resources are grouped and its events are portrayed in the default calendar views.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleGroup Resources="@Resources"></ScheduleGroup>
<ScheduleResources>
<ScheduleResource TItem="ResourceData" TValue="int" DataSource="@RoomData"
Field="RoomId" Title="Room" Name="Rooms" TextField="RoomText" IdField="Id"
ColorField="RoomColor" AllowMultiple="false"></ScheduleResource>
<ScheduleResource TItem="ResourceData" TValue="int[]"
DataSource="@OwnersData" Field="OwnerId" Title="Owner" Name="Owners"
TextField="OwnerText" IdField="Id" GroupIDField="OwnerGroupId"
ColorField="OwnerColor" AllowMultiple="true"></ScheduleResource>
</ScheduleResources>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 31);
public string[] Resources { get; set; } = { "Rooms", "Owners" };
public List<ResourceData> RoomData { get; set; } = new List<ResourceData>
{
new ResourceData{ RoomText = "ROOM 1", Id = 1, RoomColor = "#cb6bb2" },
new ResourceData{ RoomText = "ROOM 2", Id = 2, RoomColor = "#56ca85" }
};
public List<ResourceData> OwnersData { get; set; } = new List<ResourceData>
{
new ResourceData{ OwnerText = "Nancy", Id = 1, OwnerGroupId = 1, OwnerColor = "#ffaa00" },
new ResourceData{ OwnerText = "Steven", Id = 2, OwnerGroupId = 2, OwnerColor = "#f8a398" },
new ResourceData{ OwnerText = "Michael", Id = 3, OwnerGroupId = 1, OwnerColor = "#7499e1" }
};
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
0), OwnerId = 1, RoomId = 1 }
};
public class AppointmentData
{
public int Id { get; set; }
```

```

public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
public int OwnerId { get; set; }
public int RoomId { get; set; }
}
public class ResourceData
{
public int Id { get; set; }
public string RoomText { get; set; }
public string RoomColor { get; set; }
public string OwnerText { get; set; }
public string OwnerColor { get; set; }
public int OwnerGroupId { get; set; }
}
}

```

Timeline resource view

The following code example depicts how to group the multiple resources on Timeline Scheduler views and its relevant events are displayed accordingly under those resources.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleGroup Resources="@Resources"></ScheduleGroup>
<ScheduleResources>
<ScheduleResource TItem="ResourceData" TValue="int" DataSource="@RoomData"
Field="RoomId" Title="Room" Name="Rooms" TextField="RoomText" IdField="Id"
ColorField="RoomColor" AllowMultiple="false"></ScheduleResource>
<ScheduleResource TItem="ResourceData" TValue="int[]"
DataSource="@OwnersData" Field="OwnerId" Title="Owner" Name="Owners"
TextField="OwnerText" IdField="Id" GroupIDField="OwnerGroupId"
ColorField="OwnerColor" AllowMultiple="true"></ScheduleResource>
</ScheduleResources>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.TimelineWeek" MaxEventsPerRow="2"></ScheduleView>
<ScheduleView Option="View.TimelineMonth"
MaxEventsPerRow="2"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 31);
public string[] Resources { get; set; } = { "Rooms", "Owners" };
public List<ResourceData> RoomData { get; set; } = new List<ResourceData>
{
new ResourceData{ RoomText = "ROOM 1", Id = 1, RoomColor = "#cb6bb2" },

```



```

new ResourceData{ RoomText = "ROOM 2", Id = 2, RoomColor = "#56ca85" }
};
public List<ResourceData> OwnersData { get; set; } = new List<ResourceData>
{
    new ResourceData{ OwnerText = "Nancy", Id = 1, OwnerGroupId = 1, OwnerColor
    = "#ffaa00" },
    new ResourceData{ OwnerText = "Steven", Id = 2, OwnerGroupId = 2, OwnerColor
    = "#f8a398" },
    new ResourceData{ OwnerText = "Michael", Id = 3, OwnerGroupId = 1,
    OwnerColor = "#7499e1" }
};
List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
    DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
    0), OwnerId = 1, RoomId = 1 }
};
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
    public int OwnerId { get; set; }
    public int RoomId { get; set; }
}
public class ResourceData
{
    public int Id { get; set; }
    public string RoomText { get; set; }
    public string RoomColor { get; set; }
    public string OwnerText { get; set; }
    public string OwnerColor { get; set; }
    public int OwnerGroupId { get; set; }
}
}

```

Grouping single-level resources

This kind of grouping allows the Scheduler to display all the resources at a single level simultaneously. The appointments mapped under resources will be displayed with the colors as per the **ColorField** defined on the resources collection.

Example: To display the Scheduler with single level resource grouping,

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleGroup Resources="@Resources"></ScheduleGroup>

```

```

<ScheduleResources>
<ScheduleResource TItem="ResourceData" TValue="int[]"
DataSource="@OwnersData" Field="OwnerId" Title="Owner" Name="Owners"
TextField="OwnerText" IdField="Id" ColorField="OwnerColor"
AllowMultiple="true"></ScheduleResource>
</ScheduleResources>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfScheduler>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 31);
public string[] Resources { get; set; } = { "Owners" };
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
0), OwnerId = 1 }
};
public List<ResourceData> OwnersData { get; set; } = new List<ResourceData>
{
new ResourceData{ OwnerText = "Nancy", Id = 1, OwnerColor = "#ffaa00" },
new ResourceData{ OwnerText = "Steven", Id = 2, OwnerColor = "#f8a398" },
new ResourceData{ OwnerText = "Michael", Id = 3, OwnerColor = "#7499e1" }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
public int OwnerId { get; set; }
}
public class ResourceData
{
public int Id { get; set; }
public string OwnerText { get; set; }
public string OwnerColor { get; set; }
}
}

```

The **Name** field defined in the **Resources** collection namely **Owners** will be mapped within the **Group** property, in order to enable the grouping option with those resource levels on the Scheduler.

Grouping multi-level resources

It is possible to group the resources of Scheduler in multiple levels, by mapping the child resources to each parent resource. In the following example, there are 2 levels of resources, on which the second level resources are defined with `GroupIDField` mapping to the first level resource's ID so as to establish the parent-child relationship between them.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
  <ScheduleGroup Resources="@Resources"></ScheduleGroup>
  <ScheduleResources>
    <ScheduleResource TItem="ResourceData" TValue="int" DataSource="@HotelData"
    Field="HotelId" Title="Hotel" Name="Hotels" TextField="HotelText"
    IdField="Id" ColorField="HotelColor"
    AllowMultiple="false"></ScheduleResource>
    <ScheduleResource TItem="ResourceData" TValue="int" DataSource="@RoomData"
    Field="RoomId" Title="Room" Name="Rooms" TextField="RoomText" IdField="Id"
    ColorField="RoomColor"
    GroupIDField="RoomGroupId" AllowMultiple="false"></ScheduleResource>
    <ScheduleResource TItem="ResourceData" TValue="int[]"
    DataSource="@OwnersData" Field="OwnerId" Title="Owner" Name="Owners"
    TextField="OwnerText" IdField="Id" GroupIDField="OwnerGroupId"
    ColorField="OwnerColor" AllowMultiple="true"></ScheduleResource>
  </ScheduleResources>
  <ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
  <ScheduleViews>
    <ScheduleView Option="View.Day"></ScheduleView>
    <ScheduleView Option="View.Week"></ScheduleView>
    <ScheduleView Option="View.WorkWeek"></ScheduleView>
    <ScheduleView Option="View.Month"></ScheduleView>
    <ScheduleView Option="View.Agenda"></ScheduleView>
  </ScheduleViews>
</SfSchedule>

@code{
DateTime CurrentDate = new DateTime(2020, 1, 31);
public string[] Resources { get; set; } = { "Hotels", "Rooms", "Owners" };
public List<ResourceData> HotelData { get; set; } = new List<ResourceData>
{
    new ResourceData{ HotelText = "Hotel 1", Id = 1, HotelColor = "#f8a398" },
    new ResourceData{ HotelText = "Hotel 2", Id = 2, HotelColor = "#ffaa00" }
};
public List<ResourceData> RoomData { get; set; } = new List<ResourceData>
{
    new ResourceData{ RoomText = "ROOM 1", Id = 1, RoomGroupId = 1, RoomColor =
"#cb6bb2" },
    new ResourceData{ RoomText = "ROOM 2", Id = 2, RoomGroupId = 2, RoomColor =
"#56ca85" }
};
public List<ResourceData> OwnersData { get; set; } = new List<ResourceData>
{
    new ResourceData{ OwnerText = "Nancy", Id = 1, OwnerGroupId = 1, OwnerColor
= "#ffaa00" },
    new ResourceData{ OwnerText = "Steven", Id = 2, OwnerGroupId = 2, OwnerColor
= "#f8a398" },
}
```

```

new ResourceData{ OwnerText = "Michael", Id = 3, OwnerGroupId = 1,
OwnerColor = "#7499e1" }
};
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
0),
OwnerId = 1, RoomId = 1, HotelId = 1 }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
public int OwnerId { get; set; }
public int RoomId { get; set; }
public int HotelId { get; set; }
}
public class ResourceData
{
public int Id { get; set; }
public string HotelText { get; set; }
public string HotelColor { get; set; }
public string RoomText { get; set; }
public string RoomColor { get; set; }
public string OwnerText { get; set; }
public string OwnerColor { get; set; }
public int RoomGroupId { get; set; }
public int OwnerGroupId { get; set; }
}
}

```

One-to-One grouping

In multi-level grouping, Scheduler usually groups the resources on the child level based on the `GroupIDField` that maps with the `IdField` field of parent level resources (as `ByGroupID` set to true by default). There are also option which allows to group all the child resource(s) against each of its parent resource(s). To enable this kind of grouping, set `false` to the `ByGroupID` option within the `Group` property. In the following code example, there are two levels of resources, on which all the resources at the child level is mapped one to one with each resource on the first level.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleGroup ByGroupID="false" Resources="@Resources"></ScheduleGroup>
<ScheduleResources>

```

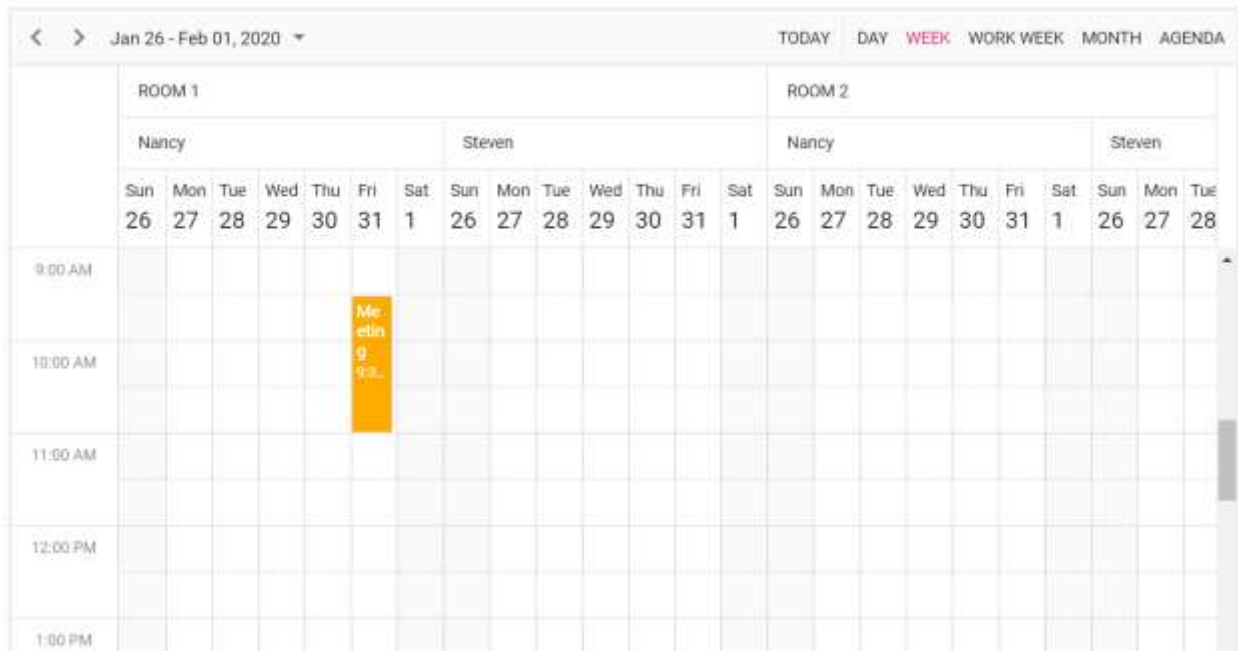
```

<ScheduleResource TItem="ResourceData" TValue="int" DataSource="@RoomData"
Field="RoomId" Title="Room" Name="Rooms" TextField="RoomText" IdField="Id"
ColorField="RoomColor" AllowMultiple="false"></ScheduleResource>
<ScheduleResource TItem="ResourceData" TValue="int[]"
DataSource="@OwnersData" Field="OwnerId" Title="Owner" Name="Owners"
TextField="OwnerText" IdField="Id" ColorField="OwnerColor"
AllowMultiple="true"></ScheduleResource>
</ScheduleResources>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfScheduler>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 31);
public string[] Resources { get; set; } = { "Rooms", "Owners" };
public List<ResourceData> RoomData { get; set; } = new List<ResourceData>
{
new ResourceData{ RoomText = "ROOM 1", Id = 1, RoomColor = "#cb6bb2" },
new ResourceData{ RoomText = "ROOM 2", Id = 2, RoomColor = "#56ca85" }
};
public List<ResourceData> OwnersData { get; set; } = new List<ResourceData>
{
new ResourceData{ OwnerText = "Nancy", Id = 1, OwnerColor = "#ffaa00" },
new ResourceData{ OwnerText = "Steven", Id = 2, OwnerColor = "#f8a398" }
};
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
0), OwnerId = 1, RoomId = 1 }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
public int OwnerId { get; set; }
public int RoomId { get; set; }
}
public class ResourceData
{
public int Id { get; set; }
public string RoomText { get; set; }
public string RoomColor { get; set; }
public string OwnerText { get; set; }
}

```

```
public string OwnerColor { get; set; }
}
}
```

The following image depicts how the scheduler will render when `ByGroupID` sets as false.



Grouping resources by date

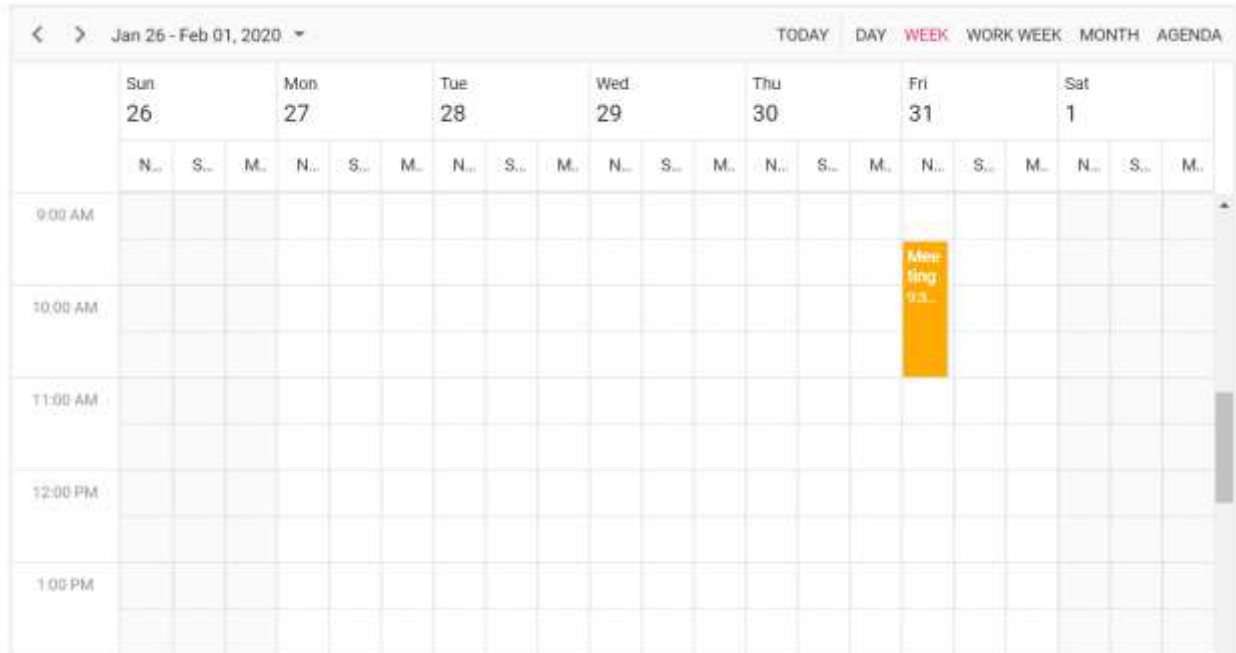
It groups the number of resources under each date and is applicable only on the calendar views such as Day, Week, Work Week, Month, Agenda and Month-Agenda. To enable such grouping, set `ByDate` option to `true` within the `Group` property.

Example: To display the Scheduler with resources grouped by date,

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleGroup ByDate="true" Resources="@Resources"></ScheduleGroup>
<ScheduleResources>
<ScheduleResource TItem="ResourceData" TValue="int[]"
DataSource="@OwnersData" Field="OwnerId" Title="Owner" Name="Owners"
TextField="OwnerText" IdField="Id"
GroupIDField="OwnerGroupId" ColorField="OwnerColor"
AllowMultiple="true"></ScheduleResource>
</ScheduleResources>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
```

```
</SfSchedule>
@code{
private DateTime CurrentDate = new DateTime(2020, 1, 31);
public string[] Resources { get; set; } = { "Owners" };
public List<ResourceData> OwnersData { get; set; } = new List<ResourceData>
{
    new ResourceData{ OwnerText = "Nancy", Id = 1, OwnerColor = "#ffaa00" },
    new ResourceData{ OwnerText = "Steven", Id = 2, OwnerColor = "#f8a398" },
    new ResourceData{ OwnerText = "Michael", Id = 3, OwnerColor = "#7499e1" }
};
List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
0), OwnerId = 1 }
};
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
    public int OwnerId { get; set; }
}
public class ResourceData
{
    public int Id { get; set; }
    public string OwnerText { get; set; }
    public string OwnerColor { get; set; }
}
}
```



This kind of grouping by date is not applicable on any of the **timeline views**.

Working with shared events

Multiple resources can share the same events, thus allowing the CRUD action made on it to reflect on all other shared instances simultaneously. To enable such option, set `AllowGroupEdit` option to `true` within the `Group` property. With this property enabled, a single appointment will be maintained within the appointment collection, even if it is shared by more than one resource – whereas the resource fields of such appointment will be in array which hold the IDs of the multiple resources.

Any actions such as create, edit or delete held on any one of the shared event instances, will be reflected on all other related instances visible on the UI.

Example: To edit all the resource events simultaneously,

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleGroup AllowGroupEdit="true" Resources="@Resources"></ScheduleGroup>
<ScheduleResources>
<ScheduleResource TItem="ResourceData" TValue="int[]"
DataSource="@ConferenceData" Field="ConferenceId" Title="Attendees"
Name="Conferences" TextField="Text" IdField="Id" ColorField="Color"
AllowMultiple="true"></ScheduleResource>
</ScheduleResources>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
```



```

</SfSchedule>
@code{
    DateTime CurrentDate = new DateTime(2020, 1, 31);
    public string[] Resources { get; set; } = { "Conferences" };
    List<AppointmentData> DataSource = new List<AppointmentData>
    {
        new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
        DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
        0),
        ConferenceId = new int[] { 1, 2, 3 } }
    };
    public List<ResourceData> ConferenceData { get; set; } = new
    List<ResourceData>
    {
        new ResourceData{ Text = "Margaret", Id = 1, Color = "#1aaa55"},
        new ResourceData{ Text = "Robert", Id = 2, Color = "#357cd2"},
        new ResourceData{ Text = "Laura", Id = 3, Color = "#7fa900"}
    };
    public class AppointmentData
    {
        public int Id { get; set; }
        public string Subject { get; set; }
        public string Location { get; set; }
        public DateTime StartTime { get; set; }
        public DateTime EndTime { get; set; }
        public string Description { get; set; }
        public bool IsAllDay { get; set; }
        public string RecurrenceRule { get; set; }
        public string RecurrenceException { get; set; }
        public Nullable<int> RecurrenceID { get; set; }
        public int[] ConferenceId { get; set; }
    }
    public class ResourceData
    {
        public int Id { get; set; }
        public string Text { get; set; }
        public string Color { get; set; }
    }
}

```

Simple resource header customization

It is possible to customize the resource header cells using built-in template option and change the look and appearance of it in both the vertical and timeline view modes. All the resource related fields and other information can be accessed within the resource header template option.

Example: To customize the resource header and display it along with designation and image, refer the following code example.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Width="100%" Height="650px" @bind-
SelectedDate="@CurrentDate">
    <ScheduleTemplates>
        <ResourceHeaderTemplate>
            @{

```

```

var resourceData = (context as TemplateContext).ResourceData as
ResourceData;
<div class='template-wrap'>
<div class="resource-image"></div>
<div class="resource-details">
<div class="resource-name">@(resourceData.Text) </div>
<div class="resource-designation">@(resourceData.Designation) </div>
</div>
</div>
}
</ResourceHeaderTemplate>
</ScheduleTemplates>
<ScheduleGroup Resources="@Resources"></ScheduleGroup>
<ScheduleResources>
<ScheduleResource TItem="ResourceData" TValue="int"
DataSource="@DoctorsData" Field="DoctorId" Title="Doctor Name"
Name="Doctors" TextField="Text" IdField="Id"
ColorField="Color"></ScheduleResource>
</ScheduleResources>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 4, 4);
public string[] Resources { get; set; } = { "Doctors" };
public List<ResourceData> DoctorsData { get; set; } = new List<ResourceData>
{
new ResourceData{ Text = "Will Smith", Id = 1, Color = "#ea7a57",
Designation = "Cardiologist", Image = "will-smith" },
new ResourceData{ Text = "Alice", Id = 2, Color = "rgb(53, 124, 210)",
Designation = "Neurologist", Image = "alice" },
new ResourceData{ Text = "Robson", Id = 3, Color = "#7fa900", Designation =
"Orthopedic Surgeon", Image = "robson" }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
public int DoctorId { get; set; }
}
public class ResourceData
{

```

```
public int Id { get; set; }
public string Text { get; set; }
public string Designation { get; set; }
public string Color { get; set; }
public string Image { get; set; }
}
}
<style>
.e-schedule .e-vertical-view .e-resource-cells {
height: 62px;
}
.e-schedule .template-wrap {
display: flex;
text-align: left;
}
.e-schedule .template-wrap .resource-image img {
width: 45px;
height: 45px;
}
.e-schedule .template-wrap .resource-details {
padding-left: 10px;
}
.e-schedule .template-wrap .resource-details .resource-name {
font-size: 16px;
font-weight: 500;
margin-top: 5px;
}
.e-schedule.e-device .template-wrap .resource-details .resource-name {
font-size: inherit;
font-weight: inherit;
}
.e-schedule.e-device .e-resource-tree-popup .e-fullrow {
height: 50px;
}
.e-schedule.e-device .template-wrap .resource-details .resource-designation
{
display: none;
}
</style>
```

<

>

Mar 29 - Apr 04, 2020

TODAY


DAY

WEEK

WORK WEEK


MONTH

AGENDA




Will Smith

Cardiologist



Alice

Neurologist



Robson

Orthopedic Surgeon

Sun

Mon

Tue

Wed

Thu

Fri

Sat

Sun

Mon

Tue

Wed

Thu

Fri

Sat

Sun

Mon

Tue

Wed

Thu

Fri

Sat

29

30

31

1

2

3

4

29

30

31

1

2

3

4

29

30

31

1

2

3

4

9:00 AM

10:00 AM

11:00 AM

12:00 PM

1:00 PM

To customize the resource header in compact mode properly make use of the class `e-device` as in the code example.

<


>

January 2020 ▾

⋮


≡

Margaret




Margaret

Fri
31



Robert



Laura

Customizing resource header with multiple columns

It is possible to customize the resource headers to display with multiple columns such as Room, Type and Capacity. The following code example depicts the way to achieve it and is applicable only on timeline views.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Width="100%" Height="650px" @bind-
SelectedDate="@CurrentDate" @bind-CurrentView="@CurrentView">
<ScheduleWorkHours Start="08:00" End="18:00"></ScheduleWorkHours>
<ScheduleTimeScale SlotCount="1" Interval="60"></ScheduleTimeScale>
<ScheduleGroup Resources="@GroupData"></ScheduleGroup>
<ScheduleResources>
<ScheduleResource TItem="ResourceData" TValue="int[]"
DataSource="@OwnersData" Field="OwnerId" Title="OwnerType" Name="Owner"
TextField="Text" IdField="Id" GroupIDField="OwnerGroupId"
ColorField="Ownercolor" AllowMultiple="true"></ScheduleResource>
</ScheduleResources>
<ScheduleTemplates>
<ResourceHeaderTemplate>
@{
var resourceData = (context as TemplateContext).ResourceData as
ResourceData;
<div class='template-wrap'>
<div class="room-name">@(resourceData.Text)</div>
<div class="room-capacity">@(resourceData.Capacity)</div>
<div class="room-type">@(resourceData.Type)</div>
<div class="room-avail">@(resourceData.Availability)</div>
</div>
}
</ResourceHeaderTemplate>
</ScheduleTemplates>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.TimelineWeek" MaxEventsPerRow="2"></ScheduleView>
<ScheduleView Option="View.TimelineMonth"
MaxEventsPerRow="2"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 31);
private View CurrentView = View.TimelineWeek;
public string[] GroupData { get; set; } = { "Owner" };
public List<ResourceData> OwnersData { get; set; } = new List<ResourceData>
{
new ResourceData{ Text = "Jammy", Id = 1, OwnerGroupId = 1, Ownercolor =
"#ea7a57", Capacity = 20, Type = "Conference", Availability = 10 },
new ResourceData{ Text = "Tweety", Id = 2, OwnerGroupId = 2, Ownercolor =
"#7fa900", Capacity = 7, Type = "Cabin", Availability = 5 },
new ResourceData{ Text = "Nestle", Id = 3, OwnerGroupId = 1, Ownercolor =
"#865fcf", Capacity = 5, Type = "Cabin", Availability = 2 },
new ResourceData{ Text = "Phoenix", Id = 4, OwnerGroupId = 2, Ownercolor =
"#fec200", Capacity = 15, Type = "Conference", Availability = 12 },
new ResourceData{ Text = "Rick Roll", Id = 5, OwnerGroupId = 1, Ownercolor =
"#865fcf", Capacity = 20, Type = "Conference", Availability = 7 },
```

```

new ResourceData{ Text = "Rainbow", Id = 6, OwnerGroupId = 2, Ownercolor =
"#1aaa55", Capacity = 8, Type = "Cabin", Availability = 3 }
};
public class ResourceData
{
    public int Id { get; set; }
    public string Text { get; set; }
    public int OwnerGroupId { get; set; }
    public string RoomColor { get; set; }
    public string Ownercolor { get; set; }
    public int Capacity { get; set; }
    public string Type { get; set; }
    public int Availability { get; set; }
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
0),
    OwnerId = 1 }
};
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
    public int OwnerId { get; set; }
}
}
<style>
.e-schedule .e-timeline-month-view .e-resource-left-td,
.e-schedule .e-timeline-view .e-resource-left-td {
vertical-align: bottom;
width: 350px;
}
.e-schedule.e-device .e-timeline-month-view .e-resource-left-td,
.e-schedule.e-device .e-timeline-view .e-resource-left-td {
width: 75px;
}
.e-schedule .e-timeline-month-view .e-resource-left-td .e-resource-text,
.e-schedule .e-timeline-view .e-resource-left-td .e-resource-text {
display: flex;
font-weight: 500;
padding: 0;
}
.e-schedule .e-timeline-month-view .e-resource-left-td .e-resource-text >
div,
.e-schedule .e-timeline-view .e-resource-left-td .e-resource-text > div {
border-right: 1px solid rgba(0, 0, 0, 0.12);
border-top: 1px solid rgba(0, 0, 0, 0.12);

```

```

flex: 0 0 20%;
font-weight: 500;
height: 36px;
line-height: 34px;
padding-left: 5px;
}
.e-schedule .e-timeline-month-view .e-resource-left-td .e-resource-text >
div:first-child,
.e-schedule .e-timeline-view .e-resource-left-td .e-resource-text >
div:first-child {
flex: 0 0 40%;
}
.e-schedule .e-timeline-month-view .e-resource-left-td .e-resource-text >
div:last-child,
.e-schedule .e-timeline-view .e-resource-left-td .e-resource-text >
div:last-child {
border-right: 0;
}
.e-schedule .e-schedule-table > tbody > tr > td {
width: 100%;
}
.e-schedule .e-timeline-view .e-resource-collapse,
.e-schedule .e-timeline-month-view .e-resource-collapse {
margin-top: 23px;
}
.e-schedule .e-parent-node .room-type {
flex: 0 0 20.8% !important;
}
.e-schedule .e-parent-node .room-capacity {
flex: 0 0 20.8% !important;
}
.e-schedule .e-parent-node .room-name {
flex: 0 0 37.8% !important;
}
.e-schedule .e-parent-node .room-avail {
flex: 0 0 20.8% !important;
}
.e-schedule .e-timeline-view .e-resource-tree-icon,
.e-schedule .e-timeline-month-view .e-resource-tree-icon {
margin-top: 22px !important;
}
.e-schedule .template-wrap {
display: flex;
height: 100%;
text-align: left;
}
.e-schedule .e-resource-cells .e-blazor-template {
height: 100%;
}
.e-schedule .template-wrap > div {
border-right: 1px solid rgba(0, 0, 0, 0.12);
flex: 0 0 20%;
font-weight: 500;
line-height: 58px;
overflow: hidden;
padding-left: 5px;
text-overflow: ellipsis;

```

```
}
.e-schedule .template-wrap > div:first-child {
flex: 0 0 40%;
}
.e-schedule .template-wrap > div:last-child {
border-right: 0;
}
.e-schedule .e-timeline-view .e-resource-cells,
.e-schedule .e-timeline-month-view .e-resource-cells {
padding-left: 0;
}
.e-schedule .e-timeline-view .e-date-header-wrap table col,
.e-schedule .e-timeline-view .e-content-wrap table col {
width: 100px;
}
.e-schedule .e-read-only {
opacity: .8;
}
@@media (max-width: 550px) {
.e-schedule .e-timeline-view .e-resource-left-td {
width: 100px;
}
.e-schedule .e-timeline-view .e-resource-left-td .e-resource-text > div,
.e-schedule .template-wrap > div {
flex: 0 0 100%;
}
.e-schedule .template-wrap > div:first-child {
border-right: 0;
}
.e-schedule .e-timeline-view .e-resource-left-td .e-resource-text >
div:first-child {
border-right: 0;
}
.e-schedule .room-type,
.e-schedule .room-capacity {
display: none;
}
}
</style>
```


< > Jan 26 - Feb 01, 2020 ▾				TODAY TIMELINE WEEK TIMELINE MONTH					
				Jan 31, Friday					
				8:00 AM	9:00 AM	10:00 AM	11:00 AM	12:00 PM	1:00 PM
Mission	25	Conference	15						
Jammy	20	Conference	10		Meeting 9:30 AM - 11:00 AM				
Nestle	5	Cabin	2						
Rick Roll	20	Conference	7						
Hangout	10	Cabin	8						
Tweety	7	Cabin	5						

Expand and collapse resource fields

It is possible to expand and collapse the resource field. By default, resource fields are expanded with their child fields. This behavior can be customized using the `ExpandedField` property. When `ExpandedField` property in resources dataSource is set to `false`, it restricts the resource fields from expanding. By default, `ExpandedField` value set to `true`.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleGroup Resources="@Resources"></ScheduleGroup>
<ScheduleResources>
<ScheduleResource TItem="ResourceData" TValue="int" DataSource="@RoomData"
Field="RoomId" Title="Room" Name="Rooms" TextField="RoomText" IdField="Id"
ColorField="RoomColor" AllowMultiple="false"
ExpandedField="IsExpand"></ScheduleResource>
<ScheduleResource TItem="ResourceData" TValue="int[]"
DataSource="@OwnersData" Field="OwnerId" Title="Owner" Name="Owners"
TextField="OwnerText" IdField="Id" GroupIDField="OwnerGroupId"
ColorField="OwnerColor" AllowMultiple="true"></ScheduleResource>
</ScheduleResources>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.TimelineWeek" MaxEventsPerRow="2"></ScheduleView>
<ScheduleView Option="View.TimelineMonth"
MaxEventsPerRow="2"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
private DateTime CurrentDate = new DateTime(2020, 1, 31);
public string[] Resources { get; set; } = { "Rooms", "Owners" };
public List<ResourceData> RoomData { get; set; } = new List<ResourceData>
```

```

{
    new ResourceData{ RoomText = "ROOM 1", Id = 1, RoomColor =
    "#cb6bb2",IsExpand=true},
    new ResourceData{ RoomText = "ROOM 2", Id = 2, RoomColor = "#56ca85" ,
    IsExpand=false }
};
public List<ResourceData> OwnersData { get; set; } = new List<ResourceData>
{
    new ResourceData{ OwnerText = "Nancy", Id = 1, OwnerGroupId = 1, OwnerColor
    = "#ffaa00" },
    new ResourceData{ OwnerText = "Steven", Id = 2, OwnerGroupId = 2, OwnerColor
    = "#f8a398" },
    new ResourceData{ OwnerText = "Michael", Id = 3, OwnerGroupId = 1,
    OwnerColor = "#7499e1" }
};
List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
    DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
    0), OwnerId = 1, RoomId = 1 }
};
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
    public int OwnerId { get; set; }
    public int RoomId { get; set; }
}
public class ResourceData
{
    public int Id { get; set; }
    public string RoomText { get; set; }
    public string RoomColor { get; set; }
    public string OwnerText { get; set; }
    public string OwnerColor { get; set; }
    public int OwnerGroupId { get; set; }
    public bool IsExpand { get; set; }
}
}

```

Displaying tooltip for resource headers

It is possible to display tooltip over the resource headers showing the resource information. By default, there won't be any tooltip displayed on the resource headers, and to enable it, you need to assign the customized template design to the `HeaderTooltipTemplate` option within the `ScheduleGroup`.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
```

```

<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleGroup Resources="@Resources">
<HeaderTooltipTemplate>
@{
var resourceData = (context as TemplateContext).ResourceData as
ResourceData;
<div class='template-wrap'>
<div class="resource-image"></div>
<div class="resource-details">
<div class="resource-name">@(resourceData.Text) </div>
</div>
</div>
</HeaderTooltipTemplate>
</ScheduleGroup>
<ScheduleResources>
<ScheduleResource TValue="int[]" TItem="ResourceData"
DataSource="@ConferenceData" Field="ConferenceId" Title="Attendees"
Name="Conferences" TextField="Text" IdField="Id" ColorField="Color"
AllowMultiple="true"></ScheduleResource>
</ScheduleResources>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 31);
public string[] Resources { get; set; } = { "Conferences" };
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
0),
ConferenceId = 1 }
};
public List<ResourceData> ConferenceData { get; set; } = new
List<ResourceData>
{
new ResourceData{ Text = "Margaret", Id = 1, Color = "#1aaa55", Image=
"margaret" },
new ResourceData{ Text = "Robert", Id = 2, Color = "#357cd2", Image=
"robert" },
new ResourceData{ Text = "Laura", Id = 3, Color = "#7fa900", Image= "laura"
}
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
}

```

```

public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
public int ConferenceId { get; set; }
}
public class ResourceData
{
public int Id { get; set; }
public string Image { get; set; }
public string Text { get; set; }
public string Color { get; set; }
}
}

```

Choosing between resource colors for appointments

By default, the colors defined on the top level resources collection will be applied for the events. In case, if you want to apply specific resource color to events irrespective of its top-level parent resource color, it can be achieved by defining `ResourceColorField` option within the `EventSettings` property.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Buttons
<SfRadioButton Label="Hotels" Name="Select" Value="Hotels" @bind-
Checked="@ResourceColor"></SfRadioButton>
<SfRadioButton Label="Rooms" Name="Select" Value="Rooms" @bind-
Checked="@ResourceColor"></SfRadioButton>
<SfRadioButton Label="Owners" Name="Select" Value="Owners" @bind-
Checked="@ResourceColor"></SfRadioButton>
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleGroup Resources="@Resources"></ScheduleGroup>
<ScheduleResources>
<ScheduleResource TItem="ResourceData" TValue="int" DataSource="@HotelData"
Field="HotelId" Title="Hotel" Name="Hotels" TextField="HotelText"
IdField="Id" ColorField="HotelColor"
AllowMultiple="false"></ScheduleResource>
<ScheduleResource TItem="ResourceData" TValue="int" DataSource="@RoomData"
Field="RoomId" Title="Room" Name="Rooms" TextField="RoomText" IdField="Id"
ColorField="RoomColor" GroupIDField="RoomGroupId"
AllowMultiple="false"></ScheduleResource>
<ScheduleResource TItem="ResourceData" TValue="int[]"
DataSource="@OwnersData" Field="OwnerId" Title="Owner" Name="Owners"
TextField="OwnerText" IdField="Id"
GroupIDField="OwnerGroupId" ColorField="OwnerColor"
AllowMultiple="true"></ScheduleResource>
</ScheduleResources>
<ScheduleEventSettings DataSource="@DataSource"
ResourceColorField="@ResourceColor"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>

```

```

<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
    DateTime CurrentDate = new DateTime(2020, 1, 31);
    public string ResourceColor { get; set; } = "Rooms";
    public string[] Resources { get; set; } = { "Hotels", "Rooms", "Owners" };
    public List<ResourceData> HotelData { get; set; } = new List<ResourceData>
    {
        new ResourceData{ HotelText = "Hotel 1", Id = 1, HotelColor = "#f8a398" },
        new ResourceData{ HotelText = "Hotel 2", Id = 2, HotelColor = "#ffaa00" }
    };
    public List<ResourceData> RoomData { get; set; } = new List<ResourceData>
    {
        new ResourceData{ RoomText = "ROOM 1", Id = 1, RoomGroupId = 1, RoomColor = "#cb6bb2" },
        new ResourceData{ RoomText = "ROOM 2", Id = 2, RoomGroupId = 2, RoomColor = "#56ca85" }
    };
    public List<ResourceData> OwnersData { get; set; } = new List<ResourceData>
    {
        new ResourceData{ OwnerText = "Nancy", Id = 1, OwnerGroupId = 1, OwnerColor = "#ffaa00" },
        new ResourceData{ OwnerText = "Steven", Id = 2, OwnerGroupId = 2, OwnerColor = "#f8a398" },
        new ResourceData{ OwnerText = "Michael", Id = 3, OwnerGroupId = 1, OwnerColor = "#7499e1" }
    };
    List<AppointmentData> DataSource = new List<AppointmentData>
    {
        new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
        DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0, 0),
        OwnerId = 1, RoomId = 1, HotelId = 1 }
    };
    public class AppointmentData
    {
        public int Id { get; set; }
        public string Subject { get; set; }
        public string Location { get; set; }
        public DateTime StartTime { get; set; }
        public DateTime EndTime { get; set; }
        public string Description { get; set; }
        public bool IsAllDay { get; set; }
        public string RecurrenceRule { get; set; }
        public string RecurrenceException { get; set; }
        public Nullable<int> RecurrenceID { get; set; }
        public int OwnerId { get; set; }
        public int RoomId { get; set; }
        public int HotelId { get; set; }
    }
    public class ResourceData
    {
        public int Id { get; set; }
    }
}

```

```

public string HotelText { get; set; }
public string RoomText { get; set; }
public string OwnerText { get; set; }
public string HotelColor { get; set; }
public string RoomColor { get; set; }
public string OwnerColor { get; set; }
public int RoomGroupId { get; set; }
public int OwnerGroupId { get; set; }
}
}

```

The value of the **ResourceColorField** field should be mapped with the **Name** value given within the **ScheduleResource**.

Setting different working days and hours for resources

Each resource in the Scheduler can have different working hours as well as different working days set to it. There are default options available within the **ScheduleResource** collection, to customize the default working hours and days of the Scheduler.

Set different work days

Different working days can be set for the resources of Scheduler using the **WorkDaysField** property which maps the working days field from the resource dataSource. This field accepts the collection of day indexes (from 0 to 6) of a week. By default, it is set to [1, 2, 3, 4, 5] and in the following example, each resource has been set with different values and therefore each of them will render only those working days. This option is applicable only on the calendar views and is not applicable on timeline views.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate" @bind-CurrentView="@CurrentView">
<ScheduleGroup Resources="@Resources"></ScheduleGroup>
<ScheduleResources>
<ScheduleResource TItem="ResourceData" TValue="int"
DataSource="@DoctorsData" Field="DoctorId" Title="Doctor Name"
Name="Doctors" TextField="Text"
IdField="Id" ColorField="Color" WorkDaysField="WorkDays"></ScheduleResource>
</ScheduleResources>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 31);
View CurrentView = View.WorkWeek;
public string[] Resources { get; set; } = { "Doctors" };
public List<ResourceData> DoctorsData { get; set; } = new List<ResourceData>
{
new ResourceData{ Text = "Will Smith", Id = 1, Color = "#ea7a57", WorkDays =
new int[] { 1, 2, 4, 5 } },

```

```

new ResourceData{ Text = "Alice", Id = 2, Color = "rgb(53, 124, 210)",
WorkDays = new int[] { 1, 3, 5 } },
new ResourceData{ Text = "Robson", Id = 3, Color = "#7fa900" }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
public int DoctorId { get; set; }
}
public class ResourceData
{
public int Id { get; set; }
public string Text { get; set; }
public string Color { get; set; }
public int[] WorkDays { get; set; }
}
}

```

<div> <div>< ></div> <div>January 27 - 31, 2020</div> <div>TODAY DAY WEEK WORK WEEK MONTH AGENDA</div> </div>												
Will Smith				Alice			Robson					
Mon	Tue	Thu	Fri	Mon	Wed	Fri	Mon	Tue	Wed	Thu	Fri	
27	28	30	31	27	29	31	27	28	29	30	31	
9:00 AM												
10:00 AM												
11:00 AM												
12:00 PM												
1:00 PM												

Set different work hours

Working hours indicates the work hour duration of a day, which is highlighted visually with active color over the work cells. Each resource on the Scheduler can be defined with its own set of working hours as depicted in the following example.

- **StartHourField** - Denotes the start time of the working/business hour in a day.
- **EndHourField** - Denotes the end time limit of the working/business hour in a day.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleGroup Resources="@Resources"></ScheduleGroup>
<ScheduleResources>
<ScheduleResource TItem="ResourceData" TValue="int"
DataSource="@DoctorsData" Field="DoctorId" Title="Doctor Name"
Name="Doctors" TextField="Text" IdField="Id"
ColorField="Color" WorkDaysField="WorkDays" StartHourField="StartHour"
EndHourField="EndHour"></ScheduleResource>
</ScheduleResources>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 31);
public string[] Resources { get; set; } = { "Doctors" };
public List<ResourceData> DoctorsData { get; set; } = new List<ResourceData>
{
new ResourceData{ Text = "Will Smith", Id = 1, Color = "#ea7a57", StartHour
= "07:00", EndHour = "13:00"},
new ResourceData{ Text = "Alice", Id = 2, Color = "rgb(53, 124, 210)",
StartHour = "09:00", EndHour = "17:00"},
new ResourceData{ Text = "Robson", Id = 3, Color = "#7fa900", StartHour =
"08:00", EndHour = "16:00"}
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
public int DoctorId { get; set; }
}
public class ResourceData
{
public int Id { get; set; }
public string Text { get; set; }
public string Color { get; set; }
public string StartHour { get; set; }
```



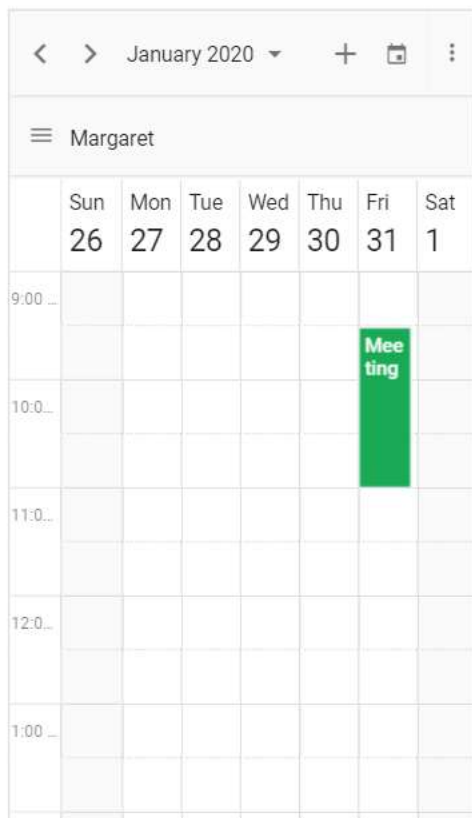
```
public string EndHour { get; set; }  
}  
}
```

In this example, a resource named **Will Smith** is depicted with working hours ranging from 8.00 AM to 3.00 PM and is visually illustrated with active colors, whereas the other two resources have different working hours set.

Compact view in mobile

Although the Scheduler views are designed keeping in mind the responsiveness of the control in mobile devices, however when using Scheduler with multiple resources - it is difficult to view all the resources and its relevant events at once on the mobile. Therefore, a new compact mode has been introduced specially for displaying multiple resources of Scheduler on mobile devices. By default, this mode is enabled while using Scheduler with multiple resources on the mobile devices. If in case, you need to disable this compact mode, set `false` to the `EnableCompactView` option within the `ScheduleGroup`. Disabling this option will display the exact desktop mode of Scheduler view on mobile devices.

With this compact view enabled on mobile, you can view only single resource at a time and to switch to other resources, there is a `TreeView` at the left listing out all other available resources - clicking on which will display that particular resource and its related appointments.



Adaptive UI in desktop

By default, the Scheduler layout adapts automatically in the desktop and mobile devices with appropriate UI changes. In case, if the user wants to display the Adaptive scheduler in desktop mode

with adaptive enhancements, then the property `EnableAdaptiveUI` can be set to true. Enabling this option will display the exact mobile mode of Scheduler view on desktop devices.

Some of the default changes made for compact Scheduler to render in desktop devices are as follows,

- View options displayed in the Navigation drawer.
- Plus icon is added to the header for new event creation.
- Today icon is added to the header instead of the Today button.
- With Multiple resources – only one resource has been shown to enhance the view experience of resource events details clearly. To switch to other resources, there is a TreeView on the left that lists all other available resources, clicking on which will display that particular resource and its related events.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Width="100%" Height="650px" @bind-
SelectedDate="@CurrentDate" @bind-CurrentView="@CurrentView"
EnableAdaptiveUI="true">
<ScheduleGroup Resources="@GroupData"></ScheduleGroup>
<ScheduleResources>
<ScheduleResource TItem="ResourceData" TValue="int"
DataSource="@ProjectData" Field="ProjectId" Title="Choose Project"
Name="Projects" TextField="Text" IdField="Id"
ColorField="Color"></ScheduleResource>
<ScheduleResource TItem="ResourceData" TValue="int[]" DataSource="@TaskData"
Field="TaskId" Title="Category" Name="Categories" TextField="Text"
IdField="Id" GroupIDField="GroupId" ColorField="Color"
AllowMultiple="true"></ScheduleResource>
</ScheduleResources>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
</ScheduleViews>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
</SfSchedule>
@code{
private View CurrentView = View.Month;
public DateTime CurrentDate = new DateTime(2020, 1, 31);
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
0), ProjectId = 1, TaskId = 1 }
};
private string[] GroupData = new string[] { "Projects", "Categories" };
private List<ResourceData> ProjectData { get; set; } = new
List<ResourceData> {
new ResourceData {Text = "PROJECT 1", Id= 1, Color= "#cb6bb2"},
new ResourceData {Text = "PROJECT 2", Id= 2, Color= "#56ca85"},
new ResourceData {Text = "PROJECT 3", Id= 3, Color= "#df5286"}
};
private List<ResourceData> TaskData { get; set; } = new List<ResourceData> {
new ResourceData { Text = "Nancy", Id= 1, GroupId = 1, Color = "#df5286" },
}
```

```

new ResourceData { Text = "Steven", Id= 2, GroupId = 1, Color = "#7fa900" },
new ResourceData { Text = "Robert", Id= 3, GroupId = 2, Color = "#ea7a57" },
new ResourceData { Text = "Smith", Id= 4, GroupId = 2, Color = "#5978ee" },
new ResourceData { Text = "Michael", Id= 5, GroupId = 3, Color = "#df5286"
},
new ResourceData { Text = "Root", Id= 6, GroupId = 3, Color = "#00bdae" }
};
public class ResourceData
{
    public string Text { get; set; }
    public int Id { get; set; }
    public int GroupId { get; set; }
    public string Color { get; set; }
}
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
    public int ProjectId { get; set; }
    public int TaskId { get; set; }
}
}

```

See Also

[How to expand or collapse a resource programmatically](#)

Timeline Header Rows in Blazor Scheduler Component

The Timeline views can have additional header rows other than its default date and time header rows. It is possible to show individual header rows for displaying year, month and week separately using the `ScheduleHeaderRow` which is applicable only on the timeline views. The possible rows which can be added using `ScheduleHeaderRow` are as follows.

- Year
- Month
- Week
- Date
- Hour

To get start quickly on customizing the header rows of timeline views on scheduler, you can check on this video:

{% youtube

"youtube:https://www.youtube.com/watch?v=2eYGFgwqK6U"%}

The **Hour** row is not applicable for Timeline month view.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="650px">
  <ScheduleHeaderRows>
    <ScheduleHeaderRow Option="HeaderRowType.Year"></ScheduleHeaderRow>
    <ScheduleHeaderRow Option="HeaderRowType.Month"></ScheduleHeaderRow>
    <ScheduleHeaderRow Option="HeaderRowType.Week"></ScheduleHeaderRow>
    <ScheduleHeaderRow Option="HeaderRowType.Date"></ScheduleHeaderRow>
    <ScheduleHeaderRow Option="HeaderRowType.Hour"></ScheduleHeaderRow>
  </ScheduleHeaderRows>
  <ScheduleViews>
    <ScheduleView Option="View.TimelineWeek"
      MaxEventsPerRow="10"></ScheduleView>
  </ScheduleViews>
</SfSchedule>

@code{
public class AppointmentData
{
  public int Id { get; set; }
  public string Subject { get; set; }
  public string Location { get; set; }
  public DateTime StartTime { get; set; }
  public DateTime EndTime { get; set; }
  public string Description { get; set; }
  public bool IsAllDay { get; set; }
  public string RecurrenceRule { get; set; }
  public string RecurrenceException { get; set; }
  public Nullable<int> RecurrenceID { get; set; }
}
}
```

Display year and month rows in timeline views

To display the timeline Scheduler simply with year and month names alone, define the option **Year** and **Month** within the **ScheduleHeaderRow** property.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="650px">
  <ScheduleHeaderRows>
    <ScheduleHeaderRow Option="HeaderRowType.Year"></ScheduleHeaderRow>
    <ScheduleHeaderRow Option="HeaderRowType.Month"></ScheduleHeaderRow>
  </ScheduleHeaderRows>
  <ScheduleViews>
    <ScheduleView Option="View.TimelineMonth" MaxEventsPerRow="10"
      Interval="24"></ScheduleView>
  </ScheduleViews>
</SfSchedule>

@code{
public class AppointmentData
{
  public int Id { get; set; }
  public string Subject { get; set; }
}
```

```

public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}

```

Display week numbers in timeline views

The week number can be displayed in a separate header row of the timeline Scheduler by setting **Week** option within **ScheduleHeaderRow** property.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="650px">
  <ScheduleHeaderRows>
    <ScheduleHeaderRow Option="HeaderRowType.Week"></ScheduleHeaderRow>
    <ScheduleHeaderRow Option="HeaderRowType.Date"></ScheduleHeaderRow>
    <ScheduleHeaderRow Option="HeaderRowType.Hour"></ScheduleHeaderRow>
  </ScheduleHeaderRows>
  <ScheduleViews>
    <ScheduleView Option="View.TimelineWeek"
      MaxEventsPerRow="10"></ScheduleView>
    <ScheduleView Option="View.TimelineMonth"
      MaxEventsPerRow="10"></ScheduleView>
  </ScheduleViews>
</SfSchedule>
@code{
public class AppointmentData
{
  public int Id { get; set; }
  public string Subject { get; set; }
  public string Location { get; set; }
  public DateTime StartTime { get; set; }
  public DateTime EndTime { get; set; }
  public string Description { get; set; }
  public bool IsAllDay { get; set; }
  public string RecurrenceRule { get; set; }
  public string RecurrenceException { get; set; }
  public Nullable<int> RecurrenceID { get; set; }
}
}

```

Timeline view displaying dates of a complete year

It is possible to display a complete year in a timeline view by setting **Interval** value as 12 and defining **TimelineMonth** view option within the **ScheduleView** tag helper.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="650px">

```

```

<ScheduleHeaderRows>
<ScheduleHeaderRow Option="HeaderRowType.Month"></ScheduleHeaderRow>
<ScheduleHeaderRow Option="HeaderRowType.Date"></ScheduleHeaderRow>
</ScheduleHeaderRows>
<ScheduleViews>
<ScheduleView Option="View.TimelineMonth" MaxEventsPerRow="10"
Interval="12"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

Customizing the header rows using template

The text of the header rows can be customized and display any images or format text on each individual header rows using the built-in **Template** option available within the **ScheduleHeaderRow**.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
@using System.Globalization
<p>Timeline header rows</p>
<SfSchedule TValue="AppointmentData" Height="650px">
<ScheduleHeaderRows>
<ScheduleHeaderRow Option="HeaderRowType.Year">
<Template>
<div class="date-text">Year: @(getYearText((context as
TemplateContext).Date))</div>
</Template>
</ScheduleHeaderRow>
<ScheduleHeaderRow Option="HeaderRowType.Month">
<Template>
<div class="date-text">Month: @(getMonthText((context as
TemplateContext).Date))</div>
</Template>
</ScheduleHeaderRow>
<ScheduleHeaderRow Option="HeaderRowType.Week">
<Template>
<div class="date-text">Week: @(getWeekText((context as
TemplateContext).Date))</div>
</Template>
</ScheduleHeaderRow>
<ScheduleHeaderRow Option="HeaderRowType.Date"></ScheduleHeaderRow>
</ScheduleHeaderRows>

```

```

<ScheduleViews>
<ScheduleView Option="View.TimelineWeek"
MaxEventsPerRow="10"></ScheduleView>
</ScheduleViews>
</SfScheduler>
@code{
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
public static string getYearText(DateTime date)
{
return date.ToString("yyyy", CultureInfo.InvariantCulture);
}
public static string getMonthText(DateTime date)
{
return date.ToString("MMMM", CultureInfo.InvariantCulture);
}
public static string getWeekText(DateTime date)
{
return CultureInfo.InvariantCulture.Calendar.GetWeekOfYear(date,
CalendarWeekRule.FirstFourDayWeek, DayOfWeek.Monday).ToString();
}
}
}

```

Row Auto Height in Blazor Scheduler Component

By default, the height of the Scheduler rows in Timeline views are static and therefore, when the same time range holds multiple overlapping appointments, a **+n more** text indicator will be displayed. With this feature enabled, you can now view all the overlapping appointments present in those specific time range by auto-adjusting the row height based on the presence of the appointments count, instead of displaying the **+n more** text indicators.

To enable auto row height adjustments on Scheduler Timeline views and Month view, set **true** to the **EnableAutoRowHeight** property whose default value is **false**.

This auto row height adjustment is applicable only on all the Timeline views as well as on the calendar Month view.

Now, let's see how it works on those applicable views with examples.

When the feature **EnableAutoRowHeight** is enabled, the row height gets auto-adjusted based on the number of overlapping events occupied on the same time range, which is demonstrated in the following example.

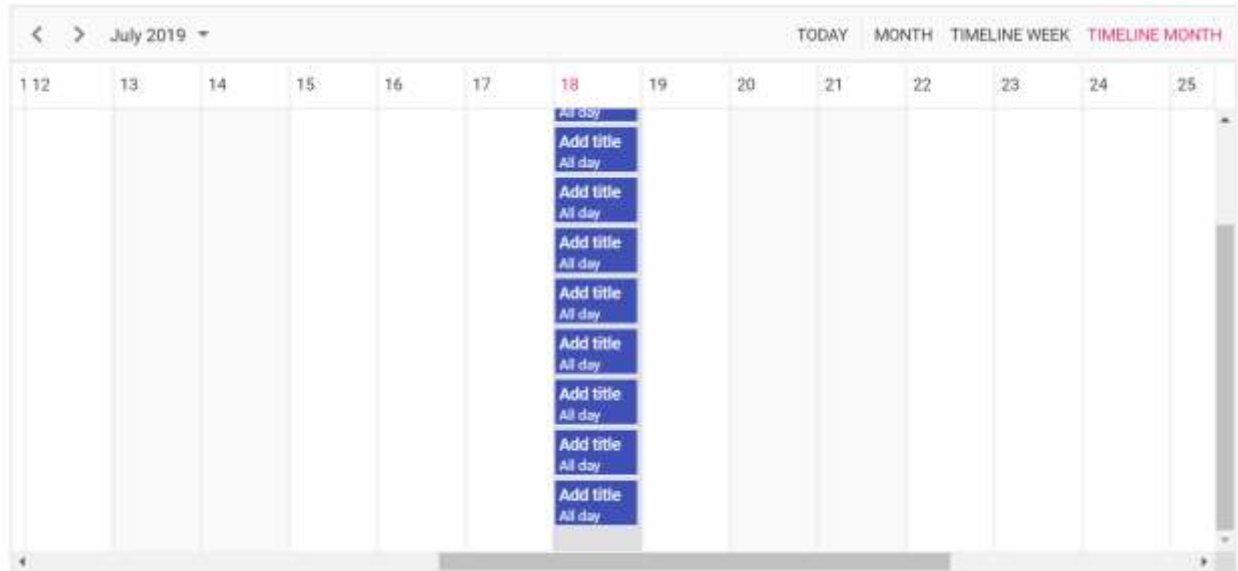
ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="650px"
EnableAutoRowHeight="true">
<ScheduleViews>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.TimelineWeek"></ScheduleView>
<ScheduleView Option="View.TimelineMonth"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

<div> <div>< ></div> <div>July 2019</div> <div>TODAY</div> <div>MONTH</div> <div>TIMELINE WEEK</div> <div>TIMELINE MONTH</div> </div>						
Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
30	Jul 1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	<div>18</div> <div>Meeting</div> <div>Conference</div> <div>Office trip</div>	19	20
21	22	23	24	25	26	27
28	29	30	31	Aug 1	2	3



Timeline views with multiple resources

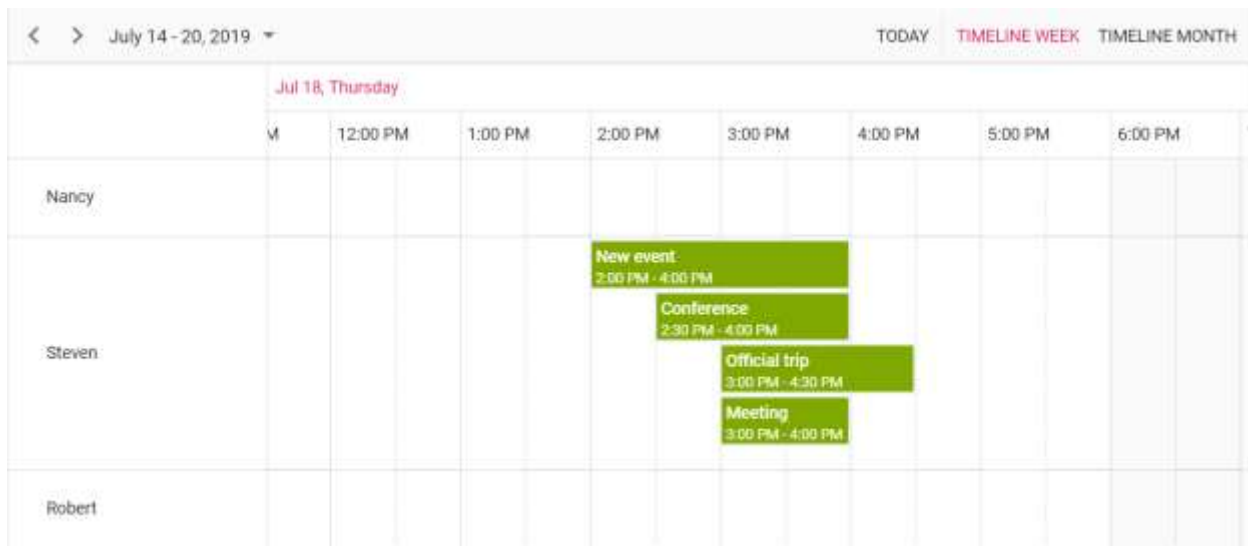
ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="650px"
EnableAutoRowHeight="true">
<ScheduleGroup Resources="@Resources"></ScheduleGroup>
<ScheduleResources>
<ScheduleResource TItem="ResourceData" TValue="int" DataSource="@OwnerData"
Field="OwnerId" Title="Owner" Name="Owner" TextField="Text" IdField="Id"
ColorField="Color"></ScheduleResource>
</ScheduleResources>
<ScheduleViews>
<ScheduleView Option="View.TimelineWeek"></ScheduleView>
<ScheduleView Option="View.TimelineMonth"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
public string[] Resources { get; set; } = { "Owner" };
private List<ResourceData> OwnerData { get; set; } = new List<ResourceData>
{
new ResourceData { Text = "Nancy", Id= 1, Color = "#df5286" },
new ResourceData { Text = "Steven", Id= 2, Color = "#7fa900" },
new ResourceData { Text = "Robert", Id= 3, Color = "#ea7a57" }
};
public class ResourceData
{
public int Id { get; set; }
public string Text { get; set; }
public string Color { get; set; }
}
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
```

```

public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
public int OwnerId { get; set; }
}
}

```



Appointments occupying entire cell

By default, with the feature `EnableAutoRowHeight`, there will be a space in the bottom of the cell when appointment is rendered. To avoid this space, set true to the property `IgnoreWhitespace` with `ScheduleEventSettings` tag helper whereas its default property value is false.

ASPX-CS

```

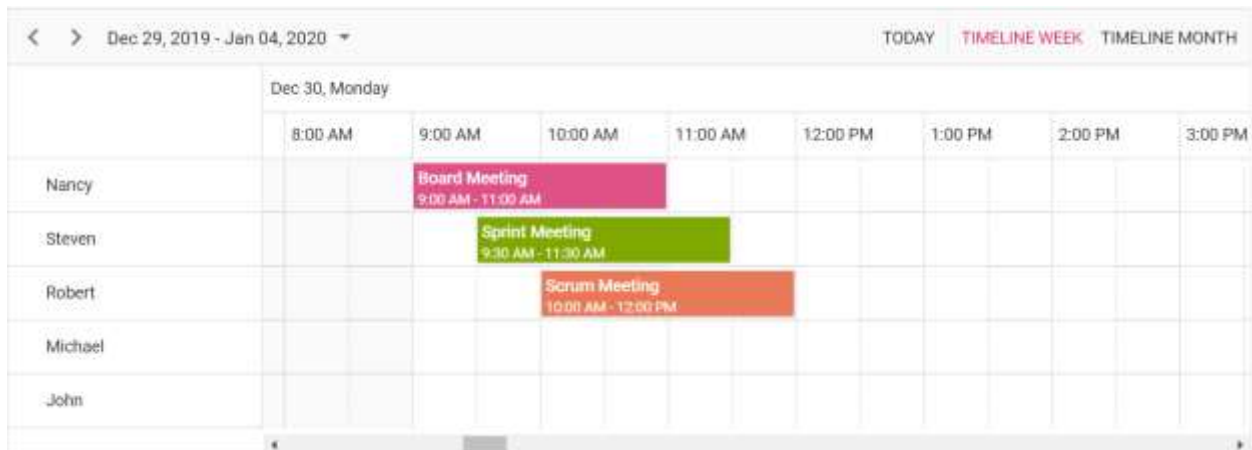
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="350px"
EnableAutoRowHeight="true" @bind-SelectedDate="@CurrentDate">
<ScheduleGroup Resources="@Resources"></ScheduleGroup>
<ScheduleResources>
<ScheduleResource TItem="ResourceData" TValue="int" DataSource="@OwnerData"
Field="OwnerId" Title="Owner" Name="Owner" TextField="Text" IdField="Id"
ColorField="Color"></ScheduleResource>
</ScheduleResources>
<ScheduleEventSettings DataSource="@DataSource"
IgnoreWhitespace="true"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.TimelineWeek"></ScheduleView>
<ScheduleView Option="View.TimelineMonth"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 12, 30);
public string[] Resources { get; set; } = { "Owner" };
private List<ResourceData> OwnerData { get; set; } = new List<ResourceData>

```

```

{
    new ResourceData { Text = "Nancy", Id= 1, Color = "#df5286" },
    new ResourceData { Text = "Steven", Id= 2, Color = "#7fa900" },
    new ResourceData { Text = "Robert", Id= 3, Color = "#ea7a57" },
    new ResourceData { Text = "Michael", Id= 4, Color = "#df5286" },
    new ResourceData { Text = "John", Id= 5, Color = "#7fa900" }
};
private List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData { Id = 1, Subject = "Board Meeting", StartTime = new
    DateTime(2020, 12, 30, 9, 0, 0), EndTime = new DateTime(2020, 12, 30, 11, 0,
    0), OwnerId = 1},
    new AppointmentData { Id = 2, Subject = "Sprint Meeting", StartTime = new
    DateTime(2020, 12, 30, 9, 30, 0), EndTime = new DateTime(2020, 12, 30, 11,
    30, 0), OwnerId = 2},
    new AppointmentData { Id = 3, Subject = "Scrum Meeting", StartTime = new
    DateTime(2020, 12, 30, 10, 0, 0), EndTime = new DateTime(2020, 12, 30, 12,
    0, 0), OwnerId = 3}
};
public class ResourceData
{
    public int Id { get; set; }
    public string Text { get; set; }
    public string Color { get; set; }
}
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
    public int OwnerId { get; set; }
}
}

```



The property `IgnoreWhitespace` will be applicable only when `EnableAutoRowHeight` feature is enabled in the Scheduler.

Header Customization in Blazor Scheduler Component

The header part of Scheduler can be customized easily with the built-in options available.

Show or Hide header bar

By default, the header bar holds the date and view navigation options, through which the user can switch between the dates and various views. This header bar can be hidden from the UI by setting `false` to the `ShowHeaderBar` property. It's default value is `true`.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" ShowHeaderBar="false" Height="550px"
@bind-SelectedDate="@CurrentDate">
  <ScheduleViews>
    <ScheduleView Option="View.Day"></ScheduleView>
    <ScheduleView Option="View.Week"></ScheduleView>
    <ScheduleView Option="View.WorkWeek"></ScheduleView>
    <ScheduleView Option="View.Month"></ScheduleView>
    <ScheduleView Option="View.Agenda"></ScheduleView>
  </ScheduleViews>
</SfSchedule>
@code{
  DateTime CurrentDate = new DateTime(2020, 1, 31);
  public class AppointmentData
  {
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
  }
}
```

How to display the view options within the header bar popup

By default, the header bar holds the view navigation options, through which the user can switch between various views. You can move this view options to the header bar popup by setting `true` to the `EnableAdaptiveUI` property.

ASPX-CS

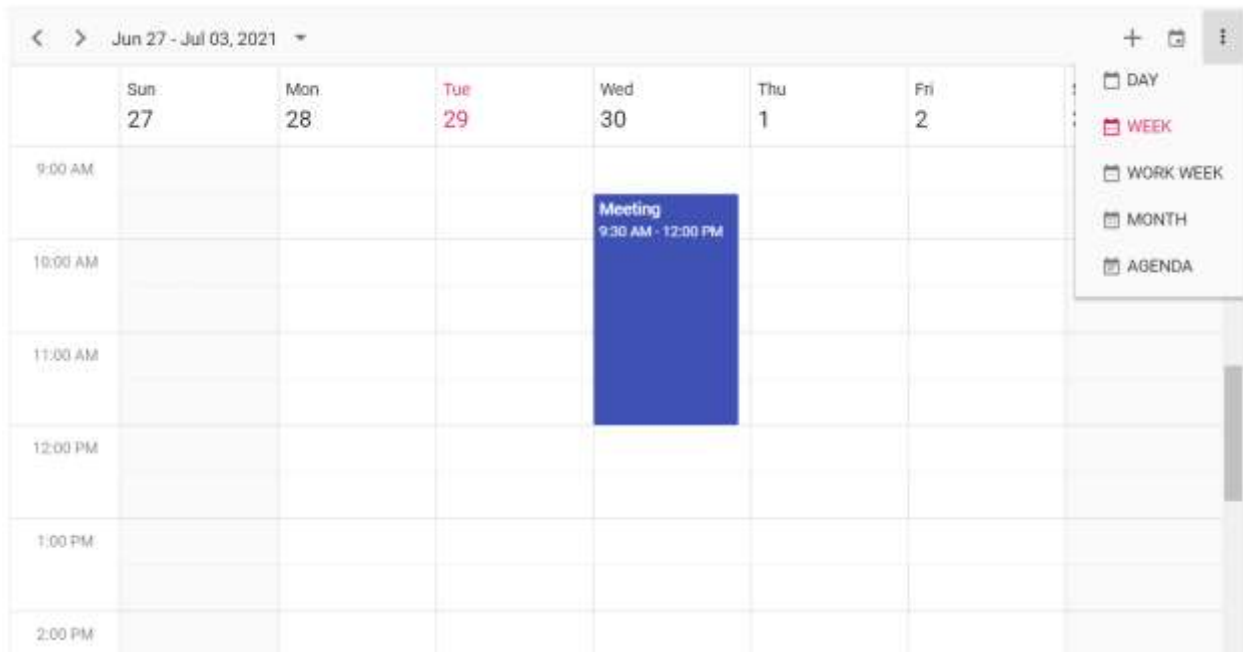
```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" EnableAdaptiveUI="true" Height="550px"
@bind-SelectedDate="@CurrentDate">
  <ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
  <ScheduleViews>
    <ScheduleView Option="View.Day"></ScheduleView>
    <ScheduleView Option="View.Week"></ScheduleView>
  </ScheduleViews>
</SfSchedule>
```

```

<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfScheduler>
@code{
    DateTime CurrentDate = new DateTime(2021, 6, 30);
    List<AppointmentData> DataSource = new List<AppointmentData>
    {
        new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
        DateTime(2021, 6, 30, 9, 30, 0) , EndTime = new DateTime(2021, 6, 30, 12, 0,
        0) }
    };
    public class AppointmentData
    {
        public int Id { get; set; }
        public string Subject { get; set; }
        public string Location { get; set; }
        public DateTime StartTime { get; set; }
        public DateTime EndTime { get; set; }
        public string Description { get; set; }
        public bool IsAllDay { get; set; }
        public string RecurrenceRule { get; set; }
        public string RecurrenceException { get; set; }
        public Nullable<int> RecurrenceID { get; set; }
    }
}

```

The Scheduler with view options within the header bar popup will be rendered as shown in the following image.



Refer [here](#) to know more about adaptive UI in resources scheduler.

Date header customization

The Scheduler UI that displays the date text on all views are considered as the date header cells. You can customize the date header cells of Scheduler using `DateHeaderTemplate`. The `DateHeaderTemplate` option is used to customize the date header cells of day, week and work-week views.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
@using System.Globalization

<SfSchedule TValue="AppointmentData" Width="100%" CssClass="schedule-date-
header-template" Height="650px" @bind-SelectedDate="@CurrentDate">
  <ScheduleTemplates>
    <DateHeaderTemplate>
      <div class="date-text">@(getDateHeaderText((context as
TemplateContext).Date)) </div>
      @ {
        @switch ((int)(context as TemplateContext).Date.DayOfWeek)
        {
          case 0:
            <div class="weather-text">25&#176;C</div>
            break;
          case 1:
            <div class="weather-text">18&#176;C</div>
            break;
          case 2:
            <div class="weather-text">10&#176;C</div>
            break;
          case 3:
            <div class="weather-text">16&#176;C</div>
            break;
          case 4:
            <div class="weather-text">8&#176;C</div>
            break;
          case 5:
            <div class="weather-text">27&#176;C</div>
            break;
          case 6:
            <div class="weather-text">17&#176;C</div>
            break;
        }
      }
    </DateHeaderTemplate>
  </ScheduleTemplates>
  <ScheduleViews>
    <ScheduleView Option="View.Day"></ScheduleView>
    <ScheduleView Option="View.Week"></ScheduleView>
    <ScheduleView Option="View.WorkWeek"></ScheduleView>
  </ScheduleViews>
</SfSchedule>

@code {
  DateTime CurrentDate = new DateTime(2020, 1, 10);
  public static string getDateHeaderText(DateTime date)
  {
    return date.ToString("dd ddd", CultureInfo.InvariantCulture);
  }
}

public class AppointmentData
```

```

{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
}
}

<style>
.schedule-date-header-template.e-schedule .e-vertical-view .e-header-cells {
padding: 0;
text-align: center !important;
}
.schedule-date-header-template.e-schedule .date-text {
font-size: 14px;
}
.schedule-date-header-template.e-schedule.e-device .date-text {
font-size: 12px;
}
.schedule-date-header-template.e-schedule .weather-image {
width: 20px;
height: 20px;
background-position: center center;
background-repeat: no-repeat;
background-size: cover;
}
.schedule-date-header-template.e-schedule .weather-text {
font-size: 11px;
}
}
</style>

```

Customization using OnRenderCell event

The date header can be customized by using `OnRenderCell` event. In the `OnRenderCell`, the argument `RenderCellEventArgs` returns the `ElementType` as `DateHeader` when the date header is rendering.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Width="100%" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEvents TValue="AppointmentData"
OnRenderCell="OnRenderCell"></ScheduleEvents>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>

```

```

<style>
.e-schedule .e-vertical-view .e-date-header-wrap table tbody td.e-header-
cells {
background-color: ivory;
}
</style>
@code{
private DateTime CurrentDate = new DateTime(2020, 3, 10);
public string[] CustomClass = { "custom-class" };
public void OnRenderCell(RenderCellEventArgs args)
{
//Here you can customize with your code
if (args.ElementType == ElementType.DateHeader)
{
args.CssClasses = new List<string>(CustomClass);
}
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 3, 10, 9, 30, 0) , EndTime = new DateTime(2020, 3, 10, 12, 0,
0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

TimelineYear header customization

The day header cells and month header cells can be customized in the TimelineYear view of the Scheduler using **DayHeaderTemplate** and **MonthHeaderTemplate**. The **DayHeaderTemplate** option is used to customize the day header cells of the TimelineYear view in both Vertical and Horizontal orientations. The **MonthHeaderTemplate** option is used to customize the month header cells of the TimelineYear view in both Vertical and Horizontal orientations.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
@using System.Globalization
<div>
<SfSchedule TValue="AppointmentData" Width="100%" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleTemplates>
<DayHeaderTemplate>
<div>@(getDayHeaderText((context as TemplateContext).Date))

```



```

</div>
</DayHeaderTemplate>
<MonthHeaderTemplate>
<div>@(getMonthHeaderText((context as TemplateContext).Date))</div>
</MonthHeaderTemplate>
</ScheduleTemplates>
<ScheduleViews>
<ScheduleView Option="View.TimelineYear" MaxEventsPerRow="2"
Orientation="Orientation.Vertical" DisplayName="Vertical Year">
</ScheduleView>
<ScheduleView Option="View.TimelineYear" MaxEventsPerRow="2"
Orientation="Orientation.Horizontal" DisplayName="Horizontal Year">
</ScheduleView>
</ScheduleViews>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
</SfScheduler>
</div>
@code{
private DateTime CurrentDate = new DateTime(2020, 3, 10);
public static string getDayHeaderText(DateTime date)
{
return date.ToString("dddd", CultureInfo.InvariantCulture);
}
public static string getMonthHeaderText(DateTime date)
{
return date.ToString("MMM", CultureInfo.InvariantCulture);
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 3, 4, 0, 0, 0) , EndTime = new DateTime(2020, 3, 5, 0, 0, 0)
},
new AppointmentData { Id = 2, Subject = "Conference", StartTime = new
DateTime(2020, 5, 1, 9, 30, 0) , EndTime = new DateTime(2020, 5, 1, 12, 0,
0) },
new AppointmentData { Id = 3, Subject = "Seminar", StartTime = new
DateTime(2020, 1, 2, 9, 30, 0) , EndTime = new DateTime(2020, 1, 2, 12, 0,
0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

Customizing header indent cells

It is possible to customize the header indent cells using the `HeaderIndentTemplate` option and change the look and appearance in both the vertical and timeline views. In vertical views, the header indent cells can be customized at the hierarchy level and the resource header left indent cell can be customized in timeline views using the template option.

Example: To customize the header left indent cell to display resources text, refer to the below code example.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleGroup Resources="@Resources"></ScheduleGroup>
<ScheduleResources>
<ScheduleResource TItem="ResourceData" TValue="int[]"
DataSource="@OwnersData" Field="OwnerId" Title="Owner" Name="Owners"
TextField="OwnerText" IdField="Id" GroupIDField="OwnerGroupId"
ColorField="OwnerColor" AllowMultiple="true"></ScheduleResource>
</ScheduleResources>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleTemplates>
<HeaderIndentTemplate>
<div class='e-resource-text'>
<div class="text">Resources</div>
</div>
</HeaderIndentTemplate>
</ScheduleTemplates>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.TimelineWeek"></ScheduleView>
<ScheduleView Option="View.TimelineMonth"></ScheduleView>
</ScheduleViews>
</SfSchedule>
<style>
.e-schedule .e-timeline-view .e-resource-left-td {
vertical-align: bottom;
}
.e-schedule .e-timeline-view .e-resource-left-td .e-resource-text,
.e-schedule .e-timeline-month-view .e-resource-left-td .e-resource-text {
font-weight: 500;
padding: 0;
}
.e-schedule .e-timeline-view .e-resource-left-td .e-resource-text > div {
border-right: 1px solid rgba(0, 0, 0, 0.12);
border-top: 1px solid rgba(0, 0, 0, 0.12);
flex: 0 0 33.3%;
font-weight: 500;
height: 36px;
line-height: 34px;
padding-left: 50px;
}
```

```

.e-schedule .e-timeline-month-view .e-resource-left-td .e-resource-text >
div {
border-right: 1px solid rgba(0, 0, 0, 0.12);
flex: 0 0 33.3%;
font-weight: 500;
height: 38px;
line-height: 34px;
padding-left: 50px;
}
.e-schedule .e-vertical-view .e-left-indent-wrap table tbody td.e-resource-
cells {
border-bottom-color: rgba(0, 0, 0, 0.12);
}
.e-schedule .e-vertical-view .e-left-indent-wrap table tbody td.e-resource-
cells .e-resource-text {
font-weight: 500;
}
.e-schedule .e-vertical-view .e-left-indent-wrap table tbody td.e-header-
cells .e-resource-text,
.e-schedule .e-vertical-view .e-left-indent-wrap table tbody td.e-all-day-
cells .e-resource-text {
display: none;
}
</style>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 31);
public string[] Resources { get; set; } = { "Rooms", "Owners" };
public List<ResourceData> RoomData { get; set; } = new List<ResourceData>
{
new ResourceData{ RoomText = "ROOM 1", Id = 1, RoomColor = "#cb6bb2" },
new ResourceData{ RoomText = "ROOM 2", Id = 2, RoomColor = "#56ca85" }
};
public List<ResourceData> OwnersData { get; set; } = new List<ResourceData>
{
new ResourceData{ OwnerText = "Nancy", Id = 1, OwnerGroupId = 1, OwnerColor
= "#ffaa00" },
new ResourceData{ OwnerText = "Steven", Id = 2, OwnerGroupId = 2, OwnerColor
= "#f8a398" },
new ResourceData{ OwnerText = "Michael", Id = 3, OwnerGroupId = 1,
OwnerColor = "#7499e1" }
};
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
0), OwnerId = 1, RoomId = 1 }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
}

```

```

public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
public int OwnerId { get; set; }
public int RoomId { get; set; }
}
public class ResourceData
{
public int Id { get; set; }
public string RoomText { get; set; }
public string RoomColor { get; set; }
public string OwnerText { get; set; }
public string OwnerColor { get; set; }
public int OwnerGroupId { get; set; }
}
}

```

Timescale Customization in Blazor Scheduler Component

The time slots are usually the time cells that are displayed on the Day, Week and Work Week views of both the calendar (to the left most position) and timeline views (at the top position). The `ScheduleTimeScale` allows to control and set the required time slot duration for the work cells displayed on Scheduler. It includes the following sub-options such as,

- **Enable** - When set to `true`, allows the Scheduler to display the appointments accurately against the exact time duration. If set to `false`, all the appointments of a day will be displayed one below the other with no grid lines displayed. Its default value is `true`.
- **Interval** – Defines the time duration on which the time axis to be displayed either in 1 hour or 30 minutes interval and so on. It accepts the values in minutes and defaults to 60.
- **SlotCount** – Decides the number of slot count to be split for the specified time interval duration. It defaults to 2, thus displaying two slots to represent an hour(each slot depicting 30 minutes duration).

Setting different time slot duration

The `Interval` and `SlotCount` properties can be used together on the Scheduler to set different time slot duration which is depicted in the following code example. Here, six time slots together represents an hour.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="650px">
  <ScheduleTimeScale Interval="60" SlotCount="6"></ScheduleTimeScale>
  <ScheduleViews>
    <ScheduleView Option="View.Day"></ScheduleView>
    <ScheduleView Option="View.Week"></ScheduleView>
    <ScheduleView Option="View.WorkWeek"></ScheduleView>
  </ScheduleViews>
</SfSchedule>
@code{
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
}
}

```

```

public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

Customizing time cells using template

The template option is available to allow customization of time slots which are as follows,

- **MajorSlotTemplate** - The template option to be applied for major time slots. Here, the template accepts `HTMLElement` as template design and then the parsed design is displayed onto the time cells. The time details can be accessed within this template.
- **MinorSlotTemplate** - The template option to be applied for minor time slots. Here, the template accepts `HTMLElement` as template design and then the parsed design is displayed onto the time cells. The time details can be accessed within this template.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
@using System.Globalization
<SfSchedule TValue="AppointmentData" Height="650px">
  <ScheduleTimeScale Interval="60" SlotCount="6">
    <MajorSlotTemplate>
      <div>@(majorSlotTemplate((context as TemplateContext).Date))</div>
    </MajorSlotTemplate>
    <MinorSlotTemplate>
      <div style="text-align: right; margin-right: 15px">@(minorSlotTemplate((context as TemplateContext).Date))</div>
    </MinorSlotTemplate>
  </ScheduleTimeScale>
  <ScheduleViews>
    <ScheduleView Option="View.Day"></ScheduleView>
    <ScheduleView Option="View.Week"></ScheduleView>
    <ScheduleView Option="View.WorkWeek"></ScheduleView>
  </ScheduleViews>
</SfSchedule>
@code{
public static string majorSlotTemplate(DateTime date)
{
  return date.ToString("hh", CultureInfo.InvariantCulture);
}
public static string minorSlotTemplate(DateTime date)
{
  return date.ToString("mm", CultureInfo.InvariantCulture);
}
public class AppointmentData
{
  public int Id { get; set; }
  public string Subject { get; set; }
}

```

```

public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

Hide the timescale

The grid lines which indicates the exact time duration can be enabled or disabled on the Scheduler, by setting `true` or `false` to the `Enable` option within the `ScheduleTimeScale` property. Its default value is `true`.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="650px">
  <ScheduleTimeScale Enable="false"></ScheduleTimeScale>
  <ScheduleViews>
    <ScheduleView Option="View.Week"></ScheduleView>
    <ScheduleView Option="View.TimelineWeek"
      MaxEventsPerRow="10"></ScheduleView>
  </ScheduleViews>
</SfSchedule>
@code{
public class AppointmentData
{
  public int Id { get; set; }
  public string Subject { get; set; }
  public string Location { get; set; }
  public DateTime StartTime { get; set; }
  public DateTime EndTime { get; set; }
  public string Description { get; set; }
  public bool IsAllDay { get; set; }
  public string RecurrenceRule { get; set; }
  public string RecurrenceException { get; set; }
  public Nullable<int> RecurrenceID { get; set; }
}
}

```

Highlighting current date and time

By default, Scheduler indicates current date with a highlighted date header on all views, as well as marks accurately the system's current time on specific views such as Day, Week, Work Week, Timeline Day, Timeline Week and Timeline Work Week views. To stop highlighting the current time indicator on Scheduler views, set `false` to the `ShowTimeIndicator` property which defaults to `true`.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="650px"
  ShowTimeIndicator="false">

```

```

<ScheduleViews>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.TimelineWeek"
MaxEventsPerRow="10"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

Working Days and Hours in Blazor Scheduler Component

The Scheduler can be customized on various aspects as well as it inherits almost all the calendar-specific features such as options,

- To set custom time range display on Scheduler
- To set different working hours
- To set different working days
- To set different first day of week
- To show/hide weekend days
- To show the week number

Set working days

By default, Scheduler considers the week days from Monday to Friday as **WorkDays** and therefore defaults to [1,2,3,4,5] - where 1 represents Monday, 2 represents Tuesday and so on. The days which are not defined in this working days collection are considered as non-working days. Therefore, when the weekend days are set to hide from Scheduler, all those non-working days too get hidden from the layout.

The Work week and Timeline Work week views displays exactly the defined working days on Scheduler layout, whereas other views displays all the days and simply differentiates the non-working days on UI with inactive cell color.

The working or business hours depiction on Scheduler are usually valid only on these specified working days.

The following example code depicts how to set the Scheduler to display Monday, Wednesday and Friday as working days of a week.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
```

```

<SfSchedule TValue="AppointmentData" Height="650px" ShowWeekend="false"
WorkDays="@WorkingDays">
<ScheduleViews>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.TimelineWeek"
MaxEventsPerRow="10"></ScheduleView>
<ScheduleView Option="View.Month" MaxEventsPerRow="2"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
public int[] WorkingDays { get; set; } = { 1, 3, 5 };
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

Hiding weekend days

The `ShowWeekend` property is used to either show or hide the weekend days of a week and it is not applicable on Work week view (as non-working days are usually not displayed on work week view). By default, it is set to `true`. The days which are not a part of the working days collection of a Scheduler are usually considered as non-working or weekend days.

Here, the working days are defined as [1, 3, 4, 5] on Scheduler and therefore the remaining days (0, 2, 6 – Sunday, Tuesday and Saturday) are considered as non-working or weekend days and will be hidden from all the views when `ShowWeekend` property is set to `false`.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="650px" ShowWeekend=false
WorkDays="@WorkingDays">
<ScheduleViews>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.TimelineWeek"
MaxEventsPerRow="10"></ScheduleView>
<ScheduleView Option="View.Month" MaxEventsPerRow="2"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
public int[] WorkingDays { get; set; } = { 1, 3, 5 };
public class AppointmentData
{
public int Id { get; set; }

```



```

public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

Show week numbers

It is possible to show the week number count of a week in the header bar of the Scheduler by setting true to the `ShowWeekNumber` property. By default, its default value is `false`. In Month view, the week numbers are displayed as a first column.

The `ShowWeekNumber` property is not applicable on Timeline views, as it has the equivalent [HeaderRows](#) property to handle such requirement with additional customization.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="650px" ShowWeekNumber=true>
  <ScheduleViews>
    <ScheduleView Option="View.Day"></ScheduleView>
    <ScheduleView Option="View.Week"></ScheduleView>
    <ScheduleView Option="View.Month" MaxEventsPerRow="2"></ScheduleView>
  </ScheduleViews>
</SfSchedule>
@code{
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
}
}

```

Different options in showing week numbers

By default, week numbers are shown in the Scheduler based on the first day of the year. However, the week numbers can be determined based on the following criteria by setting the `WeekRule` property with `CalendarWeekRule` enumeration.

FirstDay – The first week of the year is calculated based on the first day of the year.

FirstFourDayWeek – The first week of the year begins from the first week with four or more days.

FirstFullWeek – The first week of the year begins when meeting the first day of the week (firstDayOfWeek) and the first day of the year.

For more details refer to [this link](#)

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="650px" @bind-
SelectedDate="@CurrentDate" ShowWeekNumber=true
WeekRule="System.Globalization.CalendarWeekRule.FirstFourDayWeek">
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.Month" MaxEventsPerRow="2"></ScheduleView>
</ScheduleViews>
</SfSchedule>

@code{
DateTime CurrentDate = new DateTime(2020, 12, 28);
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}
```

Set working hours

Working hours indicates the work hour limit within the Scheduler, which is visually highlighted with an active color on work cells. The working hours can be set on Scheduler using the **ScheduleWorkhours** which includes the following sub-options,

- **Highlight** – enables/disables the highlighting of work hours.
- **Start** - sets the start time of the working/business hour of a day.
- **End** - sets the end time limit of the working/business hour of a day.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="650px">
<ScheduleWorkHours Highlight="true" Start="11:00"
End="20:00"></ScheduleWorkHours>
<ScheduleViews>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.TimelineWeek"
MaxEventsPerRow="10"></ScheduleView>
</ScheduleViews>
```

```

</SfSchedule>
@code{
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

Scheduler displaying custom hours

It is possible to display the event Scheduler layout with specific time durations by hiding the unwanted hours. To do so, set the start and end hour for the Scheduler using the **StartHour** and **EndHour** properties respectively.

The following code example displays the Scheduler starting from the time range 7.00 AM to 6.00 PM and the remaining hours are hidden on the UI.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="650px" StartHour="07:00"
EndHour="18:00">
<ScheduleViews>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.TimelineWeek"
MaxEventsPerRow="10"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

Setting start day of the week

By default, Scheduler defaults to **Sunday** as its first day of a week. To change the Scheduler's start day of a week with different day, set the **FirstDayOfWeek** property with the values ranging from 0 to 6.

Here, Sunday is always denoted as 0, Monday as 1 and so on.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="650px" FirstDayOfWeek=3>
<ScheduleViews>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.Month" MaxEventsPerRow="2"></ScheduleView>
<ScheduleView Option="View.TimelineWeek"
MaxEventsPerRow="10"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}
```

Scroll to specific time and date

You can manually scroll to a specific time on Scheduler by making use of the **ScrollToAsync** method as depicted in the following code example.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Calendars
<div>
<span>Scroll To</span>
<SfTimePicker TValue="DateTime?" Width="100px" Format="HH:mm" @bind-
Value="TimeValue">
<TimePickerEvents TValue="DateTime?"
ValueChange="OnValueChange"></TimePickerEvents>
</SfTimePicker>
</div>
<SfSchedule TValue="AppointmentData" @ref="ScheduleRef" Width="100%"
Height="550px" @bind-SelectedDate="@CurrentDate">
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
</ScheduleViews>
```

```

<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
</SfSchedule>
@code{
    DateTime CurrentDate { get; set; } = new DateTime(2020, 1, 31);
    SfSchedule<AppointmentData> ScheduleRef;
    public DateTime? TimeValue { get; set; } = new DateTime(DateTime.Today.Year,
    DateTime.Today.Month, DateTime.Today.Day, 9, 0, 0);
    public async Task OnValueChanged(ChangeEventArgs<DateTime?> args)
    {
        await ScheduleRef.ScrollToAsync(args.Text);
    }
    List<AppointmentData> DataSource = new List<AppointmentData>
    {
        new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
        DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
        0) }
    };
    public class AppointmentData
    {
        public int Id { get; set; }
        public string Subject { get; set; }
        public string Location { get; set; }
        public DateTime StartTime { get; set; }
        public DateTime EndTime { get; set; }
        public string Description { get; set; }
        public bool IsAllDay { get; set; }
        public string RecurrenceRule { get; set; }
        public string RecurrenceException { get; set; }
        public Nullable<int> RecurrenceID { get; set; }
    }
}

```

See Also

- [To display the current time indicator](#)
- [To set different working hours for each resources](#)
- [To set different working days for each resources](#)

Cell Customizations in Blazor Scheduler Component

The cells of the Scheduler can be easily customized using the cell template.

Setting cell dimensions in Vertical Views

The height and width of the Scheduler cells can be customized either to increase or reduce its size through the `CssClass` property, which overrides the default CSS applied on cells of vertical views.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" CssClass="schedule-cell-dimension"
Height="550px">
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>

```

```

<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
<style>
.schedule-cell-dimension.e-schedule .e-vertical-view .e-date-header-wrap
table col,
.schedule-cell-dimension.e-schedule .e-vertical-view .e-content-wrap table
col {
width: 200px;
}
.schedule-cell-dimension.e-schedule .e-vertical-view .e-time-cells-wrap
table td,
.schedule-cell-dimension.e-schedule .e-vertical-view .e-work-cells {
height: 100px;
}
.schedule-cell-dimension.e-schedule .e-month-view .e-work-cells,
.schedule-cell-dimension.e-schedule .e-month-view .e-date-header-wrap table
col {
width: 200px;
}
.schedule-cell-dimension.e-schedule .e-month-view .e-work-cells {
height: 200px;
}
</style>
@code{
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

Setting cell dimensions in Timeline Views

The height and width of the Scheduler cells can be customized either to increase or reduce its size through the `CssClass` property, which overrides the default CSS applied on cells of timeline views.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" CssClass="schedule-cell-dimension"
Height="550px">
<ScheduleViews>
<ScheduleView Option="View.TimelineWeek"
MaxEventsPerRow="10"></ScheduleView>
<ScheduleView Option="View.TimelineMonth"
MaxEventsPerRow="10"></ScheduleView>
</ScheduleViews>

```

```

</SfSchedule>
<style>
.schedule-cell-dimension.e-schedule .e-timeline-view .e-date-header-wrap
table col,
.schedule-cell-dimension.e-schedule .e-timeline-view .e-content-wrap table
col {
width: 200px;
}
.schedule-cell-dimension.e-schedule .e-timeline-month-view .e-date-header-
wrap table col,
.schedule-cell-dimension.e-schedule .e-timeline-month-view .e-content-wrap
table col {
width: 200px;
}
</style>
@code{
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

Customizing cells using CellTemplate

The **CellTemplate** is used to customize the cell background with specific images or appropriate text on the given date values.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Width="100%" Height="650px" @bind-
SelectedDate="@CurrentDate">
<ScheduleTemplates>
<CellTemplate>
<div class="templatewrap">
@{
@if ((int)(context as TemplateContext).Date.Month == 11 && (int)(context as
TemplateContext).Date.Day == 23)
{
<div class="caption">Thanksgiving day</div>
}
@if ((int)(context as TemplateContext).Date.Month == 12 && (int)(context as
TemplateContext).Date.Day == 9)
{
<div class="caption">Party time</div>
}
@if ((int)(context as TemplateContext).Date.Month == 12 && (int)(context as
TemplateContext).Date.Day == 13)

```

```

{
<div class="caption">Party time</div>
}
@if ((int)(context as TemplateContext).Date.Month == 12 && (int)(context as
TemplateContext).Date.Day == 22)
{
<div class="caption">Happy birthday</div>
}
@if ((int)(context as TemplateContext).Date.Month == 12 && (int)(context as
TemplateContext).Date.Day == 24)
{
<div class="caption">Christmas Eve</div>
}
@if ((int)(context as TemplateContext).Date.Month == 12 && (int)(context as
TemplateContext).Date.Day == 25)
{
<div class="caption">Christmas day</div>
}
@if ((int)(context as TemplateContext).Date.Month == 1 && (int)(context as
TemplateContext).Date.Day == 1)
{
<div class="caption">New Year"s Day</div>
}
@if ((int)(context as TemplateContext).Date.Month == 1 && (int)(context as
TemplateContext).Date.Day == 14)
{
<div class="caption">Get together</div>
}
}
</div>
</CellTemplate>
</ScheduleTemplates>
<ScheduleViews>
<ScheduleView Option="View.Month" MaxEventsPerRow="2"></ScheduleView>
</ScheduleViews>
</SfSchedule>
<style>
.e-schedule .e-month-view .e-work-cells {
position: relative;
}
.e-schedule .templatewrap {
text-align: center;
position: absolute;
width: 100%;
}
.e-schedule .caption {
overflow: hidden;
text-overflow: ellipsis;
vertical-align: middle;
}
</style>
@code {
DateTime CurrentDate = new DateTime(2020, 1, 15);
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
}
}

```



```

public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

Customizing cells using OnRenderCell event

The cells can also be customized by using `OnRenderCell` event. In the `OnRenderCell`, the argument `RenderCellEventArgs` returns the `ElementType` as `WorkCells` and `AllDayCells` when the cell is rendering.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Width="100%" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEvents TValue="AppointmentData"
OnRenderCell="OnRenderCell"></ScheduleEvents>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
<style>
.e-schedule .e-vertical-view .e-work-hours.custom-class {
background-color: ivory;
}
</style>
@code{
private DateTime CurrentDate = new DateTime(2020, 3, 10);
public string[] CustomClass = { "custom-class" };
public void OnRenderCell(RenderCellEventArgs args)
{
//Here you can customize with your code
if (args.ElementType == ElementType.WorkCells)
{
args.CssClasses = new List<string>(CustomClass); //The default work hours
color is changed to ivory color
}
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 3, 10, 9, 30, 0) , EndTime = new DateTime(2020, 3, 10, 12, 0,
0) }
};
public class AppointmentData

```

```
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}
```

Customizing the minimum and maximum date values

Providing the `MinDate` and `MaxDate` property with some date values, allows the Scheduler to set the minimum and maximum date range. The Scheduler date that lies beyond this minimum and maximum date range will be in a disabled state so that the date navigation will be blocked beyond the specified date range.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<p>Setting date</p>
<SfSchedule TValue="AppointmentData" Height="650px" MinDate="new
DateTime(2019, 1, 1)" MaxDate="new DateTime(2030, 12, 31)" @bind-
SelectedDate="@CurrentDate">
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
private DateTime CurrentDate = new DateTime(2020, 1, 10);
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
}
}
```

By default, the `MinDate` property value is set to `new DateTime(1900, 1, 1)` and `MaxDate` property value is set to `new DateTime(2099, 12, 31)`. The user can also set the customized `MinDate` and `MaxDate` property values.

State Persistence in Blazor Scheduler Component

State persistence allowed Scheduler to retain the `CurrentView`, `SelectedDate` and Scroll position values in the `localStorage` for state maintenance even if the browser is refreshed or if you move to the next

page within the browser. This action is handled through the [EnablePersistence](#) property which is set to false by default. When it is set to true, `CurrentView`, `SelectedDate` and Scroll position values of the scheduler component will be retained even after refreshing the page.

Scheduler ID is essential to set state persistence.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="650px" ID="Schedule"
EnablePersistence="true" @bind-SelectedDate="@SelectedDate" @bind-
CurrentView="@CurrentView">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>

@code {
public DateTime SelectedDate = new DateTime(2020, 1, 9);
public View CurrentView = View.Week;
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 6, 9, 30, 0) , EndTime = new DateTime(2020, 1, 6, 11, 0,
0),
RecurrenceRule = "FREQ=DAILY;INTERVAL=1;COUNT=5", RecurrenceException =
"20200107T043000Z,20200109T043000Z" }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public string Description { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public Nullable<bool> IsAllDay { get; set; }
public string CategoryColor { get; set; }
public bool IsReadOnly { get; set; }
public string RecurrenceRule { get; set; }
public Nullable<int> RecurrenceID { get; set; }
public string RecurrenceException { get; set; }
public string StartTimezone { get; set; }
public string EndTimezone { get; set; }
}
}
```

Exporting in Blazor Scheduler Component

The Scheduler supports exporting all its appointments both to an Excel or ICS extension file. It offers different methods to export its appointments in an Excel or iCal format file. Let's look onto the ways on how to implement the exporting functionality in Scheduler.

Excel Exporting

The Scheduler allows to export all its events into an Excel format file by using the `ExportToExcelAsync` method. By default, it exports all the default fields of Scheduler mapped through `<ScheduleEventSettings>` property.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Buttons
<SfButton Content="Excel Export" OnClick="OnExportToExcel"></SfButton>
<SfSchedule @ref="ScheduleRef" TValue="AppointmentData" Height="650px"
@bind-SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Week"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 10);
SfSchedule<AppointmentData> ScheduleRef;
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Explosion of Betelgeuse Star",
Location = "Dallas", StartTime = new DateTime(2020, 1, 8, 9, 30, 0),
EndTime = new DateTime(2020, 1, 8, 11, 0, 0) },
new AppointmentData { Id = 2, Subject = "Thule Air Crash Report", Location =
"Texas", StartTime = new DateTime(2020, 1, 9, 12, 0, 0), EndTime = new
DateTime(2020, 1, 9, 14, 0, 0) },
new AppointmentData { Id = 3, Subject = "Blue Moon Eclipse", Location =
"Australia", StartTime = new DateTime(2020, 1, 10, 10, 30, 0), EndTime = new
DateTime(2020, 1, 10, 11, 0, 0) },
new AppointmentData { Id = 4, Subject = "Meteor Showers in 2020", Location =
"Canada", StartTime = new DateTime(2020, 1, 11, 13, 0, 0), EndTime = new
DateTime(2020, 1, 11, 14, 30, 0) },
new AppointmentData { Id = 5, Subject = "Milky Way as Melting pot", Location
= "Mexico", StartTime = new DateTime(2020, 1, 12, 12, 0, 0), EndTime = new
DateTime(2020, 1, 12, 14, 0, 0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
public async Task OnExportToExcel()
{
await ScheduleRef.ExportToExcelAsync();
}
}
```

	A	B	C	D	E	F	G	H	I
	Id	Subject	StartTime	EndTime	StartTimezone	EndTimezone	Location	Description	
2	1	Explosion of Betelgeuse Sta	1-8-2020 9:30 AM	1-8-2020 11:00 AM			Dallas		
3	2	Thule Air Crash Report	1-9-2020 12:00 PM	1-9-2020 2:00 PM			Texas		
4	3	Blue Moon Eclipse	1-10-2020 10:30 AM	1-10-2020 11:00 AM			Australia		
5	4	Meteor Showers in 2020	1-11-2020 1:00 PM	1-11-2020 2:30 PM			Canada		
6	5	Milky Way as Melting pot	1-12-2020 12:00 PM	1-12-2020 2:00 PM			Mexico		
7									

Exporting with custom fields

By default, Scheduler exports all the default event fields that are mapped to it through the `<ScheduleEventSettings>` property. To limit the number of fields on the exported excel file, it provides an option to export only the custom fields of the event data. To export such custom fields alone, define the required `Fields` and pass it as argument to the `ExportToExcelAsync` method as shown in the following example. In the following code example, only 'Id', 'Subject', 'StartTime', 'EndTime' fields were exported.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Buttons
<SfButton Content="Excel Export" OnClick="OnExportToExcel"></SfButton>
<SfSchedule @ref="ScheduleRef" TValue="AppointmentData" Height="650px"
@bind-SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Week"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 10);
SfSchedule<AppointmentData> ScheduleRef;
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Explosion of Betelgeuse Star",
Location = "Dallas", StartTime = new DateTime(2020, 1, 8, 9, 30, 0),
EndTime = new DateTime(2020, 1, 8, 11, 0, 0) },
new AppointmentData { Id = 2, Subject = "Thule Air Crash Report", Location =
"Texas", StartTime = new DateTime(2020, 1, 9, 12, 0, 0), EndTime = new
DateTime(2020, 1, 9, 14, 0, 0) },
new AppointmentData { Id = 3, Subject = "Blue Moon Eclipse", Location =
"Australia", StartTime = new DateTime(2020, 1, 10, 10, 30, 0), EndTime = new
DateTime(2020, 1, 10, 11, 0, 0) },
new AppointmentData { Id = 4, Subject = "Meteor Showers in 2020", Location =
"Canada", StartTime = new DateTime(2020, 1, 11, 13, 0, 0), EndTime = new
DateTime(2020, 1, 11, 14, 30, 0) },
new AppointmentData { Id = 5, Subject = "Milky Way as Melting pot", Location
= "Mexico", StartTime = new DateTime(2020, 1, 12, 12, 0, 0), EndTime = new
DateTime(2020, 1, 12, 14, 0, 0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
}
```

```

}
public async Task OnExportToExcel()
{
    ExportOptions Options = new ExportOptions() { ExportType = ExcelFormat.Xlsx,
    Fields = new string[] { "Id", "Subject", "StartTime", "EndTime" } };
    await ScheduleRef.ExportToExcelAsync(Options);
}
}

```

	A	B	C	D	E
1	Id	Subject	StartTime	EndTime	Location
2	1	Explosion of Betelgeuse Sta	1-8-2020 9:30 AM	1-8-2020 11:00 AM	Dallas
3	2	Thule Air Crash Report	1-9-2020 12:00 PM	1-9-2020 2:00 PM	Texas
4	3	Blue Moon Eclipse	1-10-2020 10:30 AM	1-10-2020 11:00 AM	Australia
5	4	Meteor Showers in 2020	1-11-2020 1:00 PM	1-11-2020 2:30 PM	Canada
6	5	Milky Way as Melting pot	1-12-2020 12:00 PM	1-12-2020 2:00 PM	Mexico
7					

Exporting individual occurrences of a recurring series

By default, the Scheduler exports recurring events as a single data by exporting only its parent record into the excel file. If you want to export each individual occurrences of a recurring series appointment as separate records in an Excel file, define the `IncludeOccurrences` option as `true` and pass it as argument to the `ExportToExcelAsync` method. By default, the `IncludeOccurrences` option is set to `false`.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Buttons
<SfButton Content="Excel Export" OnClick="OnExportToExcel"></SfButton>
<SfSchedule @ref="ScheduleRef" TValue="AppointmentData" Height="650px"
@bind-SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Week"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
    DateTime CurrentDate = new DateTime(2020, 1, 10);
    SfSchedule<AppointmentData> ScheduleRef;
    List<AppointmentData> DataSource = new List<AppointmentData>
    {
        new AppointmentData { Id = 1, Subject = "Explosion of Betelgeuse Star",
        Location = "Dallas", StartTime = new DateTime(2020, 1, 8, 9, 30, 0),
        EndTime = new DateTime(2020, 1, 8, 11, 0, 0) },
        new AppointmentData { Id = 2, Subject = "Thule Air Crash Report", Location =
        "Texas", StartTime = new DateTime(2020, 1, 9, 12, 0, 0), EndTime = new
        DateTime(2020, 1, 9, 14, 0, 0) },
        new AppointmentData { Id = 3, Subject = "Blue Moon Eclipse", Location =
        "Australia", StartTime = new DateTime(2020, 1, 10, 10, 30, 0), EndTime = new
        DateTime(2020, 1, 10, 11, 0, 0) },
        new AppointmentData { Id = 4, Subject = "Meteor Showers in 2020", Location =
        "Canada", StartTime = new DateTime(2020, 1, 11, 13, 0, 0), EndTime = new
        DateTime(2020, 1, 11, 14, 30, 0) },
    }
}

```

```

new AppointmentData { Id = 5, Subject = "Milky Way as Melting pot", Location
= "Mexico", StartTime = new DateTime(2020, 1, 12, 12, 0, 0), EndTime = new
DateTime(2020, 1, 12, 14, 0, 0)  }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
}
public ExportOptions ExportValues = new ExportOptions { IncludeOccurrences =
true };
public async Task OnExportToExcel()
{
await ScheduleRef.ExportToExcelAsync(ExportValues);
}
}

```

Exporting custom event data

By default, the whole event collection bound to the Scheduler gets exported as an excel file. To export only specific events of Scheduler or some custom event collection, you need to pass those custom data collection as a parameter to the `ExportToExcelAsync` method as shown in this following example, through the `CustomData` option.

By default, the event data are taken from Scheduler dataSource.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Buttons
<SfButton Content="Excel Export" OnClick="OnExportToExcel"></SfButton>
<SfSchedule @ref="ScheduleRef" TValue="AppointmentData" Width="100%"
Height="650px" @bind-SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Week"></ScheduleView>
</ScheduleViews>
</SfSchedule>s
@code{
DateTime CurrentDate = new DateTime(2020, 1, 10);
SfSchedule<AppointmentData> ScheduleRef;
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Explosion of Betelgeuse Star",
Location = "Dallas", StartTime = new DateTime(2020, 1, 8, 9, 30, 0),
EndTime = new DateTime(2020, 1, 8, 11, 0, 0) },
new AppointmentData { Id = 2, Subject = "Thule Air Crash Report", Location =
"Texas", StartTime = new DateTime(2020, 1, 9, 12, 0, 0), EndTime = new
DateTime(2020, 1, 9, 14, 0, 0) },
new AppointmentData { Id = 3, Subject = "Blue Moon Eclipse", Location =
"Australia", StartTime = new DateTime(2020, 1, 10, 10, 30, 0), EndTime = new
DateTime(2020, 1, 10, 11, 0, 0) },

```

```

new AppointmentData { Id = 4, Subject = "Meteor Showers in 2020", Location =
"Canada", StartTime = new DateTime(2020, 1, 11, 13, 0, 0), EndTime = new
DateTime(2020, 1, 11, 14, 30, 0) },
new AppointmentData { Id = 5, Subject = "Milky Way as Melting pot", Location
= "Mexico", StartTime = new DateTime(2020, 1, 12, 12, 0, 0), EndTime = new
DateTime(2020, 1, 12, 14, 0, 0) }
};
public ExportOptions ExportValues = new ExportOptions { CustomData =
customData };
static List<AppointmentData> customData = new List<AppointmentData>()
{
new AppointmentData
{
Id = 1,
Subject = "Explosion of Betelgeuse Star",
Location = "Space Centre USA",
StartTime = new DateTime(2020, 1, 31, 9, 30, 0) ,
EndTime = new DateTime(2020, 1, 31, 11, 0, 0)
},
new AppointmentData
{
Id = 2,
Subject = "Thule Air Crash Report",
Location = "Newyork City",
StartTime = new DateTime(2020, 1, 31, 12, 0, 0),
EndTime = new DateTime(2020, 1, 31, 11, 0, 0)
}
};
public async Task OnExportToExcel()
{
await ScheduleRef.ExportToExcelAsync(ExportValues);
}
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

Customizing the column header texts with custom fields exporting

You can change the field names of appointment in the column header when exporting using the **FieldsInfo** option through the **ExportFieldInfo** class and pass it as an argument to the **ExportToExcelAsync** method as shown in the following code example.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Buttons

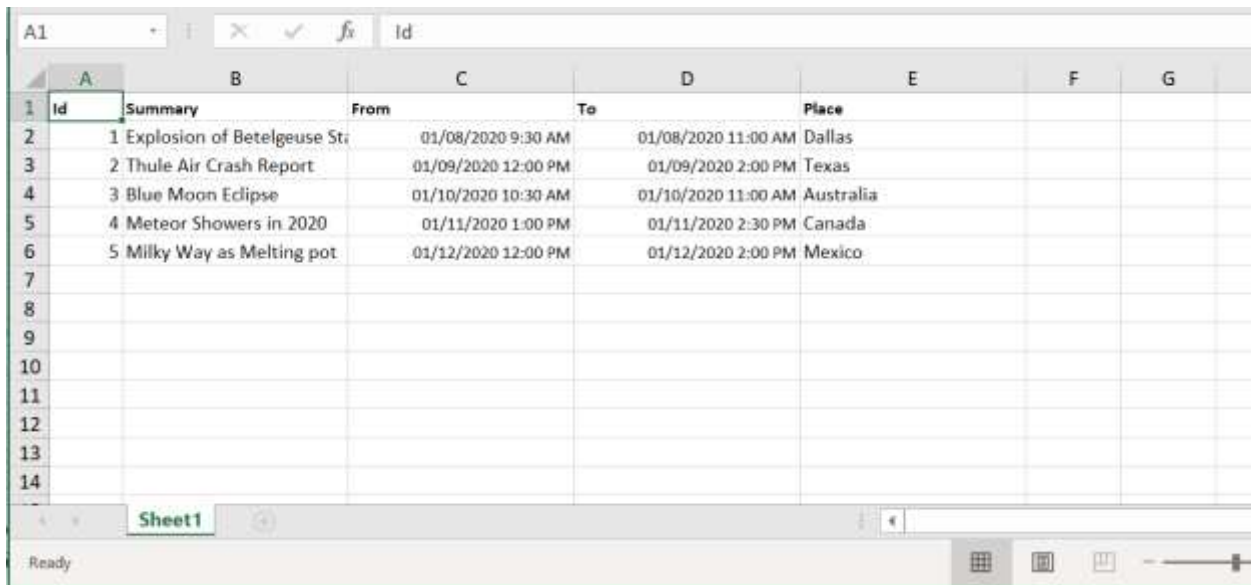
```



```

<SfButton Content="Excel Export" OnClick="OnExportToExcel"></SfButton>
<SfSchedule @ref="ScheduleRef" TValue="AppointmentData" Height="650px"
@bind-SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Week"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 10);
SfSchedule<AppointmentData> ScheduleRef;
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Explosion of Betelgeuse Star",
Location = "Dallas", StartTime = new DateTime(2020, 1, 8, 9, 30, 0),
EndTime = new DateTime(2020, 1, 8, 11, 0, 0) },
new AppointmentData { Id = 2, Subject = "Thule Air Crash Report", Location =
"Texas", StartTime = new DateTime(2020, 1, 9, 12, 0, 0), EndTime = new
DateTime(2020, 1, 9, 14, 0, 0) },
new AppointmentData { Id = 3, Subject = "Blue Moon Eclipse", Location =
"Australia", StartTime = new DateTime(2020, 1, 10, 10, 30, 0), EndTime = new
DateTime(2020, 1, 10, 11, 0, 0) },
new AppointmentData { Id = 4, Subject = "Meteor Showers in 2020", Location =
"Canada", StartTime = new DateTime(2020, 1, 11, 13, 0, 0), EndTime = new
DateTime(2020, 1, 11, 14, 30, 0) },
new AppointmentData { Id = 5, Subject = "Milky Way as Melting pot", Location
= "Mexico", StartTime = new DateTime(2020, 1, 12, 12, 0, 0), EndTime = new
DateTime(2020, 1, 12, 14, 0, 0) }
};
public async Task OnExportToExcel()
{
List<ExportFieldInfo> exportFields = new List<ExportFieldInfo>();
exportFields.Add(new ExportFieldInfo { Name = "Id", Text = "Id" });
exportFields.Add(new ExportFieldInfo { Name = "Subject", Text = "Summary"
});
exportFields.Add(new ExportFieldInfo { Name = "StartTime", Text = "From" });
exportFields.Add(new ExportFieldInfo { Name = "EndTime", Text = "To" });
exportFields.Add(new ExportFieldInfo { Name = "Location", Text = "Place" });
ExportOptions options = new ExportOptions() { ExportType = ExcelFormat.Xlsx,
FieldsInfo = exportFields };
await ScheduleRef.ExportToExcelAsync(options);
}
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```



	A	B	C	D	E	F	G
1	Id	Summary	From	To	Place		
2	1	Explosion of Betelgeuse Star	01/08/2020 9:30 AM	01/08/2020 11:00 AM	Dallas		
3	2	Thule Air Crash Report	01/09/2020 12:00 PM	01/09/2020 2:00 PM	Texas		
4	3	Blue Moon Eclipse	01/10/2020 10:30 AM	01/10/2020 11:00 AM	Australia		
5	4	Meteor Showers in 2020	01/11/2020 1:00 PM	01/11/2020 2:30 PM	Canada		
6	5	Milky Way as Melting pot	01/12/2020 12:00 PM	01/12/2020 2:00 PM	Mexico		
7							
8							
9							
10							
11							
12							
13							
14							

Export with custom file name

By default, the Scheduler allows you to download the exported Excel file with a name `Schedule.xlsx`. It also provides an option to export the excel file with a custom file name, define the desired `FileName` and passing it as an argument to the `ExportToExcelAsync` method.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Buttons
<SfButton Content="Excel Export" OnClick="OnExportToExcel"></SfButton>
<SfSchedule @ref="ScheduleRef" TValue="AppointmentData" Height="650px"
@bind-SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Week"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 10);
SfSchedule<AppointmentData> ScheduleRef;
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Explosion of Betelgeuse Star",
Location = "Dallas", StartTime = new DateTime(2020, 1, 8, 9, 30, 0),
EndTime = new DateTime(2020, 1, 8, 11, 0, 0) },
new AppointmentData { Id = 2, Subject = "Thule Air Crash Report", Location =
"Texas", StartTime = new DateTime(2020, 1, 9, 12, 0, 0), EndTime = new
DateTime(2020, 1, 9, 14, 0, 0) },
new AppointmentData { Id = 3, Subject = "Blue Moon Eclipse", Location =
"Australia", StartTime = new DateTime(2020, 1, 10, 10, 30, 0), EndTime = new
DateTime(2020, 1, 10, 11, 0, 0) },
new AppointmentData { Id = 4, Subject = "Meteor Showers in 2020", Location =
"Canada", StartTime = new DateTime(2020, 1, 11, 13, 0, 0), EndTime = new
DateTime(2020, 1, 11, 14, 30, 0) },
}
```

```

new AppointmentData { Id = 5, Subject = "Milky Way as Melting pot", Location
= "Mexico", StartTime = new DateTime(2020, 1, 12, 12, 0, 0), EndTime = new
DateTime(2020, 1, 12, 14, 0, 0)  }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
public ExportOptions ExportValues = new ExportOptions { FileName =
"SchedulerData" };
public async Task OnExportToExcel()
{
await ScheduleRef.ExportToExcelAsync(ExportValues);
}
}

```

Excel file formats

By default, the Scheduler exports event data to an excel file in the **.xlsx** format. You can also export the Scheduler data in either of the file type such as **.xlsx** or **csv** formats, by defining the **ExportType** option as either **csv** or **xlsx**. By default, the **ExportType** is set to **xlsx**.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Buttons
<SfButton Content="Excel Export" OnClick="OnExportToExcel"></SfButton>
<SfSchedule @ref="ScheduleRef" TValue="AppointmentData" Height="650px"
@bind-SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Week"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 10);
SfSchedule<AppointmentData> ScheduleRef;
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Explosion of Betelgeuse Star",
Location = "Dallas", StartTime = new DateTime(2020, 1, 8, 9, 30, 0),
EndTime = new DateTime(2020, 1, 8, 11, 0, 0) },
new AppointmentData { Id = 2, Subject = "Thule Air Crash Report", Location =
"Texas", StartTime = new DateTime(2020, 1, 9, 12, 0, 0), EndTime = new
DateTime(2020, 1, 9, 14, 0, 0) },
new AppointmentData { Id = 3, Subject = "Blue Moon Eclipse", Location =
"Australia", StartTime = new DateTime(2020, 1, 10, 10, 30, 0), EndTime = new
DateTime(2020, 1, 10, 11, 0, 0) },

```

```

new AppointmentData { Id = 4, Subject = "Meteor Showers in 2020", Location =
"Canada", StartTime = new DateTime(2020, 1, 11, 13, 0, 0), EndTime = new
DateTime(2020, 1, 11, 14, 30, 0) },
new AppointmentData { Id = 5, Subject = "Milky Way as Melting pot", Location =
"Mexico", StartTime = new DateTime(2020, 1, 12, 12, 0, 0), EndTime = new
DateTime(2020, 1, 12, 14, 0, 0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
public ExportOptions ExportValues = new ExportOptions { ExportType =
ExcelFormat.Csv };
public async Task OnExportToExcel()
{
await ScheduleRef.ExportToExcelAsync(ExportValues);
}
}

```

Export with specific date format

You can export the Scheduler data with specific date format, by defining the **DateFormat** option which accepts the MSDN date format in string type. In the following code example, the scheduler appointments are exported in 24 hour date format.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Buttons
<SfButton Content="Excel Export" OnClick="OnExportToExcel"></SfButton>
<SfSchedule @ref="ScheduleRef" TValue="AppointmentData" Height="650px"
@bind-SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Week"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 10);
SfSchedule<AppointmentData> ScheduleRef;
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Explosion of Betelgeuse Star",
Location = "Dallas", StartTime = new DateTime(2020, 1, 8, 9, 30, 0),
EndTime = new DateTime(2020, 1, 8, 11, 0, 0) },
new AppointmentData { Id = 2, Subject = "Thule Air Crash Report", Location =
"Texas", StartTime = new DateTime(2020, 1, 9, 12, 0, 0), EndTime = new
DateTime(2020, 1, 9, 14, 0, 0) },
}
}

```

```

new AppointmentData { Id = 3, Subject = "Blue Moon Eclipse", Location =
"Australia", StartTime = new DateTime(2020, 1, 10, 10, 30, 0), EndTime = new
DateTime(2020, 1, 10, 11, 0, 0) },
new AppointmentData { Id = 4, Subject = "Meteor Showers in 2020", Location =
"Canada", StartTime = new DateTime(2020, 1, 11, 13, 0, 0), EndTime = new
DateTime(2020, 1, 11, 14, 30, 0) },
new AppointmentData { Id = 5, Subject = "Milky Way as Melting pot", Location =
"Mexico", StartTime = new DateTime(2020, 1, 12, 12, 0, 0), EndTime = new
DateTime(2020, 1, 12, 14, 0, 0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
public ExportOptions ExportValues = new ExportOptions { DateFormat =
"MM/dd/yy H:mm:ss" };
public async Task OnExportToExcel()
{
await ScheduleRef.ExportToExcelAsync(ExportValues);
}
}

```

	A	B	C	D	E	F	G	
1	Id	Subject	Location	Next	descriptio	StartTime	EndTime	IsAllDay
2	1	Explosion of Betelgeuse Star	Dallas			1/8/2020 9:30	1/8/2020 11:00	
3	2	Thule Air Crash Report	Texas			1/9/2020 12:00	1/9/2020 14:00	
4	3	Blue Moon Eclipse	Australia			1/10/2020 10:30	1/10/2020 11:00	
5	4	Meteor Showers in 2020	Canada			1/11/2020 13:00	1/11/2020 14:30	
6	5	Milky Way as Melting pot	Mexico			1/12/2020 12:00	1/12/2020 14:00	
7								

Exporting calendar events as ICS file

You can export the Scheduler events to a calendar (.ics) file format, and open it on any of the other default calendars such as Google or Outlook.

The following code example shows how the Scheduler events are exported to a calendar (.ics) file by making use of the `ExportToCalendarAsync` public method.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Buttons
<SfButton Content="Excel Export" OnClick="OnExportToIcs"></SfButton>
<SfSchedule @ref="ScheduleRef" TValue="AppointmentData" Height="650px"
@bind-SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>

```

```

<ScheduleView Option="View.Week"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
    DateTime CurrentDate = new DateTime(2020, 1, 10);
    SfSchedule<AppointmentData> ScheduleRef;
    List<AppointmentData> DataSource = new List<AppointmentData>
    {
        new AppointmentData { Id = 1, Subject = "Explosion of Betelgeuse Star",
            Location = "Dallas", StartTime = new DateTime(2020, 1, 8, 9, 30, 0),
            EndTime = new DateTime(2020, 1, 8, 11, 0, 0) },
        new AppointmentData { Id = 2, Subject = "Thule Air Crash Report", Location =
            "Texas", StartTime = new DateTime(2020, 1, 9, 12, 0, 0), EndTime = new
            DateTime(2020, 1, 9, 14, 0, 0) },
        new AppointmentData { Id = 3, Subject = "Blue Moon Eclipse", Location =
            "Australia", StartTime = new DateTime(2020, 1, 10, 10, 30, 0), EndTime = new
            DateTime(2020, 1, 10, 11, 0, 0) },
        new AppointmentData { Id = 4, Subject = "Meteor Showers in 2020", Location =
            "Canada", StartTime = new DateTime(2020, 1, 11, 13, 0, 0), EndTime = new
            DateTime(2020, 1, 11, 14, 30, 0) },
        new AppointmentData { Id = 5, Subject = "Milky Way as Melting pot", Location
            = "Mexico", StartTime = new DateTime(2020, 1, 12, 12, 0, 0), EndTime = new
            DateTime(2020, 1, 12, 14, 0, 0) }
    };
    public class AppointmentData
    {
        public int Id { get; set; }
        public string Subject { get; set; }
        public string Location { get; set; }
        public DateTime StartTime { get; set; }
        public DateTime EndTime { get; set; }
        public string Description { get; set; }
        public bool IsAllDay { get; set; }
        public string RecurrenceRule { get; set; }
        public string RecurrenceException { get; set; }
        public Nullable<int> RecurrenceID { get; set; }
    }
    public async Task OnExportToIcs()
    {
        await ScheduleRef.ExportToICalendarAsync();
    }
}

```

Exporting calendar with custom file name

By default, the calendar is exported with a file name **Calendar.ics**. To change this file name on export, pass the custom string value as **FileName** to the method argument so as to get the file downloaded with this provided name.

The following example downloads the iCal file with a name **ScheduleEvents.ics**.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Buttons
<SfButton Content="Excel Export" OnClick="OnExportToIcs"></SfButton>

```

```

<SfSchedule @ref="ScheduleRef" TValue="AppointmentData" Height="650px"
@bind-SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Week"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 10);
SfSchedule<AppointmentData> ScheduleRef;
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Explosion of Betelgeuse Star",
Location = "Dallas", StartTime = new DateTime(2020, 1, 8, 9, 30, 0),
EndTime = new DateTime(2020, 1, 8, 11, 0, 0) },
new AppointmentData { Id = 2, Subject = "Thule Air Crash Report", Location =
"Texas", StartTime = new DateTime(2020, 1, 9, 12, 0, 0), EndTime = new
DateTime(2020, 1, 9, 14, 0, 0) },
new AppointmentData { Id = 3, Subject = "Blue Moon Eclipse", Location =
"Australia", StartTime = new DateTime(2020, 1, 10, 10, 30, 0), EndTime = new
DateTime(2020, 1, 10, 11, 0, 0) },
new AppointmentData { Id = 4, Subject = "Meteor Showers in 2020", Location =
"Canada", StartTime = new DateTime(2020, 1, 11, 13, 0, 0), EndTime = new
DateTime(2020, 1, 11, 14, 30, 0) },
new AppointmentData { Id = 5, Subject = "Milky Way as Melting pot", Location =
"Mexico", StartTime = new DateTime(2020, 1, 12, 12, 0, 0), EndTime = new
DateTime(2020, 1, 12, 14, 0, 0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
public async Task OnExportToIcs()
{
await ScheduleRef.ExportToICalendarAsync("ScheduleEvents");
}
}

```

Importing events from other calendars

The events from external calendars (ICS files) can be imported into Scheduler by using the `ImportICalendarAsync` method. In the following code example events has been imported from an ICS file into Scheduler with the help of Uploader. In `ImportICalendarAsync` public method, ics file is passed as string format.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Inputs
@using System.IO
<SfUploader AllowedExtensions=".ics" CssClass="calendar-import"
Multiple="false">
<UploaderButtons Browse="Choose File"></UploaderButtons>
<UploaderEvents ValueChange="OnChange"></UploaderEvents>
</SfUploader>
<SfSchedule @ref="ScheduleRef" TValue="AppointmentData" Width="100%"
Height="650px" @bind-SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
SfSchedule<AppointmentData> ScheduleRef;
DateTime CurrentDate = new DateTime(2020, 1, 10);
public async Task OnChange(UploadChangeEventArgs args)
{
foreach (var file in args.Files)
{
file.Stream.Position = 0;
StreamReader reader = new StreamReader(file.Stream);
await ScheduleRef.ImportICalendarAsync(reader.ReadToEnd());
}
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Explosion of Betelgeuse Star",
Location = "Dallas", StartTime = new DateTime(2020, 3, 10, 9, 30, 0),
EndTime = new DateTime(2020, 3, 10, 11, 0, 0) },
new AppointmentData { Id = 2, Subject = "Thule Air Crash Report", Location =
"Texas", StartTime = new DateTime(2020, 3, 13, 12, 0, 0), EndTime = new
DateTime(2020, 3, 13, 14, 0, 0) },
new AppointmentData { Id = 3, Subject = "Blue Moon Eclipse", Location =
"Australia", StartTime = new DateTime(2020, 3, 11, 10, 30, 0), EndTime = new
DateTime(2020, 3, 11, 13, 0, 0) },
new AppointmentData { Id = 4, Subject = "Meteor Showers in 2020", Location =
"Canada", StartTime = new DateTime(2020, 3, 9, 13, 0, 0), EndTime = new
DateTime(2020, 3, 9, 14, 30, 0) },
new AppointmentData { Id = 5, Subject = "Milky Way as Melting pot", Location
= "Mexico", StartTime = new DateTime(2020, 3, 12, 9, 0, 0), EndTime = new
DateTime(2020, 3, 12, 10, 30, 0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
}

```



```

public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}
<style>
.calendar-import.e-upload {
border: 0;
}
.calendar-import.e-upload .e-file-select-wrap {
padding: 0
}
.calendar-import.e-upload .e-file-select-wrap .e-file-drop, .calendar-import
.e-upload-files {
display: none;
}
</style>

```

How to print the Scheduler element

The Scheduler allows to print the Scheduler element by using the `PrintAsync` method. The Print method works in two ways.

- Using Print method without options.
- Using a Print method with options.

Using PrintAsync method without options

You can print the Schedule element with the current view by using the `PrintAsync` method without passing the `PrintOptions` options. The following example shows how to print the Scheduler using the `PrintAsync` method without passing the `PrintOptions` options.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Buttons
<SfButton OnClick="OnPrintClick">Print</SfButton>
<SfSchedule @ref="ScheduleRef" TValue="AppointmentData" Width="100%"
Height="650px" @bind-SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
SfSchedule<AppointmentData> ScheduleRef;
DateTime CurrentDate = new DateTime(2020, 1, 10);
public async void OnPrintClick()
{
await ScheduleRef.PrintAsync();
}
}

```

```

List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData { Id = 1, Subject = "Explosion of Betelgeuse Star",
        Location = "Dallas", StartTime = new DateTime(2020, 3, 10, 9, 30, 0),
        EndTime = new DateTime(2020, 3, 10, 11, 0, 0) },
    new AppointmentData { Id = 2, Subject = "Thule Air Crash Report", Location =
        "Texas", StartTime = new DateTime(2020, 3, 13, 12, 0, 0), EndTime = new
        DateTime(2020, 3, 13, 14, 0, 0) },
    new AppointmentData { Id = 3, Subject = "Blue Moon Eclipse", Location =
        "Australia", StartTime = new DateTime(2020, 3, 11, 10, 30, 0), EndTime = new
        DateTime(2020, 3, 11, 13, 0, 0) },
    new AppointmentData { Id = 4, Subject = "Meteor Showers in 2020", Location =
        "Canada", StartTime = new DateTime(2020, 3, 9, 13, 0, 0), EndTime = new
        DateTime(2020, 3, 9, 14, 30, 0) },
    new AppointmentData { Id = 5, Subject = "Milky Way as Melting pot", Location
        = "Mexico", StartTime = new DateTime(2020, 3, 12, 9, 0, 0), EndTime = new
        DateTime(2020, 3, 12, 10, 30, 0) }
};
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
}

```

Using a PrintAsync method with options

You can print the Schedule element with customized Width and Height using the `PrintAsync` method by passing the `PrintOptions` Height and Width options. The following example shows how to print the Scheduler using the `PrintAsync` method by passing the `PrintOptions` options.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Buttons
<SfButton OnClick="OnPrintClick">Print</SfButton>
<SfSchedule @ref="ScheduleRef" TValue="AppointmentData" Width="100%"
Height="650px" @bind-SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
SfSchedule<AppointmentData> ScheduleRef;

```

```

DateTime CurrentDate = new DateTime(2020, 1, 10);
PrintOptions Options = new PrintOptions() { Height = "auto", Width = "auto"
};
public async void OnPrintClick()
{
    await ScheduleRef.PrintAsync(Options);
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData { Id = 1, Subject = "Explosion of Betelgeuse Star",
        Location = "Dallas", StartTime = new DateTime(2020, 3, 10, 9, 30, 0),
        EndTime = new DateTime(2020, 3, 10, 11, 0, 0) },
    new AppointmentData { Id = 2, Subject = "Thule Air Crash Report", Location =
        "Texas", StartTime = new DateTime(2020, 3, 13, 12, 0, 0), EndTime = new
        DateTime(2020, 3, 13, 14, 0, 0) },
    new AppointmentData { Id = 3, Subject = "Blue Moon Eclipse", Location =
        "Australia", StartTime = new DateTime(2020, 3, 11, 10, 30, 0), EndTime = new
        DateTime(2020, 3, 11, 13, 0, 0) },
    new AppointmentData { Id = 4, Subject = "Meteor Showers in 2020", Location =
        "Canada", StartTime = new DateTime(2020, 3, 9, 13, 0, 0), EndTime = new
        DateTime(2020, 3, 9, 14, 30, 0) },
    new AppointmentData { Id = 5, Subject = "Milky Way as Melting pot", Location
        = "Mexico", StartTime = new DateTime(2020, 3, 12, 9, 0, 0), EndTime = new
        DateTime(2020, 3, 12, 10, 30, 0) }
};
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
}
}

```

Scheduler Dimensions in Blazor Scheduler Component

The Scheduler dimensions refers to both height and width of the entire layout and it accepts 3 types of values.

- auto
- pixel
- percentage

Auto Height and Width

When height and width of the Scheduler are set to **auto**, it will try hard to keep an element the same width as its parent container. In other words, for the parent container that holds Scheduler, it's width or height will be the sum of its children. By default, Scheduler is assigned with **auto** values for both height and width properties.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="auto" Width="auto">
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

Height and Width in pixel

The Scheduler height and width will be rendered exactly as per the given pixel values. It accepts both string and number values.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="650px" Width="550px">
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
}
}

```

```
public Nullable<int> RecurrenceID { get; set; }
}
```

Height and Width in percentage

When height and width of the Scheduler are given as percentage, it will make the Scheduler as wide as the parent container.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="100%" Width="100%">
  <ScheduleViews>
    <ScheduleView Option="View.Day"></ScheduleView>
    <ScheduleView Option="View.Week"></ScheduleView>
    <ScheduleView Option="View.WorkWeek"></ScheduleView>
    <ScheduleView Option="View.Month"></ScheduleView>
    <ScheduleView Option="View.Agenda"></ScheduleView>
  </ScheduleViews>
</SfSchedule>
@code{
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}
```

Globalization in Blazor Scheduler Component

The Scheduler integrates different date-time formats and cultures, which allows it to function globally, thus meeting the diverse needs of different regions.

You can adapt the Scheduler to various languages by parsing and formatting the date or number **Internationalization**, adding culture specific customization and translation to the text **Localization**.

Blazor Server Side

The static local texts in the Scheduler component can be changed to other culture by referring the Resource file. You can refer more details about localization [here](#). By default, Scheduler is set to follow the English culture ('en-US'). The following steps explain how to render the Scheduler in German culture ('de-DE').

- Open the **Startup.cs** file and add the below configuration in the **ConfigureServices** function as follows.

CSHARP

```

using Syncfusion.Blazor;
using System.Globalization;
using Microsoft.AspNetCore.Localization;
using SchedulerLocalization.Shared;
namespace SchedulerLocalization
{
    public class Startup
    {
        ....
        ....
        public void ConfigureServices(IServiceCollection services)
        {
            ....
            ....
            services.AddSyncfusionBlazor();
            services.AddLocalization(options => options.ResourcesPath = "Resources");
            services.Configure<RequestLocalizationOptions>(options =>
            {
                // define the list of cultures your app will support
                var supportedCultures = new List<CultureInfo>()
                {
                    new CultureInfo("de-DE")
                };
                // set the default culture
                options.DefaultRequestCulture = new RequestCulture("de-DE");
                options.SupportedCultures = supportedCultures;
                options.SupportedUICultures = supportedCultures;
                options.RequestCultureProviders = new List<IRequestCultureProvider>() {
                    new QueryStringRequestCultureProvider() // Here, You can also use other
                    localization provider
                };
            });
            services.AddSingleton(typeof(ISyncfusionStringLocalizer),
                typeof(SampleLocalizer));
        }
    }
}

```

Add [UseRequestLocalization\(\)](#) middle-ware in Configure method in **Startup.cs** file to get browser Culture Information.

- Then, write a **class** by inheriting **ISyncfusionStringLocalizer** interface and override the **ResourceManager** property to get the resource file details from the application end.

CSHARP

```

using Syncfusion.Blazor;
namespace SchedulerLocalization
{
    public class SampleLocalizer : ISyncfusionStringLocalizer
    {
        public string GetText(string key)
        {
            return this.ManageResourceManagerr.GetString(key);
        }
    }
}

```

```
public System.Resources.ResourceManager ResourceManager
{
    get
    {
        return
        SchedulerLocalization.Resources.SyncfusionBlazorLocale.ResourceManager;
    }
}
}
```

- Add **.resx** file to [Resource](#) folder and that file contains the key value pair of locale content in the following format.

ASPX-CS

```
<Component_Name> <Feature_Name> <Locale_Key>
```

- Finally, add the Scheduler component in razor page as in the following code example.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="650px" SelectedDate="@ (new
DateTime(2020, 2, 14))">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
</SfSchedule>
@code{
List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData { Id = 1, Subject = "Paris", StartTime = new
DateTime(2020, 2, 13, 10, 0, 0) , EndTime = new DateTime(2020, 2, 13, 12, 0,
0) },
    new AppointmentData { Id = 2, Subject = "Germany", StartTime = new
DateTime(2020, 2, 15, 10, 0, 0) , EndTime = new DateTime(2020, 2, 15, 12, 0,
0) }
};
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
}
}
```

Blazor Web Assembly

You can refer [here] (<https://blazor.syncfusion.com/documentation/common/localization/#enable-localization-in-blazor-webassembly-application>) to enable the localization in Blazor Web Assembly.

Setting date format

Scheduler can be used with all valid date formats and by default it follows the universal date format "MM/dd/yyyy". If the `DateFormat` property is not specified particularly, then it will work based on the system's local culture. As the system's local culture is "en-US", this makes it to follow the "MM/dd/yyyy" pattern.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="650px" DateFormat="yyyy/MM/dd">
  <ScheduleViews>
    <ScheduleView Option="View.Day"></ScheduleView>
    <ScheduleView Option="View.Week"></ScheduleView>
    <ScheduleView Option="View.WorkWeek"></ScheduleView>
    <ScheduleView Option="View.Month"></ScheduleView>
    <ScheduleView Option="View.Agenda"></ScheduleView>
  </ScheduleViews>
</SfSchedule>
@code{
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
}
}
```

Time mode

The time mode of the Scheduler can be either 12 or 24 hours format which is completely based on the system's local culture and also the Scheduler supported to customize the time mode using the `TimeFormat` property.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="650px" TimeFormat="T">
  <ScheduleViews>
    <ScheduleView Option="View.Day"></ScheduleView>
    <ScheduleView Option="View.Week"></ScheduleView>
    <ScheduleView Option="View.WorkWeek"></ScheduleView>
    <ScheduleView Option="View.Month"></ScheduleView>
    <ScheduleView Option="View.Agenda"></ScheduleView>
  </ScheduleViews>
</SfSchedule>
@code{
public class AppointmentData
{
    public int Id { get; set; }
}
```



```

public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

Displaying Scheduler in RTL mode

The Scheduler layout and its behavior can be changed as per the common RTL (Right to Left) conventions by setting `EnableRtl` to `true`. By doing so, the Scheduler will display its usual layout from right to left. Its default value is `false`.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="650px" EnableRtl="true">
  <ScheduleViews>
    <ScheduleView Option="View.Day"></ScheduleView>
    <ScheduleView Option="View.Week"></ScheduleView>
    <ScheduleView Option="View.WorkWeek"></ScheduleView>
    <ScheduleView Option="View.Month"></ScheduleView>
    <ScheduleView Option="View.Agenda"></ScheduleView>
  </ScheduleViews>
</SfSchedule>
@code {
  public class AppointmentData
  {
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
  }
}

```

See Also

- [How to change first day of the week in the Scheduler](#)

Accessibility in Blazor Scheduler Component

Accessibility is achieved in the Scheduler component through WAI-ARIA standard and keyboard navigation. The Scheduler 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.

The following ARIA attributes are used in the Scheduler:

Property	Functionalities
role	It gives assistive technologies about how to handle each element in a widget.
aria-disabled	It indicates the disabled state of the Scheduler.
aria-selected	It indicates the currently selected cell of the Scheduler.
aria-live	It indicates a string value that labels the Scheduler element.
aria-label	It indicates the disabled state of the Scheduler and its items.
aria-labelledby	It indicates editor dialog title to the user through assistive technologies.
aria-describedby	It indicates editor dialog content description to the user through assistive technologies.

Keyboard navigation

All the Scheduler actions can be controlled via keyboard keys and is available by using `AllowKeyboardInteraction` property which is set to true by default. The applicable key combinations and its relative functionalities are listed below.

Interaction Keys | Description

Alt + j | Focuses the Scheduler [Provided from application end].

Tab | Focuses the first or active item on the scheduler header bar and then move the focus to the next available event elements. If no events present, then focus moves out of the component.

Shift + Tab | Reverse focusing of the Tab functionality. Inverse focusing of event elements from the last one and then move onto the first or active item on Scheduler header bar and then moves out of the component.

Enter | Opens the quick popup on the selected cells or events.

Escape | Closes any of the popup that are in open state.

Arrow | To move onto the next available cells in either of the needed directions (left, right, top and right)

Shift + Arrow | For multiple cell selection on either direction.

Delete | Deletes one or more selected events.

Ctrl + Click on events | To select multiple events.

Alt + Number (from 1 to 6) | To switch between the views on Scheduler.

Ctrl + Left Arrow | To navigate to the previous date period.

Ctrl + Right Arrow | To navigate to the next date period.

Left or Right Arrow | On pressing any of these keys when focus is currently on the Scheduler header bar, moves the focus to the previous or next items in the header bar.

Space or Enter | It activates any of the focused items.

Page Up & Page Down | To scroll through the work cells area.

Home | To move the selection to the first cell of Scheduler.

WebAssembly Performance in Blazor Scheduler Component

This section provides performance guidelines for using Syncfusion Scheduler component efficiently in Blazor WebAssembly application. The best practice or guidelines for general framework Blazor WebAssembly performance can be found [here](#).

You can refer to our Getting Started with [Blazor Server-Side Scheduler](#) and [Blazor WebAssembly Scheduler](#) documentation pages for configuration specifications.

Avoid unnecessary component renders

During Blazor Diffing Algorithm, every views of the Scheduler component and its child component will be checked for re-rendering. For instance, having **EventCallback** on the application or Scheduler will check every child component, once event callback is completed.

You can have fine-grained control over Scheduler component rendering. **PreventRender** method helps to avoid unnecessary re-rendering of the Scheduler component. This method internally overrides the **ShouldRender** method of the Scheduler to prevent rendering.

In the following example:

- **PreventRender** method is called in the **IncrementCount** method which is a click callback.
- Now, Scheduler component will not be a part of the rendering which happens as result of the click event and **currentCount** alone will get updated.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<p>Current count: @currentCount</p>
<button class="btn btn-primary" @onclick="IncrementCount">Click me</button>
<SfSchedule @ref="ScheduleRef" TValue=AppointmentData>
  <ScheduleViews>
    <ScheduleView Option="View.Day"></ScheduleView>
    <ScheduleView Option="View.Week"></ScheduleView>
    <ScheduleView Option="View.WorkWeek"></ScheduleView>
    <ScheduleView Option="View.Month"></ScheduleView>
    <ScheduleView Option="View.Agenda"></ScheduleView>
  </ScheduleViews>
</SfSchedule>
@code {
  SfSchedule<AppointmentData> ScheduleRef;
  private int currentCount = 0;
  private void IncrementCount()
  {
    ScheduleRef.PreventRender();
  }
}
```

```

currentCount++;
}
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
}
}

```

Notes

- **PreventRender** method accepts boolean argument that accepts true or false to disable or enable rendering respectively.
- **PreventRender** method can be used only after Scheduler component completed initial rendering. Calling this method during initial rendering will not have any effect.

Avoid unnecessary component renders after Scheduler events

When a callback method is assigned to the Scheduler events, then the **StateHasChanged** will be called in parent component of the Scheduler automatically once the event is completed.

You can prevent this re-rendering of the Scheduler component by calling the **PreventRender** method.

In the following example:

- **OnCellClick** event is bound with a callback method, so once cell click event is completed the **StateHasChanged** will be invoked for the parent component.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<p style="color:green; font-size:20px">@Status</p>
<SfSchedule @ref="ScheduleRef" TValue="AppointmentData" Width="100%"
Height="550px" @bind-SelectedDate="@CurrentDate">
    <ScheduleEvents TValue="AppointmentData" OnCellClick="OnCellClick"
ActionCompleted="OnActionCompleted"></ScheduleEvents>
    <ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
    <ScheduleViews>
        <ScheduleView Option="View.Day"></ScheduleView>
        <ScheduleView Option="View.Week"></ScheduleView>
        <ScheduleView Option="View.WorkWeek"></ScheduleView>
        <ScheduleView Option="View.Month"></ScheduleView>
        <ScheduleView Option="View.Agenda"></ScheduleView>
    </ScheduleViews>
</SfSchedule>
@code{
    SfSchedule<AppointmentData> ScheduleRef;
}

```

```

string Status = string.Empty;
DateTime CurrentDate = new DateTime(2020, 3, 10);
public void OnCellClick(CellClickEventArgs args)
{
    ScheduleRef.PreventRender();
}
public void OnActionCompleted(ActionEventArgs<AppointmentData> args)
{
    ScheduleRef.PreventRender();
    if (args.ActionType == ActionType.EventCreate || args.ActionType ==
        ActionType.EventChange)
    {
        Status = "Success";    //Status become success on create and update of an
        event.
    }
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
    DateTime(2020, 3, 10, 9, 30, 0) , EndTime = new DateTime(2020, 3, 10, 12, 0,
    0) }
};
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
}
}

```

Notes

- **PreventRender** method internally overrides the **ShouldRender** method of the Scheduler to prevent rendering.
- It is recommended to use **PreventRender** method for user interactive events such as OnCellClick, OnEventClick etc. for better performance.
- For events without any argument such as **DataBound**, you can use **PreventRender** method of the Scheduler to disable rendering.

Events in Blazor Scheduler Component

In this section, the list of events of the Scheduler component have been provided which will be triggered for appropriate Scheduler actions.

The events should be provided to the Scheduler using **ScheduleEvents** tag. When using events of Scheduler, **TValue** must be provided in the **ScheduleEvents** tag.

ActionCompleted

ActionCompleted event triggers on successful completion of the Scheduler actions.

The action type that can be checked within the **ActionCompleted** event are as follows.

ActionType	Description
----- -----	
EventCreate	Triggers once event is created.
EventChange	Triggers once event is updated.
EventRemove	Triggers once event is deleted.
DateNavigate	Triggers once date navigation is performed.
ViewNavigate	Triggers once view navigation is performed.
ResourceExpand	Triggers once resource is expanded in timeline views.
ResourceCollapse	Triggers once resource is collapsed in timeline views.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<p style="color:green; font-size:20px">@Status</p>
<SfSchedule TValue="AppointmentData" Width="100%" Height="550px" @bind-
SelectedDate="@CurrentDate">
  <ScheduleEvents TValue="AppointmentData"
  ActionCompleted="OnActionCompleted"></ScheduleEvents>
  <ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
  <ScheduleViews>
    <ScheduleView Option="View.Day"></ScheduleView>
    <ScheduleView Option="View.Week"></ScheduleView>
    <ScheduleView Option="View.WorkWeek"></ScheduleView>
    <ScheduleView Option="View.Month"></ScheduleView>
    <ScheduleView Option="View.Agenda"></ScheduleView>
  </ScheduleViews>
</SfSchedule>
@code{
  DateTime CurrentDate = new DateTime(2020, 3, 10);
  string Status = "";
  public void OnActionCompleted(ActionEventArgs<AppointmentData> args)
  {
    if (args.ActionType == ActionType.EventCreate || args.ActionType ==
    ActionType.EventChange)
    {
      Status = "Success";    //Status become success on create and update of an
      event.
    }
  }
  List<AppointmentData> DataSource = new List<AppointmentData>
  {
    new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
    DateTime(2020, 3, 10, 9, 30, 0) , EndTime = new DateTime(2020, 3, 10, 12, 0,
    0) }
  };
  public class AppointmentData

```

```
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}
```

Created

Created event triggers after the Scheduler component is created.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Width="100%" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEvents TValue="AppointmentData"
Created="OnCreated"></ScheduleEvents>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 3, 10);
public void OnCreated()
{
//Here you can customize your code
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 3, 10, 9, 30, 0) , EndTime = new DateTime(2020, 3, 10, 12, 0,
0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
}
```

```
public Nullable<int> RecurrenceID { get; set; }
}
}
```

DataBinding

DataBinding event triggers before the data binds to the Scheduler.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Width="100%" Height="550px" @bind-
SelectedDate="@CurrentDate">
  <ScheduleEvents TValue="AppointmentData"
  DataBinding="DataBindHandler"></ScheduleEvents>
  <ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
  <ScheduleViews>
    <ScheduleView Option="View.Day"></ScheduleView>
    <ScheduleView Option="View.Week"></ScheduleView>
    <ScheduleView Option="View.WorkWeek"></ScheduleView>
    <ScheduleView Option="View.Month"></ScheduleView>
    <ScheduleView Option="View.Agenda"></ScheduleView>
  </ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 3, 10);
public void DataBindHandler(DataBindingEventArgs<AppointmentData> args)
{
//Triggers before the data binds to the scheduler, while performing CRUD
actions, View and Date navigations
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 3, 10, 9, 30, 0) , EndTime = new DateTime(2020, 3, 10, 12, 0,
0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}
```

DataBound

DataBound event triggers once the event data is bound to the Scheduler.

ASPX-CS


```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" @ref="@ScheduleRef" Width="100%"
Height="550px" @bind-SelectedDate="@CurrentDate">
<ScheduleEvents TValue="AppointmentData"
DataBound="OnDataBound"></ScheduleEvents>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 3, 10);
SfSchedule<AppointmentData> ScheduleRef;
public async Task OnDataBound(DataBoundEventArgs<AppointmentData> args)
{
List<AppointmentData> eventCollection = await ScheduleRef.GetEvents();
//You can get the entire appointment collections in the EventCollection
variable
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 3, 10, 9, 30, 0) , EndTime = new DateTime(2020, 3, 10, 12, 0,
0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

Destroyed

Destroyed event triggers when the Scheduler component is destroyed.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Width="100%" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEvents TValue="AppointmentData"
Destroyed="OnDestroyed"></ScheduleEvents>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>

```

```

<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 3, 10);
public void OnDestroyed()
{
//Here you can customize your code
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 3, 10, 9, 30, 0) , EndTime = new DateTime(2020, 3, 10, 12, 0,
0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

Dragged

Dragged event triggers when the dragging of appointment is stopped.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" @ref="@ScheduleObj" Width="100%"
Height="550px" @bind-SelectedDate="@CurrentDate">
<ScheduleEvents TValue="AppointmentData"
Dragged="OnDragged"></ScheduleEvents>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 3, 10);
SfSchedule<AppointmentData> ScheduleObj;

```

```

public async Task OnDragged(DragEventArgs<AppointmentData> args)
{
    await ScheduleObj.OpenEditorAsync(args.Data, CurrentAction.Save);    //To
    open the editor window at drag stop
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
    DateTime(2020, 3, 10, 9, 30, 0) , EndTime = new DateTime(2020, 3, 10, 12, 0,
    0) }
};
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
}
}

```

EventRendered

EventRendered event triggers before each of the event getting rendered on the Scheduler user interface.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Width="100%" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEvents TValue="AppointmentData"
EventRendered="OnEventRendered"></ScheduleEvents>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
<style>
.e-schedule .e-vertical-view .e-all-day-appointment-wrapper .e-
appointment.custom-class,
.e-schedule .e-vertical-view .e-day-wrapper .e-appointment.custom-class,
.e-schedule .e-month-view .e-appointment.custom-class {
background: yellow;
color: red;
}
</style>
@code{

```

```

private DateTime CurrentDate = new DateTime(2020, 3, 10);
public string[] CustomClass = { "custom-class" };
public void OnEventRendered(EventRenderedArgs<AppointmentData> args)
{
    args.CssClasses = new List<string>(CustomClass);
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
    DateTime(2020, 3, 10, 9, 30, 0) , EndTime = new DateTime(2020, 3, 10, 12, 0,
    0) }
};
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
}
}

```

MoreEventsClicked

MoreEventsClicked event triggers when the more events indicator

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Width="100%" Height="550px" @bind-
SelectedDate="@CurrentDate" @bind-CurrentView="@CurrentView">
<ScheduleEvents TValue="AppointmentData"
MoreEventsClicked="OnMoreEventsClick"></ScheduleEvents>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 3, 10);
View CurrentView = View.Month;
public void OnMoreEventsClick(MoreEventsClickArgs args)
{
    args.Cancel = true;    //To prevent showing the appointments in more
    indiactor
}
List<AppointmentData> DataSource = new List<AppointmentData>
{

```

```

new AppointmentData { Id = 1, Subject = "Meeting In", StartTime = new
DateTime(2020, 3, 10, 9, 30, 0) , EndTime = new DateTime(2020, 3, 10, 12, 0,
0) },
new AppointmentData { Id = 2, Subject = "Conference", StartTime = new
DateTime(2020, 3, 11, 9, 30, 0) , EndTime = new DateTime(2020, 3, 11, 12, 0,
0) },
new AppointmentData { Id = 3, Subject = "Meeting Out", StartTime = new
DateTime(2020, 3, 11, 9, 30, 0) , EndTime = new DateTime(2020, 3, 11, 12, 0,
0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

Navigating

Navigating event triggers before the date or view navigation takes place on Scheduler.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Width="100%" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEvents TValue="AppointmentData"
Navigating="OnNavigating"></ScheduleEvents>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 3, 10);
public void OnNavigating(NavigatingEventArgs args)
{
if (args.Action == "date") {
args.Cancel = true;    //To prevent date navigation
}
}
List<AppointmentData> DataSource = new List<AppointmentData>
{

```

```

new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 3, 10, 9, 30, 0) , EndTime = new DateTime(2020, 3, 10, 12, 0,
0) }
};
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
}
}

```

OnActionBegin

OnActionBegin event triggers on beginning of every Scheduler action.

The request type that can be checked within the **OnActionBegin** event are as follows.

ActionType	Description
----- -----	
EventCreate	Triggers on Creating new event.
EventChange	Triggers on updating an event.
EventRemove	Triggers on Deleting an event.
DateNavigate	Triggers while performing date navigations.
ViewNavigate	Triggers while performing view navigations.
ResourceExpand	Triggers while expanding resource on timeline views.
ViewNavigate	Triggers while collapsing resource on timeline views.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Width="100%" Height="550px" @bind-
SelectedDate="@CurrentDate">
    <ScheduleEvents TValue="AppointmentData"
    OnActionBegin="OnActionBegin"></ScheduleEvents>
    <ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
    <ScheduleViews>
        <ScheduleView Option="View.Day"></ScheduleView>
        <ScheduleView Option="View.Week"></ScheduleView>
        <ScheduleView Option="View.WorkWeek"></ScheduleView>
        <ScheduleView Option="View.Month"></ScheduleView>
        <ScheduleView Option="View.Agenda"></ScheduleView>
    </ScheduleViews>
</SfSchedule>

```

```
@code{
DateTime CurrentDate = new DateTime(2020, 3, 10);
public void OnActionBegin(ActionEventArgs<AppointmentData> args)
{
    if (args.ActionType == ActionType.EventRemove)    //To check for request type
        is event delete
    {
        args.Cancel = true;    //To prevent the appointment deletion
    }
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
    DateTime(2020, 3, 10, 9, 30, 0) , EndTime = new DateTime(2020, 3, 10, 12, 0,
    0) }
};
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
}
}
```

OnActionFailure

OnActionFailure event triggers when a Scheduler action gets failed or interrupted.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Width="100%" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEvents TValue="AppointmentData"
OnActionFailure="OnActionFailure"></ScheduleEvents>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 3, 10);
public void OnActionFailure(ActionEventArgs<AppointmentData> args)
{
    //args.Error catches the failure details.
}
```

```

List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
    DateTime(2020, 3, 10, 9, 30, 0) , EndTime = new DateTime(2020, 3, 10, 12, 0,
    0) }
};
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
}
}

```

OnClick

OnClick event triggers when the Scheduler cells are single clicked or on single tap on the same cells in mobile devices.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" @ref="@ScheduleRef" Width="100%"
Height="550px" @bind-SelectedDate="@CurrentDate">
<ScheduleEvents TValue="AppointmentData"
OnClick="OnClick"></ScheduleEvents>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
    DateTime CurrentDate = new DateTime(2020, 3, 10);
    SfSchedule<AppointmentData> ScheduleRef;
    public async Task OnClick(CellClickEventArgs args)
    {
        args.Cancel = true;
        await ScheduleRef.OpenEditorAsync(args, CurrentAction.Add); //To open
        editor window on cell click
    }
    List<AppointmentData> DataSource = new List<AppointmentData>
    {
        new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
        DateTime(2020, 3, 10, 9, 30, 0) , EndTime = new DateTime(2020, 3, 10, 12, 0,
        0) }
    };
}

```



```

public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
}

```

OnCellDoubleClick

OnCellDoubleClick event triggers when the Scheduler cells are double clicked.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Width="100%" Height="550px" @bind-
SelectedDate="@CurrentDate">
    <ScheduleEvents TValue="AppointmentData"
OnCellDoubleClick="OnCellDoubleClick"></ScheduleEvents>
    <ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
    <ScheduleViews>
        <ScheduleView Option="View.Day"></ScheduleView>
        <ScheduleView Option="View.Week"></ScheduleView>
        <ScheduleView Option="View.WorkWeek"></ScheduleView>
        <ScheduleView Option="View.Month"></ScheduleView>
        <ScheduleView Option="View.Agenda"></ScheduleView>
    </ScheduleViews>
</SfSchedule>
@code{
    DateTime CurrentDate = new DateTime(2020, 3, 10);
    public void OnCellDoubleClick(CellClickEventArgs args)
    {
        args.Cancel = true;    //To prevent the opening of editor window on cells
        alone.
    }
    List<AppointmentData> DataSource = new List<AppointmentData>
    {
        new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
        DateTime(2020, 3, 10, 9, 30, 0) , EndTime = new DateTime(2020, 3, 10, 12, 0,
        0) }
    };
    public class AppointmentData
    {
        public int Id { get; set; }
        public string Subject { get; set; }
        public string Location { get; set; }
        public DateTime StartTime { get; set; }
        public DateTime EndTime { get; set; }
        public string Description { get; set; }
        public bool IsAllDay { get; set; }
    }
}

```

```

public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

OnDragStart

OnDragStart event triggers when an appointment is started to drag.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Width="100%" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEvents TValue="AppointmentData"
OnDragStart="OnDragStart"></ScheduleEvents>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 3, 10);
public void OnDragStart(DragEventArgs<AppointmentData> args)
{
args.Scroll.Enable = false;    //To prevent scroll action on dragging
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 3, 10, 9, 30, 0) , EndTime = new DateTime(2020, 3, 10, 12, 0,
0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

OnEventClick

OnEventClick event triggers when the events are single clicked or on single tapping the events on the mobile devices.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" @ref="@ScheduleRef" Width="100%"
Height="550px" @bind-SelectedDate="@CurrentDate">
<ScheduleEvents TValue="AppointmentData"
OnEventClick="OnEventClick"></ScheduleEvents>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 3, 10);
SfSchedule<AppointmentData> ScheduleRef;
public async Task OnEventClick(EventClickArgs<AppointmentData> args)
{
args.Cancel = true;
await ScheduleRef.OpenEditorAsync(args.Event, CurrentAction.Save); //To
open the editor on event click
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 3, 10, 9, 30, 0) , EndTime = new DateTime(2020, 3, 10, 12, 0,
0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}
```

OnEventDoubleClick

OnEventDoubleClick event triggers when the events are double clicked.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
```

```

<SfSchedule TValue="AppointmentData" @ref="@ScheduleRef" Width="100%"
Height="550px" @bind-SelectedDate="@CurrentDate">
  <ScheduleEvents TValue="AppointmentData"
  OnEventDoubleClick="OnEventDoubleClick"></ScheduleEvents>
  <ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
  <ScheduleViews>
    <ScheduleView Option="View.Day"></ScheduleView>
    <ScheduleView Option="View.Week"></ScheduleView>
    <ScheduleView Option="View.WorkWeek"></ScheduleView>
    <ScheduleView Option="View.Month"></ScheduleView>
    <ScheduleView Option="View.Agenda"></ScheduleView>
  </ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 3, 10);
SfSchedule<AppointmentData> ScheduleRef;
public async Task OnEventDoubleClick(EventClickArgs<AppointmentData> args)
{
  args.Cancel = true;
  await ScheduleRef.OpenQuickInfoPopupAsync(args.Event); //To open Quick popup
  on double click
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
  new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
  DateTime(2020, 3, 10, 9, 30, 0) , EndTime = new DateTime(2020, 3, 10, 12, 0,
  0) }
};
public class AppointmentData
{
  public int Id { get; set; }
  public string Subject { get; set; }
  public string Location { get; set; }
  public DateTime StartTime { get; set; }
  public DateTime EndTime { get; set; }
  public string Description { get; set; }
  public bool IsAllDay { get; set; }
  public string RecurrenceRule { get; set; }
  public string RecurrenceException { get; set; }
  public Nullable<int> RecurrenceID { get; set; }
}
}

```

OnPopupClose

OnPopupClose event triggers before any of the Scheduler popups close on the page.

In case, if you need to prevent only specific popups on Scheduler, then you can check the condition based on the popup type. The types of the popup that can be checked within the **OnPopupClose** event are as follows.

PopupType	Description
Editor	For Detailed editor window.

- | **QuickInfo** | For Quick popup which opens on cell click. |
- | **EditEventInfo** | For Quick popup which opens on event click. |
- | **ViewEventInfo** | For Quick popup which opens on responsive mode. |
- | **EventContainer** | For more event indicator popup. |
- | **RecurrenceAlert** | For edit recurrence event alert popup. |
- | **DeleteAlert** | For delete confirmation popup. |
- | **ValidationAlert** | For validation alert popup. |
- | **RecurrenceValidationAlert** | For recurrence validation alert popup. |

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Width="100%" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEvents TValue="AppointmentData"
OnPopupClose="OnPopupClose"></ScheduleEvents>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 3, 10);
public void OnPopupClose(PopupCloseEventArgs<AppointmentData> args)
{
if (args.Type == PopupType.Editor || args.Type == PopupType.QuickInfo)
{
args.Data.Subject = (args.Data.Subject == "Add title") ? "New event" :
args.Data.Subject; //The default subject is changed from Add Title to New
event
}
}
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 3, 10, 9, 30, 0) , EndTime = new DateTime(2020, 3, 10, 12, 0,
0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
}
```

```
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}
```

OnPopupOpen

OnPopupOpen event triggers before any of the Scheduler popups opens on the page.

In case, if you need to prevent only specific popups on Scheduler, then you can check the condition based on the popup type. The types of the popup that can be checked within the **OnPopupOpen** event are as follows.

PopupType	Description
Editor	For Detailed editor window.
QuickInfo	For Quick popup which opens on cell click.
EditEventInfo	For Quick popup which opens on event click.
ViewEventInfo	For Quick popup which opens on responsive mode.
EventContainer	For more event indicator popup.
RecurrenceAlert	For edit recurrence event alert popup.
DeleteAlert	For delete confirmation popup.
ValidationAlert	For validation alert popup.
RecurrenceValidationAlert	For recurrence validation alert popup.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Width="100%" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEvents TValue="AppointmentData"
OnPopupOpen="OnPopupOpen"></ScheduleEvents>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 3, 10);
public void OnPopupOpen(PopupOpenEventArgs<AppointmentData> args)
{
if (args.Type == PopupType.ValidationAlert)
{
args.Cancel = true; //To prevent start and end time validation alert
```

```

}
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
    DateTime(2020, 3, 10, 9, 30, 0) , EndTime = new DateTime(2020, 3, 10, 12, 0,
    0) }
};
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
}
}

```

OnRenderCell

OnRenderCell event triggers before each element of the Schedule rendering on the page.

The **ElementType** that can be checked within the **OnRenderCell** event are as follows.

ElementType	Description
----- -----	
AllDayCells	Triggers on all day cell rendering.
DateHeader	Triggers on header cell rendering.
EmptyCells	Triggers on empty cell rendering on header bar.
MajorSlot	Triggers on major time slot cell rendering.
MinorSlot	Triggers on minor time slot cell rendering.
MonthCells	Triggers on month cell rendering.
MonthDay	Triggers on header cell in month view rendering.
ResourceGroupCells	Triggers on rendering of work cells for parent resource.
ResourceHeader	Triggers on resource header cell rendering.
WorkCells	Triggers on work cell rendering.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Width="100%" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEvents TValue="AppointmentData"
OnRenderCell="OnRenderCell"></ScheduleEvents>

```

```

<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
<style>
.e-schedule .e-vertical-view .e-work-hours.custom-class {
background-color: ivory;
}
</style>
@code{
DateTime CurrentDate = new DateTime(2020, 3, 10);
public string[] CustomClass = { "custom-class" };
public void OnRenderCell(RenderCellEventArgs args)
{
if (args.ElementType == ElementType.WorkCells)
{
args.CssClasses = new List<string>(CustomClass); //The default work hours
color is changed to ivory color
}
}
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 3, 10, 9, 30, 0) , EndTime = new DateTime(2020, 3, 10, 12, 0,
0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

OnResizeStart

OnResizeStart event triggers when an appointment is started to resize.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Width="100%" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEvents TValue="AppointmentData"
OnResizeStart="OnResizeStart"></ScheduleEvents>

```



```

<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 3, 10);
public void OnResizeStart(ResizeEventArgs<AppointmentData> args)
{
args.Scroll.Enable = false; //To prevent scrolling will resize the event
args.Interval = 10; //To change the resizing time interval from 30
minutes(default) to 10 minutes
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 3, 10, 9, 30, 0) , EndTime = new DateTime(2020, 3, 10, 12, 0,
0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

Resized

Resized event triggers when the resizing of appointment is stopped.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Width="100%" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEvents TValue="AppointmentData"
Resized="OnResized"></ScheduleEvents>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>

```

```

</SfSchedule>
@code{
    DateTime CurrentDate = new DateTime(2020, 3, 10);
    public void OnResized(ResizeEventArgs<AppointmentData> args)
    {
        args.Cancel = true;    //To prevent resize action
    }
    List<AppointmentData> DataSource = new List<AppointmentData>
    {
        new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
        DateTime(2020, 3, 10, 9, 30, 0) , EndTime = new DateTime(2020, 3, 10, 12, 0,
        0) }
    };
    public class AppointmentData
    {
        public int Id { get; set; }
        public string Subject { get; set; }
        public string Location { get; set; }
        public DateTime StartTime { get; set; }
        public DateTime EndTime { get; set; }
        public string Description { get; set; }
        public bool IsAllDay { get; set; }
        public string RecurrenceRule { get; set; }
        public string RecurrenceException { get; set; }
        public Nullable<int> RecurrenceID { get; set; }
    }
}

```

Context Menu in Blazor Scheduler Component

The context menu can be displayed on work cells and appointments of Scheduler by making use of the **ContextMenu** control manually from the application end. In the following code example, context menu control is being added from sample end and set its target as **Scheduler** and the target element is got by using **GetTargetCellAsync** public method in Blazor.

On Scheduler cells, the menu items can be displayed such as **New Event**, **New Recurring Event** and **Today** option. For appointments, its related options can be displayed such as **Edit Event** and **Delete Event**. The default event window can be opened for appointment creation and editing using the **OpenEditorAsync** method of Scheduler.

The deletion of appointments can be done by using the **DeleteEventAsync** public method. Also, the **SelectedDate** property can be used to navigate between different dates.

You can also display custom menu options on Scheduler cells and appointments. Context menu will open on tap-hold in responsive mode.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Navigations
<SfSchedule TValue="AppointmentData" @ref="ScheduleRef" Height="650px"
@bind-SelectedDate="@SelectedDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>

```

```

<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
<SfContextMenu TValue="MenuItem" @ref="ContextMenuObj" CssClass="schedule-
context-menu" Target=".e-schedule">
<MenuItems>
<MenuItem Text="New Event" Id="Add" Hidden="@isCell"></MenuItem>
<MenuItem Text="New Recurring Event" Hidden="@isCell"
Id="AddRecurrence"></MenuItem>
<MenuItem Text="Today" Id="Today" Hidden="@isCell"></MenuItem>
<MenuItem Text="Edit Event" Id="Save" Hidden="@isEvent"></MenuItem>
<MenuItem Text="Edit Event" Id="EditRecurrenceEvent" Hidden="@isRecurrence">
<MenuItems>
<MenuItem Text="Edit Occurrence" Id="EditOccurrence"
Hidden="@isRecurrence"></MenuItem>
<MenuItem Text="Edit Series" Id="EditSeries"
Hidden="@isRecurrence"></MenuItem>
</MenuItems>
</MenuItem>
<MenuItem Text="Delete Event" Id="Delete" Hidden="@isEvent"></MenuItem>
<MenuItem Text="Delete Event" Id="DeleteRecurrenceEvent"
Hidden="@isRecurrence">
<MenuItems>
<MenuItem Text="Delete Occurrence" Id="DeleteOccurrence"
Hidden="@isRecurrence"></MenuItem>
<MenuItem Text="Delete Series" Id="DeleteSeries"
Hidden="@isRecurrence"></MenuItem>
</MenuItems>
</MenuItem>
</MenuItems>
<MenuEvents TValue="MenuItem" OnOpen="OnOpen"
ItemSelected="OnItemSelected"></MenuEvents>
</SfContextMenu>
@code{
private DateTime SelectedDate = new DateTime(2020, 1, 8);
SfSchedule<AppointmentData> ScheduleRef;
SfContextMenu<MenuItem> ContextMenuObj;
private bool isCell { get; set; }
private bool isEvent { get; set; }
private bool isRecurrence { get; set; }
public AppointmentData EventData { get; set; }
public CellClickEventArgs CellData { get; set; }
public async Task OnOpen(BeforeOpenCloseMenuEventArgs<MenuItem> args)
{
if (args.ParentItem == null)
{
CellData = await ScheduleRef.GetTargetCellAsync((int)args.Left,
(int)args.Top);
if (CellData == null)
{
EventData = await ScheduleRef.GetTargetEventAsync((int)args.Left,
(int)args.Top);
if (EventData.Id == 0)
{

```

```

args.Cancel = true;
}
if (EventData.RecurrenceRule != null)
{
    isCell = isEvent = true;
    isRecurrence = false;
}
else
{
    isCell = isRecurrence = true;
    isEvent = false;
}
}
else
{
    isCell = false;
    isEvent = isRecurrence = true;
}
}
}
}
public async Task OnItemSelected(MenuEventArgs<MenuItem> args)
{
    var SelectedMenuItem = args.Item.Id;
    var ActiveCellsData = await ScheduleRef.GetSelectedCellsAsync();
    if (ActiveCellsData == null)
    {
        ActiveCellsData = CellData;
    }
    switch (SelectedMenuItem)
    {
        case "Today":
            SelectedDate = DateTime.Now;
            break;
        case "Add":
            await ScheduleRef.OpenEditorAsync(ActiveCellsData, CurrentAction.Add);
            break;
        case "AddRecurrence":
            await ScheduleRef.OpenEditorAsync(ActiveCellsData, CurrentAction.Add,
            RepeatType.Daily);
            break;
        case "Save":
            await ScheduleRef.OpenEditorAsync(EventData, CurrentAction.Save);
            break;
        case "EditOccurrence":
            await ScheduleRef.OpenEditorAsync(EventData, CurrentAction.EditOccurrence);
            break;
        case "EditSeries":
            List<AppointmentData> Events = await ScheduleRef.GetEventsAsync();
            EventData = (AppointmentData)Events.Where(data => data.Id ==
            EventData.RecurrenceID).FirstOrDefault();
            await ScheduleRef.OpenEditorAsync(EventData, CurrentAction.EditSeries);
            break;
        case "Delete":
            await ScheduleRef.DeleteEventAsync(EventData);
            break;
        case "DeleteOccurrence":

```

```

await ScheduleRef.DeleteEventAsync(EventData,
CurrentAction.DeleteOccurrence);
break;
case "DeleteSeries":
await ScheduleRef.DeleteEventAsync(EventData, CurrentAction.DeleteSeries);
break;
}
}
public List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 8, 11, 30, 0) , EndTime = new DateTime(2020, 1, 8, 12, 30,
0) },
new AppointmentData { Id = 2, Subject = "Conference", StartTime = new
DateTime(2020, 1, 10, 11, 30, 0) , EndTime = new DateTime(2020, 1, 10, 12,
30, 0) },
new AppointmentData { Id = 3, Subject = "Seminar", StartTime = new
DateTime(2020, 1, 6, 9, 30, 0) , EndTime = new DateTime(2020, 1, 6, 11, 0,
0),
RecurrenceRule = "FREQ=DAILY;INTERVAL=1;COUNT=5" }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

Style And Appearance in Blazor Scheduler Component

To modify the Scheduler appearance, you need to override the default CSS of Scheduler. Also, there is an option to create our own custom theme using [Theme Studio](#). Please find the list of CSS classes in Scheduler.

| CSS class | Purpose |

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

| .e-schedule .e-vertical-view .e-work-cells | Work cells in vertical views of scheduler |

| .e-schedule .e-month-view .e-work-cells | Work cells in month view of scheduler |

| .e-schedule .e-month-view .e-other-month | Work cells of other month in month view of scheduler |

| .e-schedule .e-timeline-view .e-work-cells | Work cells in timeline views of scheduler |

| .e-schedule .e-timeline-month-view .e-work-cells | Work cells in timeline month view of scheduler |

| .e-schedule .e-timeline-year-view .e-work-cells | Work cells in timeline year view of scheduler |

| .e-schedule .e-timeline-year-view .e-work-cells.e-other-month | Work cells of other month in timeline year view of scheduler |

| .e-schedule .e-month-agenda-view .e-work-cells | Work cells in month agenda view of scheduler |

| .e-schedule .e-month-agenda-view .e-other-month | Work cells of other month in month agenda view of scheduler |

| .e-schedule .e-year-view .e-calendar-wrapper .e-month-calendar.e-calendar .e-other-month | Work cells of other month in year view of scheduler |

| .e-schedule .e-vertical-view .e-all-day-cells | All day cells in vertical views of scheduler |

| .e-schedule .e-vertical-view .e-work-hours | Work hour cells in vertical views of scheduler |

| .e-schedule .e-month-view .e-work-days | Work day cells in month view of scheduler |

| .e-schedule .e-month-agenda-view .e-work-days | Work day cells in month agenda view of scheduler |

| .e-schedule .e-timeline-view .e-work-hours | Work hour cells in timeline views of scheduler |

| .e-schedule .e-timeline-month-view .e-work-days | Work day cells in timeline month view of scheduler |

| .e-schedule .e-timeline-year-view .e-work-cells.e-work-days | Work day cells in timeline year view of scheduler |

| .e-schedule .e-vertical-view .e-day-wrapper .e-appointment | Appointment in vertical views of scheduler |

| .e-schedule .e-vertical-view .e-all-day-appointment-wrapper .e-appointment | All day Appointment in vertical views of scheduler |

| .e-schedule .e-month-view .e-appointment | Appointment in month view of scheduler |

| .e-schedule .e-timeline-view .e-appointment | Appointment in timeline views of scheduler |

| .e-schedule .e-timeline-month-view .e-appointment | Appointment in timeline month view of scheduler |

| .e-schedule .e-timeline-year-view .e-event-table .e-appointment | Appointment in timeline year view of scheduler |

| .e-schedule .e-year-view .e-calendar-wrapper .e-month-calendar.e-calendar .e-appointment | Appointment in year view of scheduler |

| .e-schedule .e-agenda-view .e-appointment | Appointment in agenda view of scheduler |

| .e-schedule .e-month-agenda-view .e-appointment-indicator | Appointment in month agenda view of scheduler |

| .e-schedule .e-block-appointment | Block appointment in scheduler |

| .e-schedule .e-read-only | Read only appointment in scheduler. |

| e-appointment-border | Appointment which are currently selected, use the appointment class hierarchical based on your views. |

| e-selected-cells | Work cells which are currently selected, use the work cell class hierarchical based on your views. |

e-header-cells	Header cells of scheduler, use the work cells hierarchical based on your views.
.e-schedule .e-vertical-view .e-resource-cells	Resource cells in vertical views of scheduler.
.e-schedule .e-month-view .e-resource-cells	Resource cells in month view of scheduler.
.e-schedule .e-timeline-view .e-resource-cells	Resource cells in timeline views of scheduler.
.e-schedule .e-timeline-month-view .e-resource-cells	Resource cells in timeline month view of scheduler.
e-parent-node	Parent resource cells in timeline views of scheduler.
e-child-node	Child resource cells in timeline views of scheduler.

How To

Blazor Scheduler Component in WebAssembly (WASM) App

This article provides a step-by-step instructions to configure Syncfusion Blazor Scheduler in a simple Blazor WebAssembly application using [Visual Studio 2019](#).

Note: Starting with version 17.4.0.39 (2019 Volume 4), you need to include a valid license key (either paid or trial key) within your applications. Please refer to this [help topic](#) for more information.

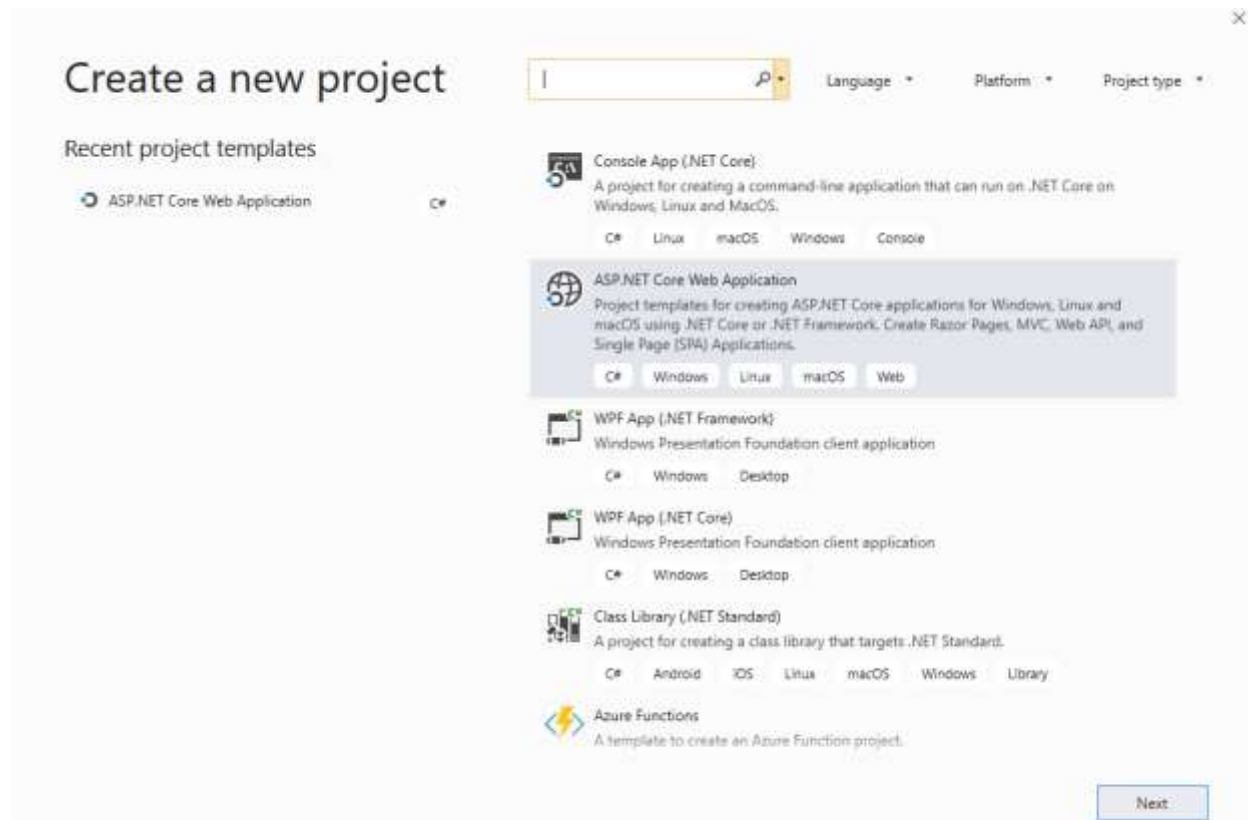
Prerequisites

- [Visual Studio 2019](#)
- [.NET Core SDK 3.1.8](#) / [.NET 5.0 SDK](#)

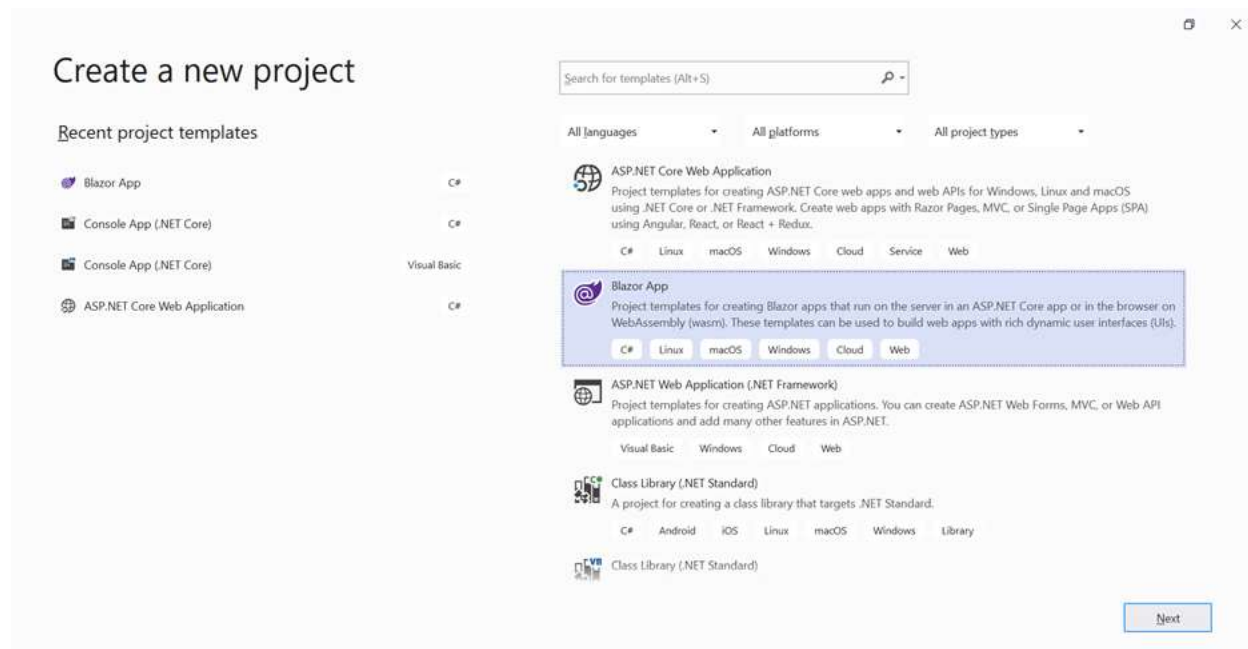
.NET Core SDK 3.1.8 requires Visual Studio 2019 16.7 or later. **.NET 5.0** requires Visual Studio 2019 16.8 or later.

Create a Blazor WebAssembly project in Visual Studio 2019

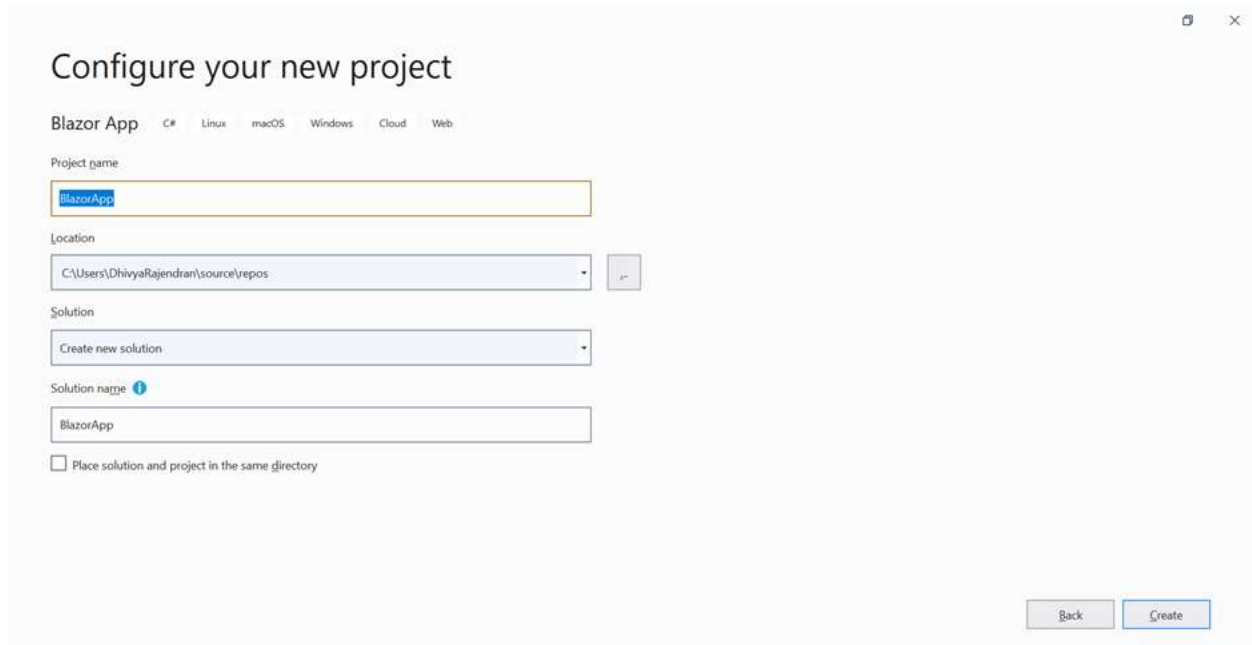
1. Choose **Create a new project** from the Visual Studio dashboard.



2. Select **Blazor App** from the template, and then click **Next** button.



- Now, the project configuration window will popup. Click **Create** button to create a new project with the default project configuration.



Configure your new project

Blazor App C# Linux macOS Windows Cloud Web

Project name

BlazorApp

Location

C:\Users\DhivyaRajendran\source\repos

Solution

Create new solution

Solution name ⓘ

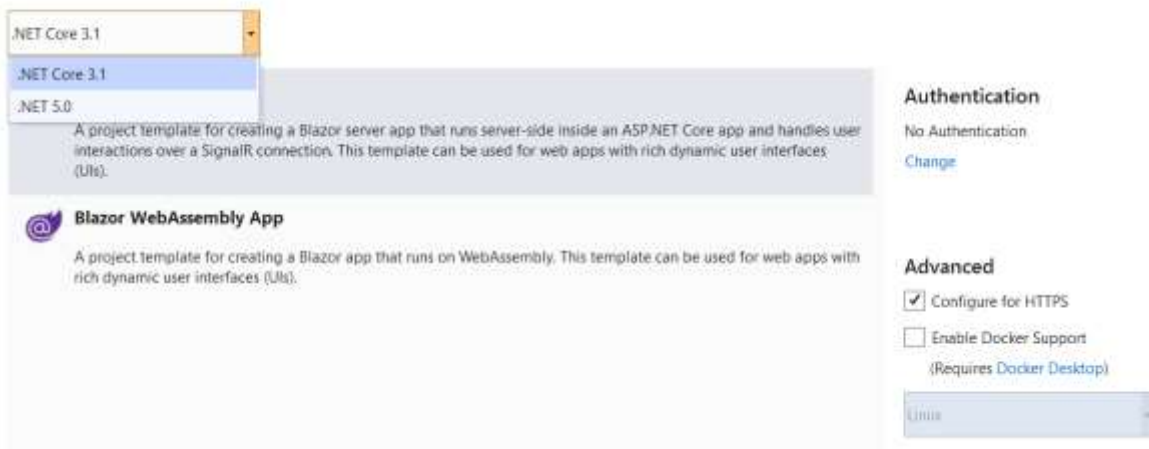
BlazorApp

☐ Place solution and project in the same directory

Back Create

- Select the target Framework **ASP.NET Core 3.1** or **.NET 5.0** at the top of the application based on the required target.

Create a new Blazor app



.NET Core 3.1

.NET Core 3.1

.NET 5.0

A project template for creating a Blazor server app that runs server-side inside an ASP.NET Core app and handles user interactions over a SignalR connection. This template can be used for web apps with rich dynamic user interfaces (UIs).

Blazor WebAssembly App

A project template for creating a Blazor app that runs on WebAssembly. This template can be used for web apps with rich dynamic user interfaces (UIs).

Authentication

No Authentication

[Change](#)

Advanced

☒ Configure for HTTPS

☐ Enable Docker Support


(Requires Docker Desktop)

Linux


- Choose **Blazor WebAssembly App** from the dashboard, and then click **Create** button to create a new Blazor WebAssembly application.

Create a new Blazor app

.NET Core 3.1

**Blazor Server App**

A project template for creating a Blazor server app that runs server-side inside an ASP.NET Core app and handles user interactions over a SignalR connection. This template can be used for web apps with rich dynamic user interfaces (UIs).

**Blazor WebAssembly App**

A project template for creating a Blazor app that runs on WebAssembly. This template can be used for web apps with rich dynamic user interfaces (UIs).

Authentication

No Authentication

[Change](#)

Advanced

☒ Configure for HTTPS

☐ Enable Docker Support
(Requires Docker Desktop)

Linux

☐ ASP.NET Core hosted

☐ Progressive Web Application

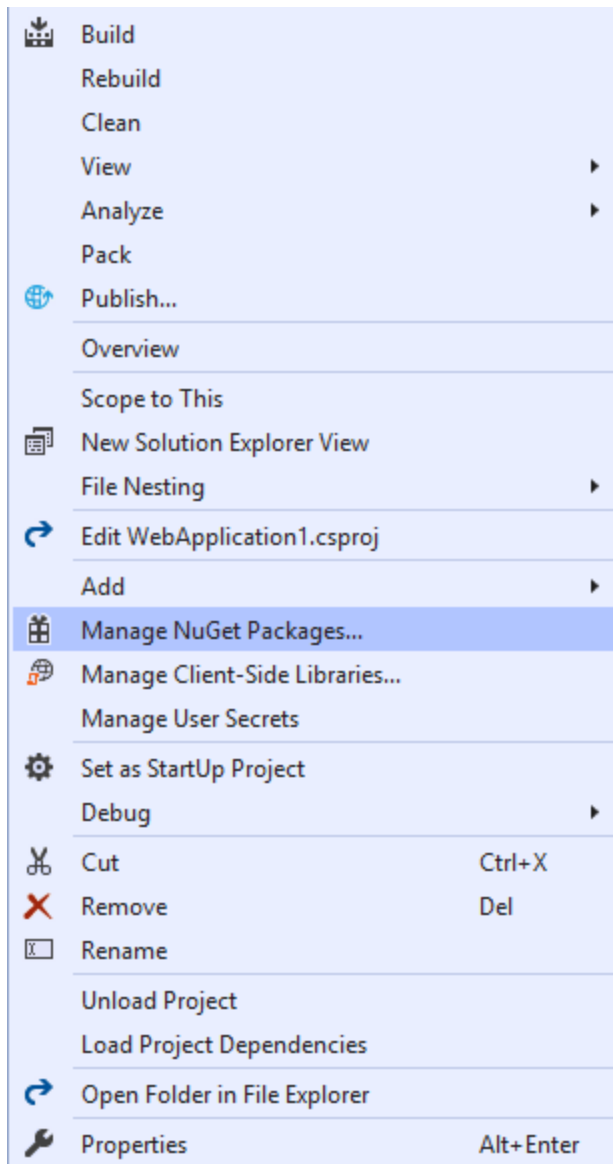
Installing Syncfusion Blazor packages in the application

You can use any one of the below standard to install the Syncfusion Blazor library in your application.

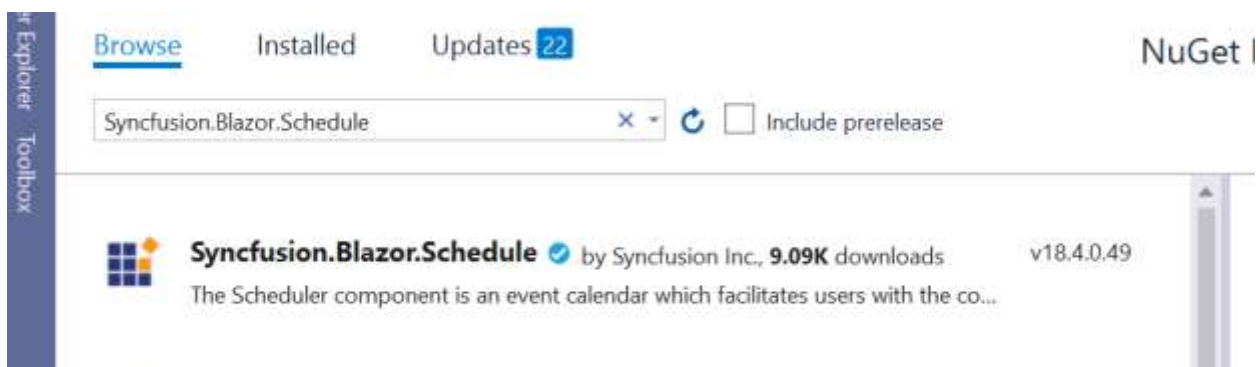
Using Syncfusion Blazor individual NuGet Packages [New standard]

Starting with Volume 4, 2020 (v18.4.0.30) release, Syncfusion provides [individual NuGet packages](#) for our Syncfusion Blazor components. We highly recommend this new standard for your Blazor production applications. Refer to [this section](#) to know the benefits of the individual NuGet packages.

1. Now, install **Syncfusion.Blazor.Schedule** NuGet package to the new application using the **NuGet Package Manager**. For more details about available NuGet packages, refer to the [Individual NuGet Packages](#) documentation.
2. Right-click the project, and then select Manage NuGet Packages.



3. Search **Syncfusion.Blazor.Schedule** keyword in the Browse tab and install **Syncfusion.Blazor.Schedule** NuGet package in the application.



4. The Syncfusion Blazor Schedule package will be included in the newly created project once the installation process is completed.
5. Add the Syncfusion bootstrap4 theme in the `element` of the `~/wwwroot/index.html` page.

HTML

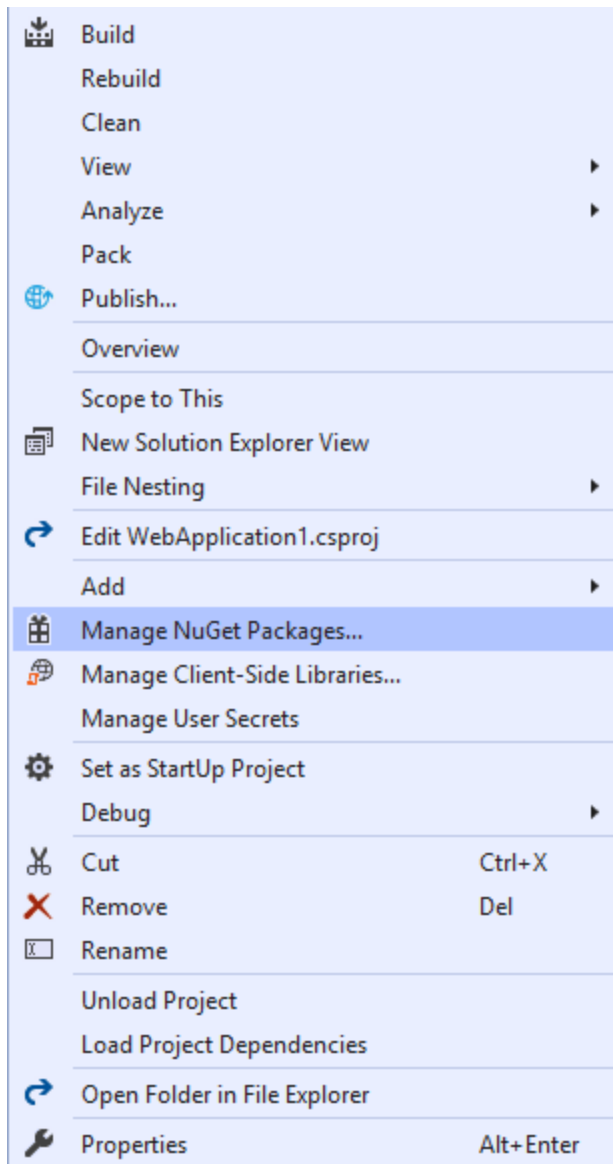
```
<head>
....
....
<link href="_content/Syncfusion.Blazor.Themes/bootstrap4.css"
rel="stylesheet" />
</head>
```

Warning: Syncfusion.Blazor package should not be installed along with [individual NuGet packages](#). Hence, the above Syncfusion.Blazor.Themes static web assets (styles) has to be added in the application.

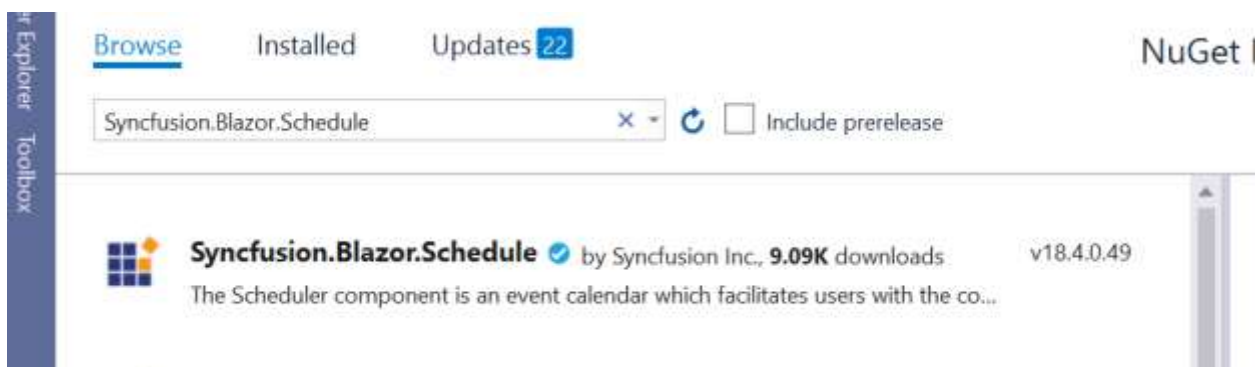
[Using Syncfusion.Blazor NuGet Package \[Old standard\]](#)

Warning: If you prefer the above new standard (individual NuGet packages), then skip this section. Using both old and new standards in the same application will throw ambiguous compilation errors.

1. Now, install **Syncfusion.Blazor** NuGet package to the newly created application by using the **NuGet Package Manager**. Right-click the project and then select Manage NuGet Packages.



2. Search **Syncfusion.Blazor** keyword in the Browse tab and install **Syncfusion.Blazor** NuGet package in the application.



3. The Syncfusion Blazor package will be installed in the project once the installation process is completed.
4. Add the Syncfusion bootstrap4 theme in the element of the `~/wwwroot/index.html` page.

HTML

```
<head>
....
....
<link href="_content/Syncfusion.Blazor/styles/bootstrap4.css"
rel="stylesheet" />
</head>
```

The same theme file can be referred through the CDN version by using <https://cdn.syncfusion.com/blazor/{{ site.blazorversion }}/styles/bootstrap4.css>.

Adding Syncfusion Blazor component and running the application

1. Open `~/_Imports.razor` file and import the `Syncfusion.Blazor` namespace.

ASPX-CS

```
@using Syncfusion.Blazor
@using Syncfusion.Blazor.Schedule
```

2. Open the `~/Program.cs` file and register the Syncfusion Blazor Service.

C#

```
using Syncfusion.Blazor;
namespace WebApplication1
{
    public class Program
    {
        public static async Task Main(string[] args)
        {
            ....
            ....
            builder.Services.AddSyncfusionBlazor();
            await builder.Build().RunAsync();
        }
    }
}
```

Initialize the Scheduler component

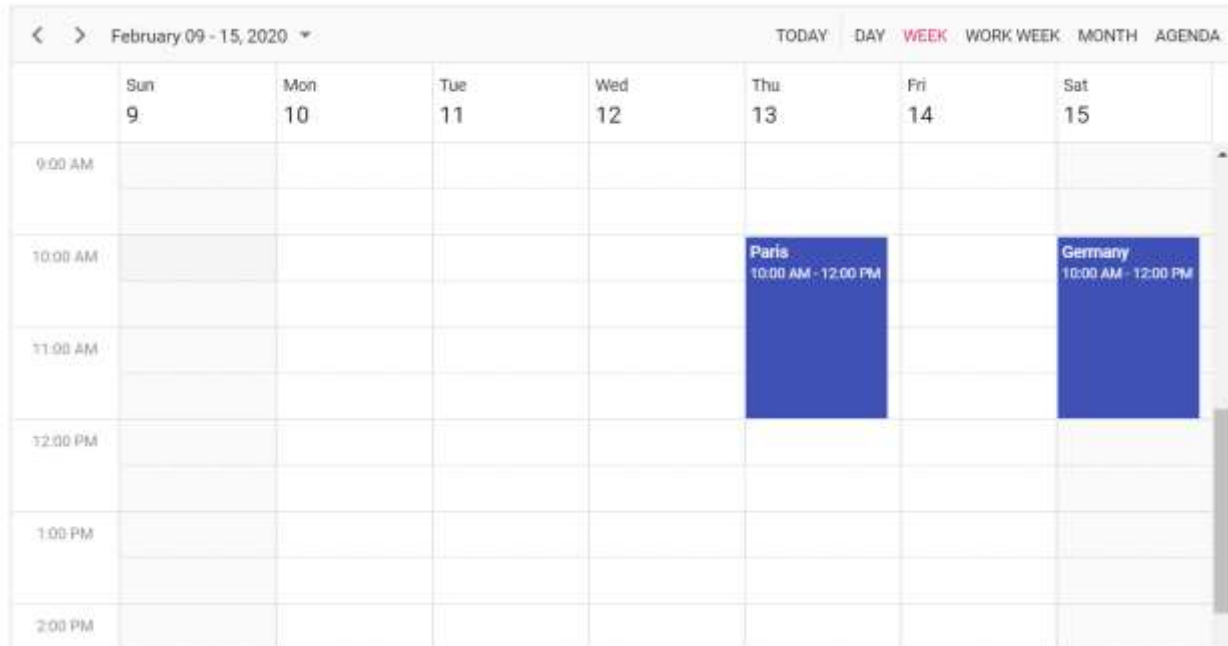
The [Blazor Scheduler](#) component can be rendered on the page by defining the `SfSchedule` tag helper. And also, define the required Scheduler views in the `ScheduleView` tag helper. Add the following code example to your `index.razor` page which is available within the `~/Pages/` folder, to initialize the Scheduler component.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="650px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 2, 14);
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Paris", StartTime = new
DateTime(2020, 2, 13, 10, 0, 0) , EndTime = new DateTime(2020, 2, 13, 12, 0,
0) },
new AppointmentData { Id = 2, Subject = "Germany", StartTime = new
DateTime(2020, 2, 15, 10, 0, 0) , EndTime = new DateTime(2020, 2, 15, 12, 0,
0) }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```



Refer to our [Blazor Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [Blazor Scheduler example](#) to understand how to manage appointments with multiple resources.

Open Editor Window on Single Click in Blazor Scheduler Component

By default, the editor window will open on double clicking the cell or appointment. In the following code example, we have opened the editor window on single click using `OpenEditorAsync` public method within `OnCellClick` and `OnEventClick` Scheduler events.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule @ref="ScheduleRef" TValue="AppointmentData"
ShowQuickInfo="false" Height="550px" @bind-SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleEvents TValue="AppointmentData" OnCellClick="OnCellClick"
OnEventClick="OnEventClick"></ScheduleEvents>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 3, 11);
SfSchedule<AppointmentData> ScheduleRef;
public async Task OnCellClick(CellClickEventArgs args)
{
args.Cancel = true;
await ScheduleRef.OpenEditorAsync(args, CurrentAction.Add); //to open the
editor on cell click
}
```



```

public async Task OnEventClick(EventClickArgs<AppointmentData> args)
{
    args.Cancel = true;
    await ScheduleRef.OpenEditorAsync(args.Event, CurrentAction.Save); //to open
    the editor on event click
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData{ Id = 1, Subject = "Meeting", StartTime = new
    DateTime(2020, 3, 11, 9, 30, 0) , EndTime = new DateTime(2020, 3, 11, 11, 0,
    0)}
};
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
}
}

```

Add Multi Events in different slots in Blazor Scheduler Component

In Blazor Scheduler, we can select the different time slots (10:00 - 10:30, 8:00 - 8:30) by holding CTRL key and click on cells using `OnCellClick` event. In the following code example, events are created on selected timeslots when clicking the **Add Appointments** button.

ASPX-CS

```

@using Syncfusion.Blazor
@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Buttons
<SfButton OnClick="OnButtonClick">Add Appointments</SfButton>
<SfSchedule @ref="ScheduleObj" TValue="AppointmentData" Width="100%"
Height="650px" SelectedDate="@ (new DateTime(2020, 3, 10))">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleEvents TValue="AppointmentData"
OnCellClick="OnCellClicked"></ScheduleEvents>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
SfSchedule<AppointmentData> ScheduleObj;
public List<CellClickEventArgs> CellDetails = new
List<CellClickEventArgs>();
public async void OnCellClicked(CellClickEventArgs args)

```

```

{
    if (args.MouseEventArgs.CtrlKey == true) //to check whether CTRL key is
        pressed
    {
        await this.ScheduleObj.CloseQuickInfoPopupAsync();
        CellClickEventArgs cell = await this.ScheduleObj.GetSelectedCellsAsync();
        CellDetails.Add(cell);
    }
}

public async Task OnButtonClick()
{
    for (int i = 0; i < CellDetails.Count; i++)
    {
        Random rnd = new Random();
        int Id = rnd.Next(1000);
        List<AppointmentData> newData = new List<AppointmentData>();
        newData.Add(new AppointmentData
        {
            Id = Id,
            Subject = "Added events",
            StartTime = CellDetails[i].StartTime,
            EndTime = CellDetails[i].EndTime
        });
        await this.ScheduleObj.AddEventAsync(newData); //to add appointments to the
        scheduler
    }
    CellDetails.Clear();
}

List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData { Id = 1, Subject = "Eclipse", StartTime = new
        DateTime(2020, 3, 9, 9, 30, 0) , EndTime = new DateTime(2020, 3, 9, 11, 0,
        0) },
    new AppointmentData { Id = 2, Subject = "Crash", StartTime = new
        DateTime(2020, 3, 11, 11, 30, 0), EndTime = new DateTime(2020, 3, 11, 13, 0,
        0) }
};

public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
}
}

```

Custom Header in Blazor Scheduler Component

The Scheduler header bar can be hidden by setting `false` to `ShowHeaderBar` and use Syncfusion Toolbar control to customize our own header. In the following code example, the Scheduler views can be changed by using the Dropdowns available in the custom Header.

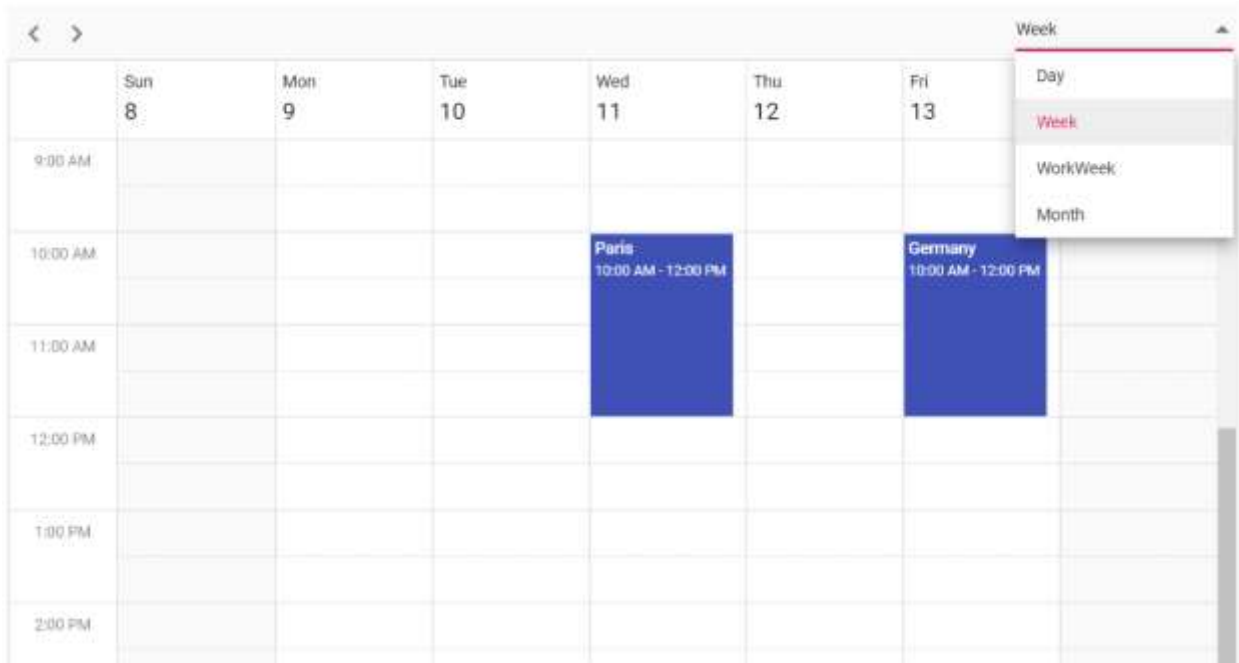
ASPX-CS

```
@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Navigations
@using Syncfusion.Blazor.DropDowns
<SfToolbar>
<ToolbarItems>
<ToolbarItem Align="ItemAlign.Left" PrefixIcon="previous" OnClick="@Clicked"
TooltipText="Previous"></ToolbarItem>
<ToolbarItem Align="ItemAlign.Left" PrefixIcon="next" OnClick="@Clicked"
TooltipText="Next"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem Type="ItemType.Input" Align="ItemAlign.Right">
<Template>
<SfDropDownList TItem="Views" TValue="string" ID="Views"
DataSource="@ViewsList" Width="120" @bind-Index="@index">
<DropDownListEvents TItem="Views" TValue="string"
ValueChange="OnValueChanged"></DropDownListEvents>
<DropDownListFieldSettings Value="Text"></DropDownListFieldSettings>
</SfDropDownList>
</Template>
</ToolbarItem>
</ToolbarItems>
</SfToolbar>
<SfSchedule @bind-SelectedDate="@DateValue" TValue="AppointmentData" @bind-
CurrentView="@MyView" Height="650px" ShowHeaderBar="false">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
<style>
.previous:before {
content: '\e937';
}
.next:before {
content: '\e956'
}
</style>
@code{
public DateTime DateValue { get; set; } = new DateTime(2020, 2, 11);
public View MyView = View.Week;
private int? index = 1;
public void Clicked(ClickEventArgs args)
{
int addDay = 7;
if (MyView == View.Day) addDay = 1;
```

```

if (MyView == View.Week) addDay = 7;
if (MyView == View.WorkWeek) addDay = 5;
if (MyView == View.Month) addDay = 30;
if (args.Item.PrefixIcon == "previous")
{
    DateValue = DateValue.AddDays(-addDay);
}
if (args.Item.PrefixIcon == "next")
{
    DateValue = DateValue.AddDays(addDay);
}
}
public void OnValueChange(ChangeEventArgs<string, Views> args)
{
    this.MyView = (View)Enum.Parse(typeof(View), args.Value);
}
public class Views
{
    public string ID { get; set; }
    public string Text { get; set; }
}
List<Views> ViewsList = new List<Views> {
    new Views() { ID= "1", Text= "Day" },
    new Views() { ID= "2", Text= "Week" },
    new Views() { ID= "3", Text= "WorkWeek" },
    new Views() { ID= "4", Text= "Month" }
};
List<AppointmentData> DataSource = new List<AppointmentData>
{
    new AppointmentData { Id = 1, Subject = "Paris", StartTime = new
    DateTime(2020, 2, 11, 10, 0, 0) , EndTime = new DateTime(2020, 2, 11, 12, 0,
    0) },
    new AppointmentData { Id = 2, Subject = "Germany", StartTime = new
    DateTime(2020, 2, 13, 10, 0, 0) , EndTime = new DateTime(2020, 2, 13, 12, 0,
    0) }
};
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
}
}

```



Prevent Event Creation and Deletion in Blazor Scheduler Component

By default, Scheduler allows the user to perform all the CRUD actions. The particular action can be prevented by setting false to the respective property. In the following code example, only edit actions are allowed as `AllowAdding` and `AllowDeleting` properties are set to false.

ASPX-CS

```
@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource" AllowAdding="false"
AllowDeleting="false"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 2, 11);
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Paris", StartTime = new
DateTime(2020, 2, 11, 10, 0, 0) , EndTime = new DateTime(2020, 2, 11, 12, 0,
0) },
new AppointmentData { Id = 2, Subject = "Germany", StartTime = new
DateTime(2020, 2, 13, 10, 0, 0) , EndTime = new DateTime(2020, 2, 13, 12, 0,
0) }
};
public class AppointmentData
{
public int Id { get; set; }
}
```

```

public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```

Show half-yearly view in Blazor Scheduler Component

The year view of the scheduler displays all the 365 days and their related appointments of a particular year. The year view can be customized by using the following properties.

- [FirstMonthOfYear](#)
- [MonthsCount](#)
- [MonthHeaderTemplate](#)

In the following code example, you can see how to render only the last six months of a year in the scheduler. To start with the month of June, `FirstMonthYear` is set to 6 and `MonthsCount` is set to 6 to render only 6 months.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
@using System.Globalization
<SfSchedule TValue="AppointmentData" Height="550px" @bind-
SelectedDate="@CurrentDate" FirstMonthOfYear="6" MonthsCount="6">
<ScheduleTemplates>
<ResourceHeaderTemplate>
@{
var resourceData = (context as TemplateContext).ResourceData as
ResourceData;
<div class='template-wrap'>
<div class="resource-details">
<div class="resource-name">@(resourceData.OwnerText) </div>
</div>
</div>
}
</ResourceHeaderTemplate>
<MonthHeaderTemplate>
@{
<div>@(monthHeaderTemplate((context as TemplateContext).Date)) </div>
}
</MonthHeaderTemplate>
</ScheduleTemplates>
<ScheduleGroup Resources="@Resources"></ScheduleGroup>
<ScheduleResources>
<ScheduleResource TItem="ResourceData" TValue="int" DataSource="@OwnersData"
Field="OwnerId" Title="Owner" Name="Owners" TextField="OwnerText"
IdField="Id" ColorField="OwnerColor"
AllowMultiple="true"></ScheduleResource>

```

```

</ScheduleResources>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Year"></ScheduleView>
<ScheduleView MaxEventsPerRow="2" Option="View.TimelineYear"
Orientation="Orientation.Horizontal" DisplayName="Horizontal Timeline Year"
IsSelected="true"></ScheduleView>
<ScheduleView MaxEventsPerRow="3" Option="View.TimelineYear"
Orientation="Orientation.Vertical" DisplayName="Vertical Timeline
Year"></ScheduleView>
</ScheduleViews>
</SfScheduler>
@code{
DateTime CurrentDate = new DateTime(2021, 7, 31);
public string[] Resources { get; set; } = { "Owners" };
public static string monthHeaderTemplate(DateTime date)
{
return date.ToString("MMMM yyyy", CultureInfo.CurrentCulture);
}
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2021, 8, 31, 9, 30, 0) , EndTime = new DateTime(2021, 8, 31, 11, 0,
0), OwnerId= 3 }
};
public List<ResourceData> OwnersData { get; set; } = new List<ResourceData>
{
new ResourceData{ OwnerText = "Nancy", Id = 1, OwnerColor = "#ffaa00" },
new ResourceData{ OwnerText = "Steven", Id = 2, OwnerColor = "#f8a398" },
new ResourceData{ OwnerText = "Robert", Id = 3, OwnerColor = "#7499e1" },
new ResourceData{ OwnerText = "Smith", Id = 4, OwnerColor = "#5978ee" },
new ResourceData{ OwnerText = "Michael", Id = 5, OwnerColor = "#df5286" }
};
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
public int OwnerId { get; set; }
}
public class ResourceData
{
public int Id { get; set; }
public string OwnerText { get; set; }
public string OwnerColor { get; set; }
}
}
<style>
.e-schedule .e-vertical-view .e-resource-cells {
height: 62px;

```

```

}
.e-schedule .template-wrap {
display: flex;
text-align: left;
}
.e-schedule .template-wrap .resource-details {
padding-left: 10px;
}
.e-schedule .template-wrap .resource-details .resource-name {
font-size: 16px;
font-weight: 500;
margin-top: 5px;
}
.e-schedule.e-device .template-wrap .resource-details .resource-name {
font-size: inherit;
font-weight: inherit;
}
.e-schedule.e-device .e-resource-tree-popup .e-fullrow {
height: 50px;
}
</style>

```

Setting Minimum and Maximum Date in Blazor Scheduler Component

In Scheduler, by default all the date ranges are available. A particular date range alone can be rendered in the Scheduler by setting the date range within `MinDate` and `MaxDate` properties. In the following code example, the Scheduler has been rendered from 2020 to 2023 alone.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px" MinDate="new
DateTime(2020, 1, 1)" MaxDate="new DateTime(2023, 12, 31)" @bind-
SelectedDate="@CurrentDate">
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 3, 11);
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Paris", StartTime = new
DateTime(2020, 3, 10, 10, 0, 0) , EndTime = new DateTime(2020, 3, 10, 12,
0, 0) },
new AppointmentData { Id = 2, Subject = "Germany", StartTime = new
DateTime(2020, 3, 12, 10, 0, 0) , EndTime = new DateTime(2020, 3, 12, 12, 0,
0) }
};
public class AppointmentData
{
public int Id { get; set; }
}

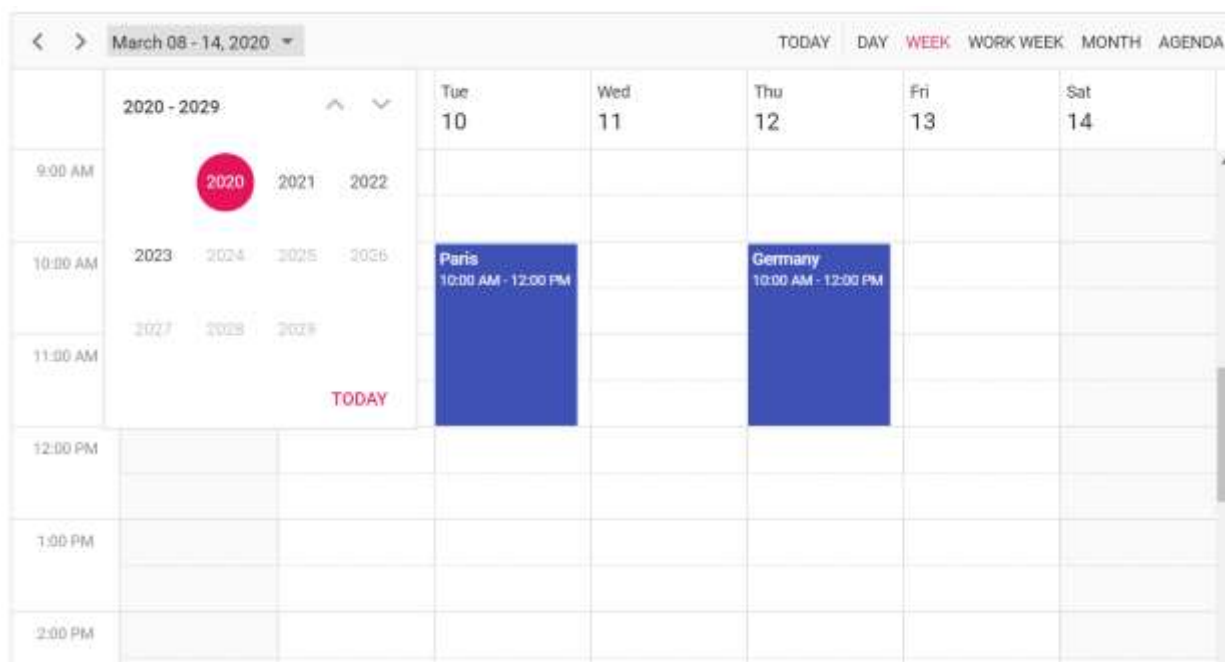
```



```

public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
}
}

```



Custom Editor With Validation in Blazor Scheduler Component

By default, in Blazor Scheduler field validation is available for built-in fields, if in case you want to validate a custom field you can go with custom editor and achieve validation using Data Annotations. Data Annotations helps you to define rules to the model classes or properties to perform data validation and display suitable messages to end users.

The Data Annotation can be enabled by referencing the `System.ComponentModel.DataAnnotations` namespace which maps the data annotations to the corresponding appointment fields. In the following code example, Syncfusion Blazor Dialog is used to render the custom editor and save/update the appointments using public methods.

ASPX-CS

```

@using Newtonsoft.Json
@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Calendars
@using Syncfusion.Blazor.DropDowns
@using Syncfusion.Blazor.Inputs
@using Syncfusion.Blazor.Popups
@using System.ComponentModel.DataAnnotations

```

```

<SfSchedule @ref="ScheduleRef" TValue="AppointmentData" Width="100%"
Height="650px" SelectedDate="@ (new DateTime(2020, 3, 11))">
<ScheduleEvents TValue="AppointmentData"
OnPopupOpen="PopupOpen"></ScheduleEvents>
<ScheduleGroup Resources="@Resources"></ScheduleGroup>
<ScheduleResources>
<ScheduleResource TValue="int[]" TItem="ResourceData"
DataSource="@OwnersData" Field="OwnerId" Title="Owner" Name="Owners"
TextField="OwnerText" IdField="Id" ColorField="OwnerColor"
AllowMultiple="true"></ScheduleResource>
</ScheduleResources>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
<SfDialog @bind-Visible="@DialogVisibility" IsModal="true" Width="575px"
ShowCloseIcon="true">
<DialogTemplates>
<Header> <div>Appointment Details</div> </Header>
<Content>
<EditForm Model="AppointmentValidation" OnValidSubmit="@OnValidSubmit">
<DataAnnotationsValidator />
<ValidationSummary />
<table class="custom-event-editor" width="550px" cellpadding="5">
<tbody>
<tr>
<td class="e-textlabel">Customer</td>
<td colspan="4">
<SfTextBox @ref="SubjectObj" @bind-
Value="@AppointmentValidation.Subject"></SfTextBox>
<ValidationMessage For="@(() => AppointmentValidation.Subject)" />
</td>
</tr>
<tr>
<td class="e-textlabel">From</td>
<td colspan="4">
<SfDateTimePicker TValue="DateTime?" ID="Start" @bind-
Value="@AppointmentValidation.StartTime"></SfDateTimePicker>
<ValidationMessage For="@(() => AppointmentValidation.StartTime)" />
</td>
</tr>
<tr>
<td class="e-textlabel">To</td>
<td colspan="4">
<SfDateTimePicker TValue="DateTime?" ID="End" @bind-
Value="@AppointmentValidation.EndTime"></SfDateTimePicker>
<ValidationMessage For="@(() => AppointmentValidation.EndTime)" />
</td>
</tr>
<tr>
<td class="e-textlabel">Appointment Type</td>
<td colspan="4">

```

```

<SfDropDownList TValue="int?" @bind-
Value="@AppointmentValidation.AppointmentType" ID="OwnerId"
DataSource="@AppointmentTypes" TItem="AppointmentType">
<DropDownListFieldSettings Value="Id"
Text="Text"></DropDownListFieldSettings>
</SfDropDownList>
<ValidationMessage For="@(() => AppointmentValidation.AppointmentType)" />
</td>
</tr>
<tr>
<td class="e-textlabel">Owner Detail</td>
<td colspan="4">
<SfDropDownList TValue="int" @bind-Value="@AppointmentValidation.OwnerId"
ID="IdOwnersData" DataSource="@OwnersData" TItem="ResourceData">
<DropDownListFieldSettings Value="Id"
Text="OwnerText"></DropDownListFieldSettings>
</SfDropDownList>
<ValidationMessage For="@(() => AppointmentValidation.OwnerId)" />
</td>
</tr>
<tr>
<td class="e-textlabel">Notes</td>
<td colspan="4">
<SfTextBox @ref="NotesObj" @bind-
Value="@AppointmentValidation.Description"></SfTextBox>
<ValidationMessage For="@(() => AppointmentValidation.Description)" />
</td>
</tr>
</tbody>
</table>
<button type="submit" class="e-btn">Save</button>
</EditForm>
</Content>
</DialogTemplates>
</SfDialog>
@code{
SfTextBox SubjectObj;
SfTextBox NotesObj;
private string Action;
Boolean DialogVisibility = false;
private int Id;
SfSchedule<AppointmentData> ScheduleRef;
private string[] Resources { get; set; } = { "Owners" };
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
0), OwnerId = 1 }
};
private List<ResourceData> OwnersData { get; set; } = new List<ResourceData>
{
new ResourceData{ OwnerText = "Nancy", Id = 1, OwnerColor = "#ffaa00" },
new ResourceData{ OwnerText = "Steven", Id = 2, OwnerColor = "#f8a398" },
new ResourceData{ OwnerText = "Michael", Id = 3, OwnerColor = "#7499e1" }
};
List<AppointmentType> AppointmentTypes = new List<AppointmentType>() {
new AppointmentType{ Id= 0, Text= "Not started" },

```

```

new AppointmentType{ Id= 1, Text= "In progress" },
new AppointmentType{ Id= 2, Text= "Completed" }
};
public AppointmentData AppointmentValidation = new AppointmentData();
private async void OnValidSubmit() //triggers on save button click
{
    DialogVisibility = false;
    AppointmentData EventData = new AppointmentData();
    EventData.Subject = AppointmentValidation.Subject;
    EventData.EndTime = (DateTime)AppointmentValidation.EndTime;
    EventData.StartTime = (DateTime)AppointmentValidation.StartTime;
    EventData.Description = AppointmentValidation.Description;
    EventData.OwnerId = AppointmentValidation.OwnerId;
    EventData.AppointmentType = AppointmentValidation.AppointmentType;
    EventData.Id = Id;
    if (Action == "CellClick")
    {
        await ScheduleRef.AddEventAsync(EventData); //to add new appointment
    }
    else
    {
        await ScheduleRef.SaveEventAsync(EventData); // to save the existing
        appointment
    }
}
private void PopupOpen(PopupOpenEventArgs<AppointmentData> args)
{
    if (args.Type == PopupType.Editor)
    {
        args.Cancel = true; //to prevent the default editor window
        this.Action = args.Data.Id == 0 ? "CellClick" : "AppointmentClick"; //to
        check whether the window opens on cell or appointment
        if (Action == "CellClick")
        {
            AppointmentValidation.StartTime = args.Data.StartTime;
            AppointmentValidation.EndTime = args.Data.EndTime;
            AppointmentValidation.Description = args.Data.Description;
            AppointmentValidation.OwnerId = args.Data.OwnerId;
            Random random = new Random();
            Id = random.Next(2, 1000);
        }
        else
        {
            Id = args.Data.Id;
            AppointmentValidation.StartTime = args.Data.StartTime;
            AppointmentValidation.EndTime = args.Data.EndTime;
            AppointmentValidation.AppointmentType = args.Data.AppointmentType;
            AppointmentValidation.OwnerId = args.Data.OwnerId;
        }
        AppointmentValidation.Description = args.Data.Description == null ? "" :
        args.Data.Description;
        AppointmentValidation.Subject = args.Data.Subject == null ? "Add title" :
        args.Data.Subject;
        DialogVisibility = true;
    }
    if (args.Type == PopupType.QuickInfo)
    {

```

```
args.Cancel = true;
}
}
class GroupData
{
public int OwnerId { get; set; }
}
public class ResourceData
{
public int Id { get; set; }
public string OwnerText { get; set; }
public string OwnerColor { get; set; }
}
public class AppointmentType
{
public int Id { get; set; }
public string Text { get; set; }
}
public class AppointmentData
{
public int Id { get; set; }
[Required]
[StringLength(16, ErrorMessage = "Identifier too long (16 character limit).")]
public string Subject { get; set; }
[Required]
public int? AppointmentType { get; set; }
[Required]
public DateTime? StartTime { get; set; }
[Required]
public DateTime? EndTime { get; set; }
[Required]
public int OwnerId { get; set; }
public string Location { get; set; }
public string EventType { get; set; }
public string Description { get; set; }
public bool IsAllDay { get; set; }
}
}
<style>
.custom-event-editor td {
padding: 7px;
padding-right: 16px;
}
</style>
```

The validation is applied on clicking the save button with empty fields as in the following image.

Appointment Details
×

- The AppointmentType field is required.

Customer

From

To

Appointment Type
The AppointmentType field is required.

Owner Detail

Steven

Notes

Save

Expand And Collapse Resource Dynamically in Blazor Scheduler Component

In Blazor Scheduler, a resource can be expanded or collapsed by clicking the expand/collapse icons. You can also programmatically expand or collapse the resource using public methods

`ExpandResourceAsync` and `CollapseResourceAsync` respectively. The following code shows how to expand and collapse the resource **Room 1** on external button click.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
@using Syncfusion.Blazor.Buttons
<SfButton Content="Expand Resource" OnClick="OnExpandButton"></SfButton>
<SfButton Content="Collapse Resource" OnClick="OnCollapseButton"></SfButton>
<SfSchedule @ref="ScheduleRef" TValue="AppointmentData" Height="550px"
@bind-SelectedDate="@CurrentDate">
<ScheduleGroup Resources="@Resources"></ScheduleGroup>
<ScheduleResources>
<ScheduleResource TItem="ResourceData" TValue="int" DataSource="@HotelData"
Field="HotelId" Title="Hotel" Name="Hotels" TextField="HotelText"
IdField="Id" ColorField="HotelColor"
AllowMultiple="false"></ScheduleResource>
<ScheduleResource TItem="ResourceData" TValue="int" DataSource="@RoomData"
Field="RoomId" Title="Room" Name="Rooms" TextField="RoomText" IdField="Id"
ColorField="RoomColor"
GroupIDField="RoomGroupId" AllowMultiple="false"></ScheduleResource>
<ScheduleResource TItem="ResourceData" TValue="int[]"
DataSource="@OwnersData" Field="OwnerId" Title="Owner" Name="Owners"

```

```

<TextField="OwnerText" IdField="Id" GroupIDField="OwnerGroupId"
ColorField="OwnerColor" AllowMultiple="true"></ScheduleResource>
</ScheduleResources>
<ScheduleEventSettings DataSource="@DataSource"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.TimelineDay"></ScheduleView>
<ScheduleView Option="View.TimelineWeek"></ScheduleView>
<ScheduleView Option="View.TimelineMonth"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
DateTime CurrentDate = new DateTime(2020, 1, 31);
SfSchedule<AppointmentData> ScheduleRef;
public string[] Resources { get; set; } = { "Hotels", "Rooms", "Owners" };
public List<ResourceData> HotelData { get; set; } = new List<ResourceData>
{
new ResourceData{ HotelText = "Hotel 1", Id = 1, HotelColor = "#f8a398" },
new ResourceData{ HotelText = "Hotel 2", Id = 2, HotelColor = "#ffaa00" }
};
public List<ResourceData> RoomData { get; set; } = new List<ResourceData>
{
new ResourceData{ RoomText = "ROOM 1", Id = 1, RoomGroupId = 1, RoomColor =
"#cb6bb2" },
new ResourceData{ RoomText = "ROOM 2", Id = 2, RoomGroupId = 2, RoomColor =
"#56ca85" }
};
public List<ResourceData> OwnersData { get; set; } = new List<ResourceData>
{
new ResourceData{ OwnerText = "Nancy", Id = 1, OwnerGroupId = 1, OwnerColor
= "#ffaa00" },
new ResourceData{ OwnerText = "Steven", Id = 2, OwnerGroupId = 2, OwnerColor
= "#f8a398" },
new ResourceData{ OwnerText = "Michael", Id = 3, OwnerGroupId = 1,
OwnerColor = "#7499e1" }
};
List<AppointmentData> DataSource = new List<AppointmentData>
{
new AppointmentData { Id = 1, Subject = "Meeting", StartTime = new
DateTime(2020, 1, 31, 9, 30, 0) , EndTime = new DateTime(2020, 1, 31, 11, 0,
0),
OwnerId = 1, RoomId = 1, HotelId = 1 }
};
private async Task OnCollapseButton()
{
await this.ScheduleRef.CollapseResourceAsync(1, "Rooms");
}
private async Task OnExpandButton()
{
await this.ScheduleRef.ExpandResourceAsync(1, "Rooms");
}
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public string Location { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
}

```

```

public string Description { get; set; }
public bool IsAllDay { get; set; }
public string RecurrenceRule { get; set; }
public string RecurrenceException { get; set; }
public Nullable<int> RecurrenceID { get; set; }
public int OwnerId { get; set; }
public int RoomId { get; set; }
public int HotelId { get; set; }
}
public class ResourceData
{
public int Id { get; set; }
public string HotelText { get; set; }
public string HotelColor { get; set; }
public string RoomText { get; set; }
public string RoomColor { get; set; }
public string OwnerText { get; set; }
public string OwnerColor { get; set; }
public int RoomGroupId { get; set; }
public int OwnerGroupId { get; set; }
}
}

```

Enable scroll option on all-day section in Blazor Scheduler Component

When there are larger number of appointments in all-day row, it is difficult to view all the appointments properly. In that case you can enable scroller option for all-day row by setting true to **EnableAllDayScroll** whereas its default value is false. When setting this property to true, individual scroller for all-day row is enabled when it reaches its maximum height on expanding.

This property is not applicable for Scheduler with height **auto**.

ASPX-CS

```

@using Syncfusion.Blazor.Schedule
<SfSchedule TValue="AppointmentData" Height="550px"
EnableAllDayScroll="true" @bind-SelectedDate="@CurrentDate">
<ScheduleEventSettings
DataSource="@generateObject()"></ScheduleEventSettings>
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
<ScheduleView Option="View.Week"></ScheduleView>
<ScheduleView Option="View.WorkWeek"></ScheduleView>
<ScheduleView Option="View.Month"></ScheduleView>
<ScheduleView Option="View.Agenda"></ScheduleView>
</ScheduleViews>
</SfSchedule>
@code{
SfSchedule<ScheduleData.RoomData> ScheduleObj;
private View CurrentView = View.Week;
public DateTime CurrentDate = new DateTime(2021, 6, 29);
public List<AppointmentData> generateObject()
{
List<AppointmentData> appData = new List<AppointmentData>(25);
for (int a = 0; a <= 25; a++)
{
appData.Add(new AppointmentData

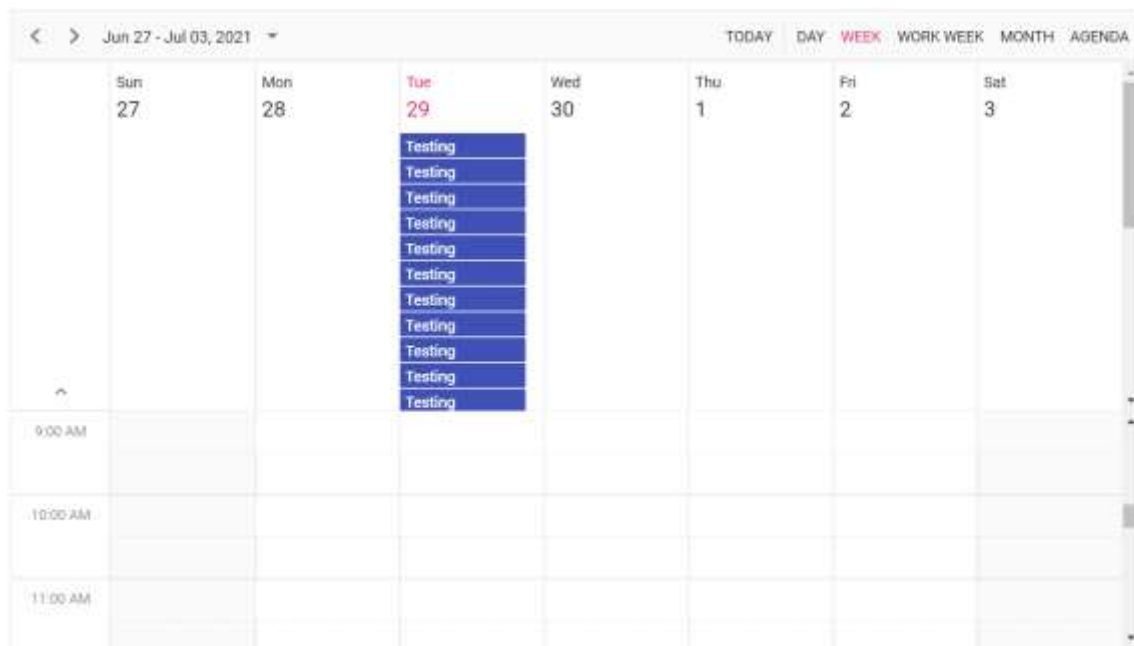
```



```

{
    Id = a + 1,
    Subject = "Testing",
    StartTime = new DateTime(2021, 6, 29, 0, 0, 0),
    EndTime = new DateTime(2021, 6, 30, 0, 0, 0),
    IsAllDay = true
});
}
return appData;
}
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string Location { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public bool IsAllDay { get; set; }
    public string RecurrenceRule { get; set; }
    public string RecurrenceException { get; set; }
    public Nullable<int> RecurrenceID { get; set; }
}
}

```



Sidebar

<!-- markdownlint-disable MD009 -->

Getting Started with Blazor Sidebar Component

This section briefly explains about how to include a **Sidebar** in the Blazor server-side application. Refer [Getting Started with Syncfusion Blazor for Server-Side in Visual Studio 2019](#) page for the introduction and configuring the common specifications.

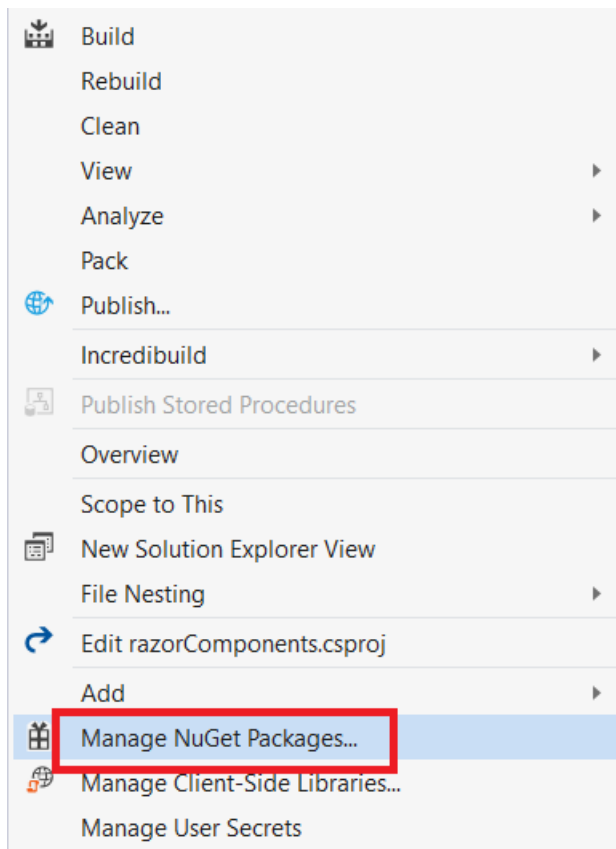
Importing Syncfusion Blazor component in the application

Any one of the below standards can be used to install the Syncfusion Blazor library in the application.

Using Syncfusion Blazor Individual NuGet Packages [New standard]

Starting with Volume 4, 2020 (v18.4.0.30) release, Syncfusion provides [individual NuGet packages](#) for the Syncfusion Blazor components. We highly recommend this new standard for your Blazor production applications. Refer to [this section](#) to know the benefits of the individual NuGet packages.



1. Install **Syncfusion.Blazor.Navigations** NuGet package to the application by using the **NuGet Package Manager**. Refer to the [Individual NuGet Packages](#) section for the available NuGet packages.



2. Search **Syncfusion.Blazor.Navigations** keyword in the Browse tab and install **Syncfusion.Blazor.Navigations** NuGet package in the application.

[Browse](#) [Installed](#) [Updates](#)

☒ Include prerelease

 **Syncfusion.Blazor.Navigations**  by Syncfusion Inc., **4.96K** downl
A package of Blazor navigation components such as Accordion, Context-...

- Once the installation process is completed, the Syncfusion Blazor Navigation package will be installed in the project.
- Add the client-side style resources through [CDN](#) or from [NuGet](#) package in the `element` of the `~/Pages/_Host.cshtml` page.

HTML

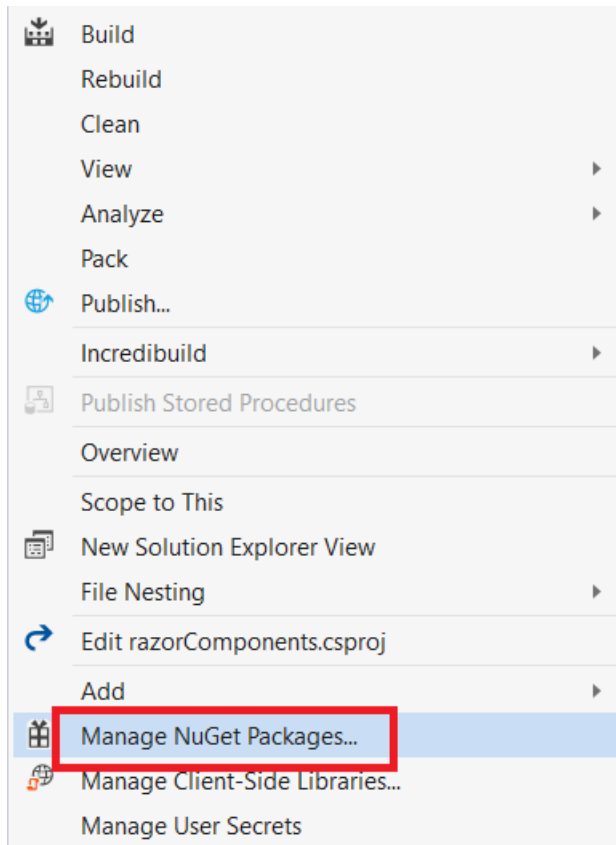
```
<head>
...
...
<link href="_content/Syncfusion.Blazor.Themes/bootstrap4.css"
rel="stylesheet" />
</head>
```

Warning: `Syncfusion.Blazor` package should not be installed along with [individual NuGet packages](#). Hence, the above `Syncfusion.Blazor.Themes` static web assets (styles) have to be added in the application.

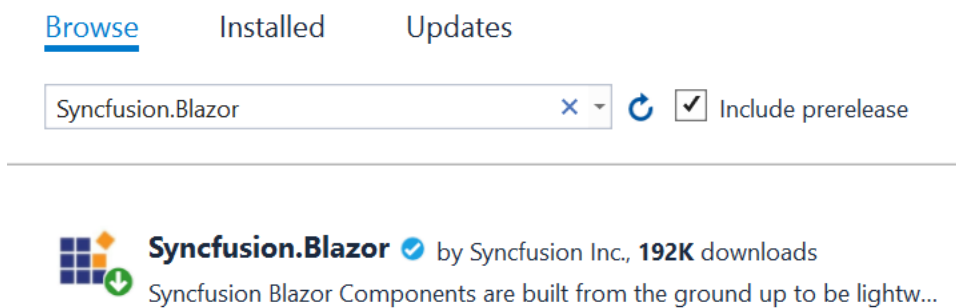
Using Syncfusion.Blazor NuGet Package [Old standard]

Warning: If you prefer the above new standard (individual NuGet packages), then skip this section. Using both old and new standards in the same application will throw ambiguous compilation errors.

- Install **Syncfusion.Blazor** NuGet package to the application by using the **NuGet Package Manager**. Right-click the project and then select **Manage NuGet Packages**.



2. Search **Syncfusion.Blazor** keyword in the Browse tab and install **Syncfusion.Blazor** NuGet package in the application.



3. Once the installation process is completed, the Syncfusion Blazor package will be installed in the project.
4. The client-side style resources can be added through [CDN](#) or from [NuGet](#) package to the element of the `~/wwwroot/index.html` page in Blazor WebAssembly app or `~/Pages/_Host.cshtml` page in Blazor Server app.

ASPX-CS

```
<head>
```

```
<link href="_content/Syncfusion.Blazor/styles/bootstrap4.css"
rel="stylesheet" />
</head>
```

ASPX-CS

```
<head>
<link href="https://cdn.syncfusion.com/blazor/{{ site.blazorversion
}}/styles/bootstrap4.css" rel="stylesheet" />
</head>
```

For Internet Explorer 11 kindly refer the polyfills. Refer the [documentation](#) for more information.

ASPX-CS

```
<head>
<link href="_content/Syncfusion.Blazor/styles/bootstrap4.css"
rel="stylesheet" />
<script
src="https://github.com/Daddoon/Blazor.Polyfill/releases/download/3.0.1/blaz
or.polyfill.min.js"></script>
</head>
```

Add Syncfusion Blazor service in Startup.cs (Server-side application)

Open the **Startup.cs** file and add services required by Syncfusion components using `services.AddSyncfusionBlazor()` method. Add this method in the **ConfigureServices** function as follows.

CSHARP

```
using Syncfusion.Blazor;
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void ConfigureServices(IServiceCollection services)
        {
            ....
            ....
            services.AddSyncfusionBlazor();
        }
    }
}
```

Add Syncfusion Blazor service in Program.cs (Client-side application)

Open the **Program.cs** file and add services required by Syncfusion components using `builder.Services.AddSyncfusionBlazor()` method. Add this method in the **Main** function as follows.

CSHARP

```
namespace BlazorApplication
{
```

```
public class Program
{
    ....
    ....
    public static async Task Main(string[] args)
    {
        ....
        ....
        builder.Services.AddSyncfusionBlazor();
    }
}
}
```

Adding component namespace to the application

Open `~/_Imports.razor` file and import the `Syncfusion.Blazor.Navigations` namespace.

ASPX-CS

```
@using Syncfusion.Blazor
@using Syncfusion.Blazor.Navigations
```

Adding component to the Application

Now, add the Syncfusion Blazor Sidebar component in any web page (razor) in the `Pages` folder. For example, the Sidebar component is added in the `~/Pages/Index.razor` page.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<div id="header" style="height:45px;text-align:
center;color:white;background-color:midnightblue;font-size:1.2rem;line-
height:45px;">
Header
</div>
<SfSidebar Width="250px">
<ChildContent>
<div style="text-align: center;" class="text-content"> Sidebar </div>
</ChildContent>
</SfSidebar>
<div class="text-content" style="text-align: center;">Main content</div>
<style>
.e-sidebar {
background-color: #f8f8f8;
color: black;
}
.text-content {
font-size: 1.5rem;
padding: 3rem;
}
.main > div {
padding: 0px !important;
}
</style>
```

Run the application

After successful compilation of the application, simply press **F5** to run the application.



Enable backdrop

Enabling the **ShowBackdrop** in the Sidebar component will prevent the main content from user interactions.

Here, the DOM elements will not get changed. It only closes the main content by covering with a black backdrop overlay and focuses the Sidebar in the screen.

The following example shows a Sidebar component with enabled backdrop.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
@using Syncfusion.Blazor.Buttons
<div id="header" style="height:45px;text-align:
center;color:white;background-color:midnightblue;font-size:1.2rem;line-
height:45px;">
Header
</div>
<SfSidebar @ref="sidebarObj" ShowBackdrop="true" Width="250px" @bind-
IsOpen="SidebarToggle">
<ChildContent>
<div style="text-align: center;" class="text-content">
<span>Sidebar</span>
<span>
<SfButton @onclick="Close" CssClass="e-btn close-btn">Close
Sidebar</SfButton>
</span>
</div>
</ChildContent>
</SfSidebar>
<div class="text-content" style="text-align: center;">Main content</div>
@code{
SfSidebar sidebarObj;
public bool SidebarToggle = false;
public void Close()
```

```

{
  SidebarToggle = false;
}
}
<style>
.e-sidebar {
background-color: #f8f8f8;
color: black;
}
.text-content {
font-size: 1.5rem;
padding: 3rem;
}
.main > div {
padding: 0px !important;
}
</style>

```

Note Class can be added to the sidebar element using `HtmlAttributes` property. In that class, you have to add new styles or override existing styles of sidebar element.



Animate

Animation transitions can be set while expanding or collapsing the Sidebar using the `Animate` property. By default, `Animate` property is set to true. `EnableRTL` will display the sidebar in the right-to-left direction.

ASPX-CS

```

@using Syncfusion.Blazor.Navigations
@using Syncfusion.Blazor.Buttons
<div id="header" style="height:45px;text-align:
center;color:white;background-color:midnightblue;font-size:1.2rem;line-
height:45px;">
Header
</div>
<SfSidebar @ref="sidebarObj" Animate="false" EnableRtl="true" Width="250px"
@bind-IsOpen="SidebarToggle">

```



```

<ChildContent>
<div style="text-align: center;" class="text-content">
<span>Sidebar</span>
<span>
<SfButton @onclick="Close" CssClass="e-btn close-btn">Close
Sidebar</SfButton>
</span>
</div>
</ChildContent>
</SfSidebar>
<div class="text-content" style="text-align: center;">Main content</div>
@code{
SfSidebar sidebarObj;
public bool SidebarToggle = false;
public void Close()
{
SidebarToggle = false;
}
}
<style>
.e-sidebar {
background-color: #f8f8f8;
color: black;
}
.text-content {
font-size: 1.5rem;
padding: 3rem;
}
.main > div {
padding: 0px !important;
}
</style>

```



Navigation with Sidebar

Any HTML element can be placed in the Sidebar content area. Sidebar supports all types of HTML structures like `TreeView`, `ListView`, etc.

In the following example, the Sidebar is rendered with Accordion component in its content for navigation between the pages in a portal.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<div id="head">
<SfToolbar>
<ToolbarItems>
<ToolbarItem PrefixIcon="e-tbar-menu-icon tb-icons" TooltipText="Menu"
OnClick="Toggle"></ToolbarItem>
<ToolbarItem Align="@ItemAlign.Center">
<Template>
<div class="e-folder">
<div class="e-folder-name">Employee Portal</div>
</div>
</Template>
</ToolbarItem>
<ToolbarItem PrefixIcon="e-tbar-search-icon e-icons"
Align="@ItemAlign.Right">
</ToolbarItem>
<ToolbarItem PrefixIcon="e-tbar-settings-icon e-icons"
Align="@ItemAlign.Right">
</ToolbarItem>
</ToolbarItems>
</SfToolbar>
</div>
<SfSidebar @ref="sidebarObj" Width="280px" @bind-IsOpen="SidebarToggle">
<ChildContent>
<span class="e-avatar e-avatar-large e-avatar-circle image"></span>
<div class="text" style="width: 100%;font-size: 30px;text-align:
center;">John </div>
<SfAccordion>
<AccordionItems>
<AccordionItem Header="Dashboard" IconCss="e-dashboard e-icons"
Expanded="true">
<ContentTemplate>
<ul>
<li><span class="e-icons e-content-icon">Sales Dashboard</span> </li>
<li><span class="e-icons e-content-icon">Analysis Dashboard</span> </li>
</ul>
</ContentTemplate>
</AccordionItem>
<AccordionItem Header="Profile" IconCss="e-profile e-icons" Expanded="true">
<ContentTemplate>
<ul>
<li><span class="e-icons e-content-icon">Attendance</span> </li>
<li><span class="e-icons e-content-icon">Perfomrance</span> </li>
<li><span class="e-icons e-content-icon">Team mates</span></li>
</ul>
</ContentTemplate>
</AccordionItem>
<AccordionItem Header="Achievements" IconCss="e-achievements e-icons">
<ContentTemplate>
<ul>
<li><span class="e-icons e-content-icon">LeaderBoard</span> </li>
<li><span class="e-icons e-content-icon">Awards</span> </li>
```

```

</ul>
</ContentTemplate>
</AccordionItem>
<AccordionItem Header="Scheduler" IconCss="e-scheduler e-icons">
<ContentTemplate>
<ul>
<li><span class="e-icons e-content-icon">Calendar</span> </li>
<li><span class="e-icons e-content-icon">Meetings</span> </li>
</ul>
</ContentTemplate>
</AccordionItem>
</AccordionItems>
</SfAccordion>
</ChildContent>
</SfSidebar>
<div class="maincontent text-content" style="text-align: center;font-size:14px;margin-top: 8%;">
<div class="loading"></div>
</div>
@code{
SfSidebar sidebarObj;
public string Target = ".maincontent";
public bool SidebarToggle = false;
public void Toggle()
{
SidebarToggle = !SidebarToggle;
}
}
<style>
/* Sidebar styles */
.e-sidebar {
background-color: #f5f5f5;
color: black;
}
li {
list-style-type: none !important;
}
.text-content {
font-size: 30px;
padding: 3rem;
}
.e-tbar-search-icon:before {
content: "\ec0d";
color: white;
font-size: 16px;
}
.e-tbar-settings-icon:before {
content: "\e679";
color: white;
font-size: 16px;
}
.e-tbar-menu-icon:before {
content: "\e718";
color: white;
font-size: 16px;
}
.e-toolbar .e-toolbar-items, .e-toolbar .e-toolbar-item .e-tbar-btn.e-btn {

```

```
background-color: midnightblue;
color: white;
font-size: 16px;
}
.main > div {
padding: 0px !important;
}
.e-avatar.image {
background-image:
url (data:image/png;base64,iVBORw0KGgoAAAANSUHEUgAAAOsAAADSCAYAAACrfzewAAAAGX
RfWHRTb2Z0d2FyZQBZBG9iZSBjBjWFNjZVJlYWR5ccllPAAAFN0RVh0QXUxUVGFuAElzb2xhdGVkIH
BvcnRyYXWl0IG9mIGEgc2VuaW9yIGV4ZWNldG12ZSBidXNpbmVzc21hbi4gQ2hlZXJmdWwgYW5kIG
luIGEgc3VpdC7YIXYmAAAAJnRfWHRTb2Z0d2FyZQBZBG9iZSBjBjWFNjZVJlYWR5ccllPAAAFN0RVh0QXUxUVGFuAElzb2xhdGVkIH
NvbRk9EtwAAABCaVRYdERlcl2NyaXB0aW9uAAAAAABJc29sYXRlZCBwb3J0cmFpdCBvZiBhIHNlbm
lvciBleGVjdXRpdmgYnVzaW5lc3NtYW4uIENoZWVYznVsIGFuZCBpb3BhIHNlaXQuG5vJrQAAAC
ppVFh0Q29weXJpZ2h0AAAAAABZdXJpIEFyY3VycyAtIHd3dy5hcmN1cnMuY29tjYy53wAAxORJRE
FUEnrsvWewbelZHvisnHY++dwcOqnVaoSEAjBGgAnjYYCZweAahiL9wFUD2MZgKOWCe2Zcw9gMxg
YbUzZDsLHBQyhgsggokQIAlo4DUjVrqdDvcePLOe+W15n3etVt2+ceUC3e3umF9Xbvfufvsvfba53
zP9zxvNuq6Rrvala7X/jLbH0G72tWctV3talcl1nalqWVru9rVrtfostsfwet//euf/hFrVlr2lq
tk5FsY5klhDkfdzywsZ4gir7fbPx5snH+8MxxWsMzTKi8mG8PNpRP6+dULVloP4+tkGa03+PW3fv
s9v+iWqC8cPv/cl73wiU/81VWSRrbluGmeWWVZCz4r9Hq+W8vXnu3Btg2YrptZvosqy7B54Wo5HZ
+Vq/k46Tjhklvnzr3HRPX+MPRF9DvRvS/6H74xbX/KLVjb9Sdc/+Zf/Ij1/Mc/9uhslnxfGLif7b
pucDw+dc8mc3d7YxOWYWIleLMdC6ZpwqqAGiUs25UvSmyPeihXM3hhhDjO4Tk2DMtGgaS0a6Osqq
K0LKd0HTNZLWdZFvEPB93ev/zq7/h7R+1PvwVru/4Lloc+/H77Z//pD/3PdRL/n57v9E7OZSh1Sx
etW4f3cHg0wf7mNgadEP3NEZZZhelsiZOTY5hGiXPb55GXGRzLRZEU0QkDlGkM0wJ8LxT2HaAoE6
TTEwF4A96oM4JhxHCKKFue3YtNw7pnWob3BBuj3/r67/6RVfsbacHarv9sffgPf8++eeuZr/vQe9
/zf9y+eXvTjXzXyEvkRYnBYIA8WWI42ERvNMT07BiOgM+PQjzzzA3klQlHmHU47AoATQgLwzZ8hF
EAlyxQxAsBaYlKAFpmcwGzjSxZwXMDRL48xzNRipRO5DGY5Oc6TvJ64YfBBzf3tr7tm777R59vf0
MtWP/Mr996968Yq/nZl0zOTv/vxz/84c2zwwOXj7uWhVT+2ALC85t9kbcGMLi4deuOsKawZxCIBv
aAskLgBiKJEWx7XWTy70E3hCeANMoci9VU7FhHgN1B2OkjyRIki6Uo5QqOAHXCoYIaLM2kBYJQt
+GI7I66nSRLqKkV6tFURTV8wf9b/7uf/CvT9rfWAvWP5uOo/e+e5THq1+6c+Pxt92782Jw7/Ypxt
OJMJ+JUAC2zDPsbW0L+wmCBHyT8RJFXWE2n2MigOt3hsKqOWZT+bq3harI000KYJM5dgcRhhsD1I
aNZJkIqMVu9S1h0UjYdSHS14BjBwLICvHsFKV83w88vZYjBnCvG8Dz5DAQDZ2mcWna/tlscvK33/
rn/9t/+uXf8Deq9rfXgVXPzPrYRz/4RR9+36//i9u3bm5mcWpVZYHx6Ri2EwiQRiGKA/YHI4RWDS
/sQhg007RAXaWoCnLYVL5AvfunqAUhg1DkcXCsmN+BKMqsLW5K7/tU15vwPc7CsrlfIHnJqH5nH
IjyPPmiAY95GL70u5dLhoQd8XWzfNcZLQPL4iEied6WMhjmdxP3rtjW/5kq/8pu+Ztr/FFqx/qt
edO3eMs+Pb3/sf3vsb33X75vP946MjQHggLwJESu1NU9hwq+dib28PCwFQlHswwhPffPxxRfGekd
it81WCeLkSoIYiiyt5TY2e760/tYXQ8UQd58hFLhcC6t0rF+UXX8ibJOpw4nvIqSASuxK8L5HIe4
sUR5GVgu8Khchhy/Yb21fY1nUFqHJQiBxGQACH/btllbzrW/7OP3+m/Y22YP1TuT74/vdak+NbP3
t468aXH9+5FxdZ0cU8I10KcSg2IwE27EbY6Hdw9cIegY2qtuEpt3F4eCD/PsDO/g6sukQyW2A137
OEdbMsRjcIOfeJmWM7IOlj9EUOd4cjlEmGxWys4R0C1xWQylmA+emRvO8KGxsBCAIfSfW+kiTICi
iAGQI6Oz0Fd8tIDoCT8ZnYtiV2RFqbjovAj46LMv7ab/uBf/Vb7W+2BeufqvW+d/+8/exHP/Bviz
T7wul06s5nMRIDWE1XsvltAVIqQ0lgy9DH/o6AbDlFZlJ6is0osvbewQlWi5XYoUNk8YJOW5wsC1
y8chmm18X4bAbHzNGJQlhwgKDDekspYF0o6x4eHiL0QtKbQdisucrdQbeHwOBvCLV7vpilWY3pYg
xH0GxafFwAXvtIxcaldC6tnrD5HFY5g2tW6Pch0yJP/u6lhx78wa/6th9sN9bLvNrc4E/D+uV/9r
fnj//er/9aEhdfGC+Wrspt+W92OsWm2IxxeHc5gC73QDnRskIvgWT2UgiSTcHQ3SCPjqOgeleH
HYgVPuYmcmuLKzjVqY06pzbA47MIsKlgCxSsXGnAkTJkuVwmNh6K6dw1xOUMY5fGHGOL9hsTwTgK
7RnJYiKxeLcdwS9kkwvCeLXZrkSHwCvRDD0hWCM05Hnrkqjy/j/Eyx3w+76+W6ffdfPKZb21/y
2zvu7XL/2T7zUObh/886OT0/9178Ildyw2qmU5uH37NnZ2dsBc3jxOcFlkb9coR0JAAUsta8Frm
KT21hkwI2PfxRXLl+HMrLYmmVZYpVl8AMHfgenz7dDH8V0KaBbwVtpJS2zBUOh8q6tUht23BFBp
cCtFM4j0NoMIAnMjgWcPtBBE/YOC9ihCLFJyeHYtv6Io+XqAwHruNjLvfaGXaxXKWYjcewhH8tZF
M/8r7ue3/83f+u/Y23YH3drp/5gb/2bXeev/F/RZ1hEKc5RqOR2JATFTldz0UpduS1/U34hjBeX8
ArEpRgnh68iKC/JRI0Q1IaOLl1E51OB4v5ClMBo8e4ab/LMCsWkymIEDCtELPQyoAjadnqPJYbN
gu0niJnQsXRPrWmsW0WCdJGKu5fG8hoLXghC4WSazx2aIy0RM2zeT1k/EMK7Gpnd5Q2DyF7ZL1DR
hVAKpAM6WF3LII6Sw97gTul3zHP/6Vj7a/9Rasr7v1Cz/4Xe88Ojn+Ddu0+nZnW9hrrOGY5exEWB
DCc20MzRRYnqK/exm9rQuoRbbmZYJsfIqUVqP8unKng/jkSMBXCMYdJAJ2U8B5cniMoNeH5Xo4vn
ML4c55sStdYVFHbN0U2eQqMQB3tE3QFyJpA/ibIOzvPC+MLsxtmcyKgMNDQ+znMl1hfHQPlimyV+
```

6NDirX9zDPKzx14wW5Virfc9DphogEvEYyR6/bxYRxXLF7DbM+PH/5ypu/8W/92L32t9+C9XWzfu
vnfyS69dTtT7uGs5+kS8zTEq7Yqb2OLwVa5bRpi+SdHmISAC2/dBnwDIKWGJPLmZTmLaDdDERgA
oMRPaaixmqA0xsCTSVJnRFAZO4wyuHADHLZyHQlgx3DqPrBIgynMcuX4p7Li4ewtm2EVvexepHB
R5vILlR4DnYHVOR94jgdfroBTbmSA3CxtLYW9KbUME74gauHs8x8npBLEOjwWgEfQ8/+09ea0cBh
sdLKczUDmUdfHkm9761jd+xbf872W7C/7rVlvP+iqsxz7wW8bzj/3Rv7r44Ft34tUU5ZmFnrPCKO
xjcbFsdA33dQz17AwA+wdfU6HEskpoArKS2VuxaTGcwN1NNbwrQmjGEPzn8XvdUJkuMDlH4Prm
GhPHcelWIO/+o1jE/uIhz1kYk9WgpbWmJXVt0+wkueyl1zORa5LeyYTpElZxi98bMQic0LkdoLgJ
ZeKUNCWhTnd4R8tQDkfZbzGYYdB5PjHINOhFRoObb5y2tu757DakoAh2Jrz+G43fsOb978PrnQ97
c7oWXW1/z6xR/5W18Vdjd/srfR79x86km17UKjSesbji4in96B16yw6RXYF78Bb7CDmqVubiDcGw
ICSIZjitkZzHSHtqhQ47oA2EKxONFYuFgksSnXzOtIr5+sxLzCLJVLZwe3hXkz1CJ9jUqYciiMen
YE/u4tU+RqQUzGSPIMgnfIg3CCEQoBcyLvVxnMXPJECVtik6aYno01A+o0rvHcyRhntXweOQwu7m
1hc2MoctrBcnyIzd0dVueJQqimZhC+82v/xj/8ZLsbWrC+Ztf/+2Pf3w0t95nOaHtnMj4WOZrA9T
pAivJVwFAKI9mLY+z0fexfuQg3DOBEKdidjpiRtlipuchbEcyrGLWwslmL5WqLZB3swVhNRKaWAt
YUZSHPlEQKE1ZdCJ5NsXFz1CKLM6+PSuzJ+PQqleOo19eyPSSTO5CXgJUctUjeymqYfih2Mhyr0
zp5/4IfPidLkwTmrCRLFeI5wssRMrfOprgmeMV5vK8R+95WK7L8jsLYXcDy8k9DDd3cCb2difsPO
P2oof/01/5obzdFX+y1cZZX+GVjsf/yO8MNzUbqKSXdYB8fiZMlmbXcgI7W2LklQLUC3B9YbDhSF
gt1PQ+xyjVE2xanoLUFansu7cYABHKivJDY58nw4gV+Rz2O0hsCsFoxCdyGYTbpgkQoxuZ4i+2J
SW2KB1MhF2XsAZ7YuZKjZzLUwaynvL9SLHRiCMHfX76Ip9GvW68FmFU2RwTEsUcYSe3GN3OMBo2M
GVvQGud13YogyeffJJ9KNQO1WsFiKr9y6px/jc3r4cPuG1Ms2/s90RLVhfk+vHvvsbznmo/eVVX1
jt43taKxp6wllCXmWmYoL8Rujg+hserCC2nz/agu14InfkxUUsvFYIS4o0rRMBrkjWKhc6qEOhe
WqUmOahufBEoAJKQprT2Dbtti6kT7OsI0v1+9Ycp3kCH45FZBtwjYF9FYJR97DdD2Erqmmqel3YE
MeFzC7AlB6kQMBX7Qxgiff0+1SFnJY+AriUAA+6AS4thXijf1A5HqMZz75DHs+yb2UiMcn8IOeSG
a5pgvXtK2/9ms/9Xd77c5oHUYvuSU25U+ledFnIgFBNUh3MWESRBoLeJboVctcvnI//H4Ig8n0jM
uInDQFpBXrT41cm5fQQuXMAWFLCHxtV4C61JTAmrm/NDdLQ0HtkCXzXK4lwObrBMS0HyHXoeMoE/
uxni6EhU1hvExACg3JQK5ZCWsWsbyf2MqGIE9lB3KemJpWSHlci8laiQSuRGJTGttC3V71y73K/Q
5WuLgMNP57eDbBnedfxOXrl1EIq2bxbRFh7Q2O6lpgPktns78u7fku7O1qb9TWzfuYH/so5q8YTlh
32g96GsCq0NG167y56dgHz+Bb2t4VZ98/DEtlrslesSAMcSFjIZmhFw5jQSxSalaSvy91TJH0PAZ8
nFRPqqnSngMcoMyHIGRRUOtWYROkes9nE5mWrIhZdi9pNoZwUYa12Z5aTySpieor3GVQumJwpw5S
hQ4Av1CmAfrML4tby2lu8xRMSvKwEjw0m8x8BPDw9nODZaYY0GOHBN7+J8kCv74dk/wD+oI8yjs
+KqnzDV3zL3zlsd0nLrK+JdXJw+I+2hoNOZ3QBoe+iPxhgeXyErelduOMbYhNa2Ni9ADsUVhWNqE
3OBBYlRXesAECgZQhQDAGwQFclL5PsYUQKHnWzgs3RRDZXAR9Kvi+otnNbGbAWu9X0+7D6DgqCUP
5YYgObAnRT2VKuWMDaTG4I+HKXa+kFpofZIG9XCggUANyJlesJ+OkpFlnN1xpySKgniXfgOvpewf
4edvJCClVLEUZHj8PY97F0813wexojzBPXcYGO3frcX/YS89MvaXdlarJ/29ZN/9lu3+73hu4L+hi
UbE4FrwREPHARis4rM7FQxts+ff6AGmjVklKayZmWufx3WEBAmUoi6HcGn/Ftsv9oLUfm+gEtA7H
S0zpVAMBwxAl15vrBuzdhL2TRuqI1KXhVBFdvSccluwDdoSZYEJS0bcnkdrJrwzSTIBTGLIWhqy
xpUh3lIDDRXI9MSi92KcAV9Ov78F540LhRFz35TPsbATYF4D6LB7JCVEGNZDEVW9pvDo1VYi3OJu
/8zZ/74VG7U1qwggbkive3N3bP9/cvPygEmMIT5skWZ+pQclZ3NN3Pls3NGlFLAFFTnQqXGgI0g7
8RT4AjwDJstUgVSHC6AsqRPsdcWy61R4AK61UNqGsBDsiMfQBF44bIXDZzq+X71KH8Q2Zk/FbQA1
MAV1apdjHUmS1/arBgoAETpa5RFs21RGqbGkiCSuZK5DL/TcDy0LDls7iiHgYC2K2e3NP0DMvpBK
vJjJfSRA16ssNBB4PNrb5p5H+93SktWD+t690/84/twc6FrnxunrNsIbnQ66on15weIhAjt1fOEf
R7YnJ6zJ3V0lKrQjTht3Ewyc4WxBI8jKca31BAE6ASzoLYjaYtXwtwa2FRZV7ap0QD/yi45VdK0N
PCIVtbdeMgkgOjoowVaV3R6SRAZtqiAbmOMDtByTOgzku9B5OHAtMUbxltlcnjTcqilQirrhVAEc
9hk43lXlU2y2dxuxGG53ex3XWxPLqr3ul0voRnNbW0jhwYtutYnuN/83v+zY867Y5pwfppWxvnLr
yj0+v3ev0NWLKRQ5G+hoDGrzP4WGqPI2PdYJvikiGYxtVnqfMGakbTiRqguvw70McMka6Qxwlcyl
gmStQG2a+Rz2yEZkCAKzRNc5ZgZ6vRau1E4kHAfk1ltVJ7txagM1m/2QQ8JFiA7kd9jcwBtkSil7
V8WWnnQ8OUuyV4S3keHU9kbWHxms+130sZuFBZ7Pc66I8iDJwK+Wyq12S9LWPGeTxDulqyI2PPd9
wvbnDfCSGP2xL2+KHhYDNgFg+9pI5RCEgt7frQwQpelFMWZCsWst+GCG117hDUBr3CAko6cEoBzb
2WwzmcFCGaUyPjiY6hoQlVYZaZGlb7V+Vrhp+kb8FxAQU5EAgI9ZialbCpHuiACpnwui0SwulbW
tea30dvt7UeSONjXW0oCoqfd9aAMsC9Yr3k2R6/YZVLA2PNeV9bbGP+xsjAavI5vEh/LCnmVvsb8
rnVfJ3d7gRuJH3D574o982213TgvXV9wAfH/a7o+1rQbenhd11LbadSMfAyoVRPQGqsKtnNqAmm4
n0VVChcQqZTkcB0yTU2vq4fs9ogEmgK0hoW9aNHG6+znVUBsGiMVO6hGxbmY7JF0rcL0Xo8lgek
ewe2oHN3HcQmtDK7Gv2aStkteoM0k90mulyrARHVelraGbgmTMNWscUVQHVakqgemU1NVsrjba7M
MRFo5PTxD4PWRZhpVIZ1gNGwuyt7PVpHU0tWB99Vc8H39mEERBJxoIOir4VY7AMRHYIoPZQdsyPr
Xx2Smf07ZkpQxbjdrCsw6BYSg4XwIZ1qAkge4THKoXV/txrJilNUURAI4dTwpsBuQV2hs2Mpskh
74n1E17KhMKNetm01krMM9q7kmL5C9NQGCTi86pGxrfb/yGmTK0IYAj9cqCepCDos0lvuJ14xeqa
3qeo7Yrj7M2a12QaSctuU+yjRHltCodYLV8eFfbHdOC9ZXfYn99tW+z96hArAshueH2io09AQArt
PI3DJXCYs1CzLGabih2qYwbPXoaviFErYicA11RL0EXgXOmlGV5YoliAg4JkjUpjqflHmtJiREJl

RAkpUNfulpD2G1QfOkcTQR5ML6ajeT8TWkBGVbvrdKbcNoDhM6ruhBZvIGBULROJd4v0ym4PcJaH
aciMR2HUUi6SfHIipcxNOpsHil3RuLUqzyweZ3fOKP3tdK4Rasr946075reJ735fKlVScr3cRhZw
g7n8EUhqnzpoMgQaVdFFTumuqlJRAuxl/wYberT04ndT+JEAlVHYdpF36UgyV8peRE9dQ8CurEm
SOoY4ieq8ImJee91K8p3Yow7MGMDmdQpY6r0x6mwk6YUvDfukxq3mN5mAUDYBp09a13i/Zm9dgE7
ZPZVihie8y55htT1lWF4UuilN6hi0NNVX5StvXpKup2LPRttjNG+0OasH6qq0iy70iSXYWkFUiB2
3DhJPNEHAijZH/RyeSbTTZPwy1MGlBbVPm/UIDQqqSSV5qz1oKguIlcIosrcwmtklAc8wjK27Izp
ocQVlbedSZ2KNWA36zXjun5NoEi3qfq7W0thvblgLZahi7YUo6raz1tWqVxsru4CzmJmuJtb08/9
JcC4VluiJyzspZK3mnELAYOMQ85ADV5ARRt6OebpbxOXJwFEXmlvHqz7c7qAXrq7aSxfScYlmu9k
haNhUziM90gBQTE4yy8ciqA9hsmJIxU7iupht+yImEdSIElvFWeQ3jsJoYoQRpKssRiASuemjXDi
eVu+vnGcwYliHDiPlCwlzWqGQatrcx5q19bVYD7SZDdOXg8LUkA4lNTOatOE4+zPx63WYiaM9tM
WLANIxm3RCxnMlvESlsJ4mIBfWtjSVXDUMBPyrU7FjG7bOqCjKoxMyDCzP/452B7VgfrWpftfzIp
679MjSk2pysJPTlJSxjga0WRnRNNZJC5SMTGiwBCBMvn/Ju+v6Dduqu7T+VAhGAURnEUFIKqNmD
UOpKoJq9DDTHCpRln+aOxV0wih3SOYZ6yJ/PQAe2IjBx6cwNUqHzIuJbTGbUv2Gi4VhMq4ZFuLXR
ZtPUT4t0Ump9lcraV2VjRFBHR65bG8R7nOZ6518p0t7+vJvSzv3YRjQ9VEXhkaexUD9vITf/Rbbr
uJwRc+KsuwrG8QYFiZ2GH0mTqVAIB5vEYjdzWsQucoQUMcsi5VHUmNTdowot1U0NTW2hkETbiv1b
FTaWyWWK/Z/rPKm+R9dscvVvL3StMJa/U6C+DIcGQvpW6Ra0md7UXq6GIc1BBpTNO4oDOoND/FwB
btZ+YRC5Dp0YXGbK1PsXe5znRSu5bOK3muzZTJ21LWZ5UODxXapBrSIft6tlbzyLPg+hFWB3cEzK
16uhnOqaoksAznUruLwRc+4uvujSeNLfk9LJvZKulcom2YLuh+aexTw197c82mVQodRbRJC4+bt
nKvipl68aGVKcQ7UWm81EGE1TKsk2aYS0AZJKEMixDKWTL1NbnGWnWJOUzhtRshklpxQwL2w1bh1
dpqZ187TAJQ2t110UANIDu+Pp3k45IoMr7LP01k8xc07gMzSEMyGuTrGw298CDgYkPhj5JmZODrB
jCYuzVNZKml/FsLBJajpA6J7Bd27X+p3YntWB9xZdIUlfKMSdkHWhewLZMIRij8cASTHbjjKFdSH
tVAVYk+ph+f52s/5INWZ1Nzi7DIE3ywh/8NamDiUCHvctMJ5XguSbyqylKFhQprY237UiZdelabv
KJ2YG/Shsl0O/R2K7M8VVpvrZ12X0CmkFlwFDGD2xP7Xix2ycUKapBwuUbUuVw2RbDql17Jf2M6
W1bbl6X37kwRv04cn3e5ubyNMklkLwszjL52y3T9OvandSC9ZUHq2GOhMmsNjK3LVPKRO01xkHJM
SrBEmljnxYx27aXXnMV0eTFnurTUqJ62hZmtqWpWfALgJRTYMC2VYspyhCf5Wk4JIEDN2aprqlF
E7mDFfn2jsYGEHxgCawSQ2ZiWHRbW2eUu5Tu2u7Vxqv9qT8hgrg5jLbAnbsrKGLTNkUR4y6oFWT7
LTyHyGd6gaNO1Q3tdzGo+3gNsKOVd8ZmykBqxENnUuh8HnymeKZ6dbhzefTnrd1IL1FV3ZYnqeYF
WPbZV8KjzCmk9DM/6aPWiRkVTiWio91Q6k04ZJCmRJ2pKaYG+qnDTVO7xOOeSvqjIUDC95fJWVNF
W+asrejCZv2HyJicmM9OZSlq6lq8ZCCVQy7TrNkV5bdVQXlnqj69JYJ1W4TfjHbhq1NfFXgjjU8I
geKIzhOmzillGvth4YPIyYSMEujrnYqQWPUZazUyNTFPQOV5iSr5zyOEfSGKOWX87M23vr/s9pOES
+HIziPH7XsiJnwM1aMUGoyhU0ZDiTcVS7cb5odhO9qMxU0s4MrvZWein2SQ+S5jBgbfcxPMN6VQ
KRQc46bzzARiNbNUKI3SJqPR+0rpXhHM16YjFBqsUBRenltSlRKVsrbAWpif2FAKfUZP66CdFo65
ZQJS/BzkJ3UCbDbNif5Rq2SOWyKxvLwMoesi0PKUfYNpHPHjj6+XnPbmeoDi92jQhMt1UtG5ubTj
OND6s97srzH5YHj9od1YL1lQNrEn+B2HaWxYJuTxgOK3W+MGZJSaqbds2CBWFIQMrm18JzGJ9K1l
eQsf0oXE020JpU2I3+0TSiJsUQTJ4n65aZAoisquEWdJwUBiyz5RqclpbSEZBk0Mq15br8dybPyb
SHcLUUiaAeqh06lgrt6vFVB5YqcjYcd5tECDqEBOzFmMybKK3Q0UJ0ercdX0M2prMuHpAP7Hcj8O
fC/GRTrrcSW3VxOkVanilrfKuPdLkQUeFacpB8vrzsd9od1crgV2x5fu+trbq00EmcYQ9KxsawFE
Ysa7VVFWSUwKX1qr99U4e6NtNYtSJAEE5RYFv1upBcilNldSjXJmrFUErokiXUo8zCclbBYF0XS1
nNVk2amlg39rCpCzWKYF5LvgsQVrlyrKsdy3qJkPK1PziTAHMeDDrYQ3K26CjrKgxZ01GEWjvKK
2X5WHEVjGwtPcwM6L4M9AmbFmsMWLTD9YhI7F72U5G1ILjGo39zkwnuXfbCVyzqlqPcAvWV26dHd
0zirroMRGCzh21F1/qu1LVTsfBsvG+qqwLk+WpAoW5v3ipaoa/inYdiWS4CkaDwGbnhrJxHOMrtY
i8eMkNrYkQBogB5slz+bDV2MNQgIrNGi81zsm/tZpGy+AsBWnVNBNOzkT1uSBw7REju9QR5jfk+
t5Oh19xenri0Rk7FwnAegBxCkAAsraWDOWJ0zq+drxnX5oxXVbXK7J+LET9nTqHak8ZPhmPMbi4K
4w61I/blnmW8vFuN2TrQx+5Q68dBWb3KjciMgX6kDRDgoq/2x1TAWPxjJL/ZvNy7BmWbKWxe+tQz
0F7UOzSejRdisEOD8G4ZI8sblREY2hDHJuJq7JAcAJ8Rp1hMfMxtbsswFXEzoL0j0HVHQK52hms
rfnk4pn/UgYcyTt8NuDgQfDwL5LEhjppRq4j/7AFfGTADSasJ/IPLb4oHjSpXpSp1khkhWU0v4RM
v3+slnIkBztSMHRz6fKtt63R6y1UzuXJSI77EYT/4dw3J/btQZJu22ash6si/DMGzXDSzanzYT3A
R+SolOEdtRR1sYjrdWS97WHR7oSOLXzBCYAqzds42dWRNUgHZ2GxSDavMVCuFoaLzUe71xlMag
SXZpPpJKynLM3aUnY/ZOyV9aaLWIBYi+yVA0RYMRMwywQ0cTYXhjXfKpXKSNPpWJ4jKkktUtlJq
tcg0OZ0yTXgVSDblfn6iC2lKldd4HR5gZck5ed6RS1xQZb7HUjs29o6bBm8ZdHvUaAeqrxV8su9H
Ot5MDobG/KT6tAPDmBv3/ZEuZn6/8WrC1YX/5VFrnLlLsYSVRisslYSYnom+rosbyuhlCgLhu7CW
y81OmB7LquD9XqmlgAxDk4haWF6FYQ0dOsQXvZ/JjMI4o4RxAMVUqyNtaWa1B2lvFK5a1LFrcF9s
uJ9ksqrabrQ5aIdF0ZmowwFxxk7m08wn+dIqwwH4xiJAGdje4iT8aqp6mlyIgVkp2q/eqICpnIftm
Ogw5k7dDQvc+3R1OtH8DbPw+oN4ct9c65rJeA0OyOOYW88yFpXFMh9HiCZL+RzGRi84e2IV4keaJ
X8DBzXs0Tmb8obn7Q7qwXry76E1VzLkU0mm1NDI1kmgDWbroFoql5MS192eSBzaaKC/akYKN2t1K
nL20/KxpVNHKca5shXidqRSTLDwYtPYTLZgMOTol4HYTfC5tYAvh2h2+/D6W1pb19z7SjSxmUOD5
C5SmqtHU2bTvse5WQ8x9FkiWVSyiZ2LJP1T+MEzz5+oxnzMezplLh0MsVKZPQyXuhn7XZC9OT9S2

ToC1qHnQCx3C/tDh85g7d3EXWvi/zG02RJ0XjGqjacYCCffakTA5gZxaJ8ThyIx2fwN3bgyL2uZv
Jcz2cPxCvyVk+206sF6ysigzURQNjNZEd9sq2wX5EXWnzdrE0bZ5M2JmPNKctiaMMx1U4Yc3nvac
SzKWYiNU8Pj7Caz4TxTB2pOBaQTsczYUxj7SUEy2dUYTkvBLCZJsJHhaH5tq7PeTRJk6SfTAT4S6
TCYqvVSuRuLNeLFYAHkxnGIonnAtClvGcih8Y8yTTmy8T+6WyFwAtF4u6iPLmneEaPD8YmAu0Ya1v
AcE6f5HMNFhq1BB16wRMhEiJvPwjvx0Nu9Xw6bObCaoHR7cmDNm/TE7iZ8+R111tjW3gDx4gyb1x
/B6uwudq49BMdzrdVifmWw3e6rFqyvDLUKeURi12Vqb6qzlxOgWG5WNYVp9OpWdlOramj4w0A500
V8dg/L6S1Op1PN5Bkfc1CFUeeUibWw3ZTslyOJM8wF2AS/bwuHloyXGvLcTPNzbWsBMxIQhR3UIn
NZgcM4KgS4qdiOs9MJYgF+KrbqTK6fiEzPmQIO9uMwdJskjL6tDMs2NFrlw3sUBTAA7iJOx7FtbG
pHCEfsS0e+x1GT9I+dnjXy24ieg51X2Lp+CQiOkN19CoM9YVd5/2D/GhtUwZR7s8UsiDrb8nkLuc
9Zk6lFZ5njsuuEnBfB9XZTtWB9hcBaq6OHDpQqrnU6m5GSOTz508Q4aZvSqcTeuTZHIC4XGN+5JW
x6ojL0+J6wVlrg5OQUnqYGLgJSHO5Xs/v4sJmD/2oox30tZKGQ48FQHkuzCmsuOJknDJGZxRpYb
obDpW1a8hrzo4EjKUALlF2ZQ0piw0sDrwSGR3tXEMitvHZbKnJ+Cfsti/XFVscS2HHOGXGLdi8LM
czS5WwA7mnMrCRPCCHKYdlPsaLz91BRx6L5dCxPvkk9q/dj5Obt9DfP490lWrROxVFBbohN3NuRn8
cMG/uXkCzmWo7HvGLbGTBt8f52U7VgfaV0cFFkiwaMstnjxQJdOxT+kc3HjCQmEWifX081cTqf4u
zoFpYnx8JKRxbPV7LBBU6Dq9i6782wukOsUgY0bPSFPV2xM/N0jMPju5hMzmAkY3gmsCm24IYDLX
C3aKPSZmXTNDPXRpmNIQltibzeOtqKnZ009AMwo7G1gPIevfDlQvFxy8gOLmJTWHAY7Ehu2KzRk
EH/cEW0igUoMVYZobYrY1cPj6+I4Qd4treeRRdExlt4403IGbnxukN+FmBXhDg71NPCXhDnN2Qx4
Z9bRPDn81SrnFUP4XcDzA6fwm7IrXPjy4iGm6xXM9yveBKu6lasL4iK0+WhSPSRkie5UTO0jZL1x
0MtTsg27LQuUTPatBFLbL36Pzt2KP7UISPYHBtgB7bm8wmqJdz3eyXxQ4U+02YT+y89EDAOkFWTo
XNKB0DLMVWPFfmeH5a4HF5j1HXwedfuKSJ9jXzbPFSiqPO9ITF+Cnn2pQihS+/HdNqAwOxg0flGO
XiAM+c3MY4zfHg7gZCkcEnYuN+UJj/9ixXNg7ks+wJa/ZdA5c2R5gIo946Eds5CuCi9K3E3u7K6w
YPvg3Vncdg9/ZkZ62QUXX0NpEGfbznY0/gbuxiIrat60V448UNXBiew87VK0w11AL5+e0XMbn3/A
OBXby3+8BbvqbzwGe3XuEWrc/fKqoqNZjpwC77XtPYOxMpWKLpZ2TVTQVLofm8JordR5BcP4f0+B
jORGSvudLG22RRAPQJDunJCeaFoQkUs6NPYDE90CwfpgyG8pytQQ9bTo7NbqkzVO/OxfzmY8/jfx
xuyHtOkZSZSu7Tsznu3BKpvTQQ5yEWIp27nUuwHntWbOIEx8sbmManKIX579/oafjp6dLDjXyIrT
d9AW598P0aj+24LqyQ1T1T1IsxNgNH5HGCu/Jnqz/A9sZQExs4KPr441/AU3Kf3XMDvPDIdfy5N7
0di9fEYx84gDXcxAOBHTx6PsS+vUCnOMP4yEMlMrxaiH3+O7+Brb0dq//AQ59XHT71vnz/DW90uo
N2gHALlpdnOY6fpnlckrWyLIcJgOTXcc6xGWWTulczK0jsVOciph+/DUekb3ZYIGzkIPAYB7W1Rn
R2PEHOV9k+bh3ew3N3nsfB+J4AusR2x8aWa+FonuLm4S3sjHrYcCs4wpoDsVex8Vn43RcNbI0KDD
s50vEBbj3xhDbT5jmxEFt0FW7BfPEOsnlktkdoHiOxmnhvq7NSjg2MY2/t45Iu+Ek+9530K5rPjA4
xGmxiJjB1GJjblALkl9/3MeIyt7gC2SNk747nOn8V8Iffgaa6x5UYimWtRA66a9E66wLd8yaN4aF
+uYy40i+pgUiLa3MfSk48jETvY9jsCXrmfN70Ny1s3rdW//dlz09uXd/Hg599rd1kLlpdlhb1+kS
1XVS7yLrc8jM+mOJ1VwmZTXNvqoLs9QM3EiH6EFz/6BK7v7ONQgOjVBVzZ1H5oYT4/FWDDQ3d3H8
8f3MNx7OCJpTDPuYexHFzCH3zofbjujHBVWPsNvQCh2IcLsSGZ7ueyaRrHVayW20/XuN84g+dswz
q3iSvWFYyFwe/eOsdTZ6fq2InLFL36DGFYohd5uPnCsXbqj4IeLm8NsXH2HL50v4M7k1v47x+6oF
I+EIBdoLi2MkQ7QwHgAHoxq2dyKO32e/DlQAo9B0ZPbNTJDVzs2qJ+r+K/ecsurg6PEJoB3vLgFe
TLiXrZ9hyuEcoBkc6Pcf6Bh5G+8DRGPbGVr+7DvfaQHGWVxvPKrQ8PH8CDaMHagvVlWk99NDTcns
MxetPFFNG5K8JaTyN46DPxcx96G1/8lj+nntRCGGd8+z3IbUd+6Ck6viFArZFMD/HxDzwOY7CFTw
qYB8Ju7/iiz8T4Y0+iDCO8cPsZ7cHkuzYGwnCDgYvnp3GqtapbcLH7wEUEYpQ+WXTx9PNPC+AeRX
+4o6V6jLcyq5FvVudkhheOb2K+NUC/zvGOTcZUHVy/dg6RPdfsiZ3Mip3JPVyyClzZCZENe4hnYy
yWmXZVZLXmpfv2MZlM8PzhDBd7G8LkHbh+E5IaOz4eLi5wKWDpmygEeV3yiTvW7nsjSq+ZrmfJe7
HBW0dYebqKEQ766Fy8iqCcodjaB2PW8XyGjGM25rPddoO1YH1ZVvK7P2sElf07qVeNpqsMvuPizk
d+H1tXrmAmGzpNc9wbN/WfTJjPaUuKXRIJ9C3ZAlTk6+FTT2EpYM7PZrgiLLw5yNA7vYF3DGycrM
7wxjfdD1MYbtPJ0C/nqOIFNq5fUvA7jOnINv29aA8fv2Xi8+T13kggPBxqHrHthQorAh05Rab+CS
KxP48Xxyqd06SCJzLcDeWPYWmyvuYwRoGmMZrqpqnRMwjQ7zSzXk3PwjKvLOzHr58Xv5fY1NsUL
fTxUoOqxOziyvRXGzfpah/Tw6ACHZvqBlVLGa3NnsOXodAzD7JAtrJnrRO3t7mH7kBf1MNxFcuH
/zp58SBk6s8uDuon11LVhflhX/h3cH5ei+a9H1RzHQzCQLfdn8L7z4rNijAT6n62D45K8qkxTTY9
zrj+DmE5Wt1sMqlkJbqDx4/3lcf/RBFISd2bx0UpliuS7wwIbIZJHqYapCcnKmsc+krNEfMR4JeK
GHjrCRFRfoZRMMSL3xJZkY21hJW2twpEVkbDmngDlMfLM9QJv2e+iW2dwHRscleO4dlMVIzYjUx
SZHGK5x00nYm3ulojsLcQs7UQR+j0Bdlwh6m4iEhmCJyXYWclYojDkc7KiBvSOrxP4bWf69mLilI
Iy7Mv9LuGEBryNfZHyHty6xJ33fwid7Q6yowOxYz8BI8nK8tkbk3aXtWB9WZZLOX56fGIVexN0e5
vy0zQw39lFdHaCbZGsf58qYQNAwGryNfjRQhv4MEyCh1NweT1Rz7vnSIhM4HEGL1r15vUEKwFpa
eZeb3zCZxxLJs3FJtYANpxhF1XOPDKjzYRRCF6Yn/et7HC/rnz8DmegiEjZtpb2gcCrudjtDPC9W
kMV2zqw2xXmBroOsKRnUBsxBqWa8AS8FtkQO3OxHs0tWIOy4MIxnGFIzPVDHVm4+KVq+jKa+kFj9
0ap/EAhtiknuOhTur+swTBbhMyLUSGW1GK0mC8NZf3CmDOz+S+Au2cOH3qcRSzFSphcSOXn9V4LC
pA3vRsPG13WQvWl+eHd/Fh+DdvohBpateJbNQBzvwOels+/MCCmy00kZ3eXo5n3JXNfiIseWlv0H
S6FxbLxgeoPAGK4SEf39MZMLUobopQNni+FnbJahz81UypMZIuFyG0Pvf0L6G6NhKfS3DnNcC47xs

bOm3SQspbX6cRzAesq0bK0TuhjuNVlBBYfk8eMkbyHyPTOSGSuXbHQR2R7LHZmL0jmK/NmZAcrcK
 pEUwvnpfy09jdw/tKGfD5fy+gKjPDMc9wxB7hcPYt40kd/fwvp6QGi8xe1lM7QwQNNQzUrt1E7Ic
 KN3aZzYjLD9IXnYmV92azYmc1gZSWsRVzKoXO73WUtWF+eH96VhxLz5LakL9qui0JAQjB4fiQbf4
 auKSCzI5GRbPxt4tq2jX9/08B9ZK3VVDve22GgA51SsV/ZuCzN6YRZN9L2u/ILEqbhFPJViQc3ha
 1LDwqjMtTR9O61BEyf+Pg43PP78GhvcleT+yJJJI2X8600sVYz6ixzRqDnSHuf/EuPhwHeJu8dz
 pdwBKG9ARMjny/9G1kSYokaRqjMR+YhQns4v/g9esIhbltK9D2oXmeYWLyeCzpyVDegu+XWMgZsZ
 ov0B31sDp4HsH2xaaQnjN9BH12d4RaDqEqzeHJAVfIYZUDPYtoXx6Xa5bjI2gz1EVA2ptbbf00Fq
 wvkwe2E8sh0Nd6A1dyX7cQDfsYsquDeMJzEEkjFk2c0+NQDvgbw9KHB7dw+7WpnZVEEMNX1/E6o
 ls2js3dVxGJXYmnVJMgmAxdzgQ2/DiNa2qIRuB/Xjp6RXJ+JEXZriwuofw8qPCcIzrCtNRZgoVsh
 yt4GS5JOG0Ns0bdgSg+xfPY/X0s/hDAV1/Y4i9wMM5odNQOjQqcSQUKdomriZzegMdmA8rV08ty
 wwSStMk0qLyn1h8Ac7MXyxlzNRAOGFPcyfvYFglIfDuTcim+2o36RcyqED19DGBDpVz3wbz/7m72
 Jx4yb23vYQZETRIONDNYhYfLNB1K60MbsH6MoH12mdU9R/84sTI51umMVCHTudssCUdR0zzY4tQ9g
 dlz1zKRWHCR0YV/t2H5niHF2FnNITNdilig3rdCPbVh2Qzs7ugdVlE7GmrI7PSRY6hoPVMtN5gq
 efu6eS8z4/xe6jblbGYvhEJ70pMdc6zY1AZZZTWQodL9LwiSPseUGk6sZqAScfC7v1ODo1scybae
 dBs070zwbCBGrBabsboYB6MwxNYQ8pxBwLpAL6HLmRbMuNuAUO2HhyEMitnF0/aqYCNNmADSbr9
 GJxCHR+rmA2Z0jfOyXfw2jqIfOhkjszU1NIRfnS7jb3aX9xrel7S5rwfqyrWpr76fN8dn3iVZ1Kf
 MisfWG5y6IJJ6thw8TuEvtGazeVQHsX/icz8Ann76Nm2J/Ojpg2YcrdqvNWTRJ3fRVElnMqpoVHS
 0CqtUilecdyIEQYCBg+IzdHrxrQ1STU7gXrmsT7Vokb80xi+xcyJGTHGRFZ1mskrjmYCpT7N+mRV
 rQF4ntNp0VPWG7QGxQTiq36NU2bJ3BqvfjCbgEiIa2NC2UvdnvqdBWp34zFMu3tJjclnmrCfqXL2
 J170SbrzlH3WwAPimEJntVYGmZVaiEJ7/9x9kByZtSTq7dwZL2Np65FFURyex9+De37ff/qVtqm
 EL1pdlv1ftXftl6/mPfu402XY6MqNIzdQAt2daMfQ/KfBQfvtPMfmHvIast4c33bTWzZ7zueoJ4xT
 zjxjHETqRfTykGqGIB5Aq4vAvTibRZGethIUzLpmz21j47LCgw68rRmalf1p1Zzr0ShMR6maWjj
 p7HEcbtbfIwGKhgEhq1sizeF1tYLZjodTmNAEDeclxlb42cjOpGDLWExlNFQ/BzG6MANqKAY6ldP
 xAB2+5vUhjwdFoQz+zNmBjK1WWx9GxJKbDcP8CbnMuLAvz2Z0x1dGXpRP5vx1+6Rf9aLu7WrC+vG
 ANR88ahrWs07QDL9E+vsbiRMAMiOMQKHb5s5qJ4+r8EWZkmVvtsDIIm0Lahlk53S2HTESNAqhke4R
 yaMtF5p0KDja3qNmPGTXPdeziKYASBtjhl3l6Dj+vQnAS2ALNYLJWltT8vbVx+e6VT2/Q+TauZM2
 NHbL3iK6C1owOnCfs3tZ2oSZCGcgJkTdm0mwFesrXXTB9ggzUW13O8pcmp7sLIZcGEh03tVSHRGk
 wIceU+zVI+p9itBgvo5SDZevghDPe2UuWwCsup3nc9nUz9q1vfFH7/9ay6n+22h6t/5XLuf/tWd
 Xb+3kx2sp6KZLw9l1UH/ODOGes7ioaliBI2biMQBObtnIjtd0s7abPYVCCKePqDBpt5s0xkLTz+L
 XdyFH9ZdEeLZtr0mnF2ThcBAV7ObGpt/YgZo9gdidkCCeLRbJmYjuWWE6Wme/mWJwsMTmeYX6cIM
 1K9f4W6VIHV7FdKJPqmR31BHKPlMasrxd14IZdzWKiDVRrZPV143J5X07Os0yOkPQa+cx/y+fuN1
 ScKcfSumC/ZB4sHB1C21nY14s6Gqu15Bt2JZ/g8Pj3d3/191svcMusr8wq9q79gHVy8+uFLvo0FR
 0OiloPniJgtVkaB0ax99K68712zGdjNRalW0xBKLUMjW27CUIzWaLkOAO0M2pMlbmcxZqqjaiN/N
 nfl3FQM2qGMuusCzSjLGyRq/Ikjlwkk8bab6lpAcoSOqMwkBrC/rNaMQ0+JoeA64QobFEBSaLynd
 evyfh1MxNWO6DK95rPxj+Vxk8rAazl2zrysRawVnQ4sWRQHtfG56IULB4yPLzYT1ns9er2Ewi2t5
 AcnegIDj+wY7GZv7fdUS1YX7HlfeHX3ivf/eM3jTp8xKinzUQ3LTivmtmplalymLFGbn72C2ZohX
 ZbrcoIKUlp6+WNjHXksdxtQLCYK0NWHJDMhmhNRbvI744w7fr1PbcZjkzUCfNW9LaaJ1UJGdsRR1
 wucqzo4BHbNHItBSJtz/ksQV6YSOM5VrMU/X4fgdiaRtjXnsRk89rINNeYrF3UTb/jeix2p9Uuub
 MvMbOS7DDSaXVM+E91uBWdR/J3p6efm/HaSmRwwFCX1YwP4TBlhnKMvESwNRzb++c+2e6oFqyv6B
 IW/HbL9H7ddMLA6ggjOgPhyEQZkDm3ovcayUqpqr313fUQqEJZ7iXGIXVpy1IBncpiAnU1bebMzJ
 aaQkj70GKYRWg309MJpHyuHSKMSNiMtmQuh4HfgxNm6NC5wyFZ906aWaucx7PuJ+xpN8MUZehjsU
 qRrk7QTSt0uyV8eR/baIYn6ygPsxmgVRrNjFieKzo9wG4SHTzuUFmVcreeJ804S68r99pMtuMB4b
 ihmGN2d4DECzlHo5HElpV5w41/En3dd7W2agvWV3jtXv+D6vhwZgyGgQG/abxdxJoLwM6Ghs5hNd
 DMaLQa+ViYG6KnjhU4XjMdnC3PBETS2sAWMGW+1ILuarHiLBiYgQtLgG732ICJ6XrCfMteZ9MwNG
 S4XbmGXC+ZC/MJcMXuNISd2cp0Y3OEhQCJANQhzWJ7up6Dwe4xX8ZwfUdb0uTHRYjjLnrM3SXABF
 g6LZ1eZgEm7VV6rgu09vCa7ohWVw6G/khYukQxm8D2LA33uBv7cs8R8ulr49gSBne60+rB5mHRcy
 p03vIQcOt07O1v/3S7kVoH0yu+3Ic/pxDW+3kj9ErTjdnUODUvJyGi053SYVnNk66o85ghGyln+of
 uGg62WC2FFAec7VPOUKQLzQ+Oz44xP7qL5ewMmQdTddDZgBL5OPTeilsxuV50+pgCTaQ1Moget6W
 ln5nzkZ7VqBjgHnb4eHKz4qVJmFTXDr7SpeNn0b0rTFMcHhgz708MyLxLDL+6Zi86bTGcrpVA4SAW
 uWofFncRiCzkKaDtDmFEHjtyTYQYKSFyYfBr/luOIIIsb+BHKmOPaQjJeoTybYe/Aq9r/0XUvv0X
 cdtDupBeur42ga7PyQYdoLTcaPVzrvRhMcmGZHFspWzShFsiY9NXwsjZs5q3HwtOscT1AJOMvTI6
 SHAtB7dzG5dU87UJSbe/C297R1p87LqW0dKFW76xEczHwk6ARYFduOTg+RxxRW2fT8He2i/8g7YY
 3OoRApnIlNmxyzOjF5lRYsak7vSEGu+cRyvM5TUSNJ+vGq9xblCZN32f8nUzOLZaZ20uGd0Vm5
 U2LI+p2pZDyG0KGKr/ZKvZbBqno0MCPaDiYALmzOvv/IT3yLtaCdZK4FdnWbtXbhrjk1112/2C8r
 W21WYttWNDM38Vwpa1FQDKtmKr0pYk63L3r0TqLgWUqx1WIoGXxyeYjWcqxTcfeip62+d0irgOZR
 YQIWwyLDQ1cbVUOWpw7o58XVJCL5Y6bIq9fAuGXs5fxM1nnsBTf/hB5GdzjmyGhmZdB1HgaMIDJT
 kPA0cksLtpa9eGUTg91/v3mTZJNpc3pfeayRls006x2Vu/LwB1xIYVZhYV4G+dk2tEak6xYwWURT

yDHYpMnp2KvTrUOTwW84djJo14ZXHv9m+3O6gF66snhbfO181zj32fs3vpx+145bJYnF5fJvgVOh3OUGeNWTsaDrU4cqI2NTGhzthNotA84ZUW6+TWgbJaIdJ199HPQm9zGy4nzjFUY11aLbN85lnMDk/k2hX8Xket54Mg0ASIjJU886X2/c2FacutLfzhz/0Cnvj4k2KjhsKitpi8JkKR0z0mLuRVA3q3FDmcaNcITkpnG9XSNRfi7Fbu3dMKI68pNGAiBkM469k9JUdMLqZajRMQyMVSDyp6wz1+s16NYW+e059DpZ0oKrm202QtXXvDrN1BLVhf3dXf/ZXy6PYP+5bv6kRxOn4KS0FaMRRC2VilmhhPkLHxdpU2iQ4Zm4idHmIQAdyenIn918PmlevocHiTG+gcV7phmZpneyxk34MxTuA6poAkRlwniP1U27JkHM8xnYv8zVEIkj/8yMxcOVtiMBhpGxdWAWw+p3nC9OzaRjNc2aY5zeOF7xXnmpzBEY5UCYncH+W0XedNaZ8oAospjQJmhoepYSmR2ffFPd9WuG7pYqmCYPqj2qy00+Xf6c0b6F57s5gMObyHPrPtCtGC9dVd/oNvGeeHd4/qwOubS3pNLa0nJTA9o1TAMXSjfYSZMFA1YyB18PlyhuV0ipOTI3Ua9fqbcJ0I1jCfzXS+SoBR1M28VWExc3q3r11DIoCg04cxy3i5QC7Pz8wAK7vEqLzhxbunmC0SOJHLEK7WizqsgtFQiiVxXdzScogcHS+L4Hz2/bP3CUIuR0qKFwVGPguZqKXao2Kn0eJtWM87Dci3tvZROJzocS6lU2tamqwXqJmPjCsbUeardFDlSJH3+BP2LCYrb99hT7rjdPS1YX/VVV9UPVfPZP6yN2tXYDaUiZSXdoDoVWbiLxmwdq+lmFiwxW6m3d3pyCLPRTTaQbS1C6fbXyphi/wUoLDjAutCSzKfsDZ/gT4HFQvgMgGCKaYQjMXWFbt1zAlyixiWsOuAs3fkPmwBaEivrNyPbcjrWFjOBH69nihoo5HtTGxwupFOWXcZM64ypNkcUa8PsARuNtMyPibr04ucZKkeHlU81zpx2uFkf4P9z0vmcVlN6qTbRWky66pGfiz2sMh/78rlXIzzZbtzWrc++mv73K8Wt2/9gOmLFI4Nraqp8xqVU2lCacFRrhbqISWbFfInk38nB3eRC2iCfh/h7h7c7gAFvb0cSOVxspujXSAsyliGajqOxmNtYWmT81jeh1UvQbSBPKNHnkPkM62wGeWhA5J5eFgNcFgl5Fh+Y0vbOi2HiVdasE6mpKfX7ERNJQ5TjjNHvc+2zThyprIXzIQi4A1XJxXf8xmHS2naYSWHTMaJ7PJ8x4/U1taWMyKtaevqrXSGpbu391P+mz6najdOC9ZXn1kHo1NjffBaIiM5zMnmXhMT4q612yDDLbpvlWHKjiZCUDrmRo5gdxuelxNAWxq7LAQAddiFLWwLJuszv5gN10T2klmNym3qZjkSUMQyJWflcvK4B3e4gVQYsBTblI4mvq81X2dychAODUFprG1WhxPb7abLIUNACYsI0hQZmTvy4dk8RAY6MU/eRrOgSo3KMrEilAoexmETud+e2NgGv8N8ZsNXe5whHAGmJn5oGqMA2B90F2K/Hjnn7vv+dte0YP20LLMsU9uoy4Ld8ikNBQiGxbAI61YzTc4vNUmiaiaha36w7OMo0H5IpidSV+y9Mk3XsjrTPk1kThZ71/K8S1WwXInOHZbilam8r8jitNBfqtPpiFqew83JeGJ7Bn6TlMFqGKvSQnFmRDX2sshweU1Bxk9SdQCvq1JAncMRWT3EAEa/B3aw4THDgvs6pjIw9L5rMrcw8+LouOlkeLGr8pa9FR3a6MyBplHOYnj+RwugjBeu7XyhZXqPu9cfbTtCtGD99Czn4n1V8cxjL9iev2VoGVmm7EfHkEnGZDG42HmpSEJH5KjWmwroXGEwm8Op2KGQ4yFXM7FVV7CrEVJhLEcAZlshHJeF447GTg3H0DmwRVo3Sf0qW8U2je14bHHaEfvyTy7WjflEmyp7MPVbQCCOpI0mTG2pNB8xYa5pX2lmicujpNRAL40cG/WG12qEFkyQEuI7XsDBsOTzsEPPjAylKrzj7lWm08pnp4XatTtOJnwcUkzlKdbQ9cfWnf/WD7W5pwfrpZ9cs/S7ZnL9qWG7HsDJN6SOWCAybalITIRJULCw3WJ8awe4ahw+zmyxWzojEndl6DsnhITpRD4HYsEGXrUqZnFCKpGU7ULPpxCDgZVaQl3iW2BFC7Ea2jMmcJSxh23ReiN3YUxnLipksTrDKCsTpAmnOFjKV2KQCULMFfDtUaxxOxrzNsbW1hlPqa1B/KNVnUngjLG7Sbyc5yEHn9jkppyuBON9TKIFNsZ0NsbM6o1WWUTWE975O4z+PH2l3SgvU1slbLhlu+ZLzz1M6VV/cW6rrcctxn9aAlT5ZwdozNThRvLWEDWhckSuHAorLXQFiMVsGM+PsFK/n373olm3HKEsBA3Rtto9sRO1JsQKvHXksCUPnDZIXaaxxGZE1OX6/EB10tDgWIIlPzBeYLNjKxzT4WVq86GAwFiJdCjJ8WGZ2GmMYpnj89xsHxArP5Kcapgy/5rIcUrI4nhwnL54Qxo6ivziIW09tBI+8LUQLWxlBbrFpBoF5iOrxMNMkjU4vhIWZGi0au87TtCdYc9bWx/Hd8cY13fPH/Wv7CT3yngcljgtP7yIALy8csWwvF2Y2QtphlK3ZmvzbYTcFkaIlM4bEZ11053ji5m08fyTgmczxwPnL+ILPCTDY3UNV6zyCR2X7j1NkN1Lk2xkWz90Re9NFmiwRn4iMXsaabeRe20G0swXz2ZXyr3sIht3F/I9eQHAXgrPXQ3ryJLztEV587Em8MH8OU0tk9+YQb9jfx7bYvyFvn3J+jisHDSNJ7EAhDOxubMIMhlgcH6nHmb2d7KijBxMPD04XaOp11/Kbxex0PNV1C9YWrK+ttfVv3xyPf+mHf0mMue80TcMqs1odPSazeMh069rQkvaosGLkulgsBGjLMwHnFIssE/nZFes0QSSAGXsZ5ls2+u96A6Ktq7B7EbBziqB/AFx6GEHnN4Dum4QtrwsA72Lx4feizjuIPu9z5aCYXxnuj0USTxFeuAqsDmGPhsJ1GZxdOWD07aK6cQuF58CSbbz13FW8661vFhtWGFQOEudkNmO+BqU6i9vZiSLsw032MD44ahL5ffZKtpVJKfWZtWXJZ2LqpeX6KfKOWFalURbPtbuJBetrbwW999bJ9NthltztYpsyJ9bS+la2gCnNwh1P7CpoCnN51YNVcqIpfaMl1/Cm7hYujy7gD288i7ldaIlosPcgTAG7WU8FZQPU58loW3AuPyigGGpD8HrQR3jfpvZfsgUoVTGFI+Xa113JonB7cxdpvJCDI0Nad5GerfDsnRPkYtO+c+savvCdb2H/M400aQIhm4S7dEhxOPRcJHegSRvMelochWhYiJ7l1l8BCRU9RbD9sCmNq9ZdG02q/7iUz9oyawvW196yRjuPlXfPsqrMAw6jYq3rhkzIdIyhm3w9zU0Y1dYduCKNu72BMQ3VydHzj3G/gO6tb3sb3vd778WLn3wak887w9AWW1eAbmjskoQl0jU4L7K6gyqeok6OVH26gw02Y2QSGSGNTvclj1SZpZeWjWMWlNnrzXmtili2sJgv8d+ffjIcfvV/nrlYc5CorGBoV29RGVqSK3rISxhQG9kebGj/OfqfaBYIeb0PYFSKFbcfXUJDDVqZMgjAt7WoodkBMoh7bFK0F62vQM9zfOinvPZ2I7dhnLybNv03ipq6TCQ1q7FlNJ4XA0eJtJ2VCBcckLhHu7WnR+ODcRXz1xT0cfOLjOH7/h7F6wwznxd5kIn+8OIPvbYtE9chcwpZLLZE7eu4Ii3S0jQvssTTF3VtneOYjjYMRUH3gw49jLvbjWZzhdLbC5197AF92/iIuXxDbVkiSBeeszElYe8tQkhwuji1MWwdqBwcCVFokbzIdw2JfKXmWE4o60HpbR/OH7bC7zpAktSVpxZBQOiuTsBe306MF62tudR58W3X65Ad+zSqyr2/KWZo5NGyKzRpUzRIQhmSFDgGsM1HZZSEvBIAdTQv02GdYNr+/u43LnbCjF+E2Vh+9gV/5Zx/BrePb+N2P/TG+6LM+E32hQHZxsDoOjm6dIk6b0ZJMUCiF0a1BoOx3xjrZeYxYpPdbd67h7Q9s4OpGD9GmfL/LXN8YvshwV1gx4XARAVwqrMpxj+wr7AQBJCjUuGk8mau3moX22hJVC6E52i

domsSZTeoiEyOYC21U9rxz9TOKdme0YH1NrtJxftTMkq+Wndwx2amfgNRSULMJgaBxwtSGLSQbIX
dEPjJBggkEiKUZdMGHiSp0YPUjWA/fD+d4jK/54i+hKYhv/bIMY7E35+M5Tg8PcDA9QrB1DlbX1i
7+bFa2XCzQ292BmVWYHIy1CdqOAK7fZcoiG57J+9mlpho6TNLQ8r4ErhiambagkXtgGiJFvC9yfb
gjH8fC8vAGDLfWuCqnPLtiSxPclM10NnFejo69y5byGValle2/u90RLVhfu3br5rnHi8WTM7MsO9
zs2ktXc4Q1i1b7MhWsVmFFTr7SdMDUbRInfKcPU0dYWDo3lV4ads13RK5WwmpmUsDd3UTn8nmUSw
H8coVqGevfiTAouzSwf5QTiZ0q148zkctDEcpAJLknwHMc1AJS2s0JJ+DJPUWugFFb78vbMbaqec
SlTlPn6cARIBymFZ+NUcaplsnZ4VAY14PlN9U6BC8/pvaYSuU+GumfWZ77q+2O+BOYU+2P4NVZG2
//Clav/nCV5V1VEJCC69KAlh0WoK062Ux/hYytYEwmFHjodMQmZdaPNgevYXFwa5ao46iSF9QDsS
EHYiM67CAhNq9IWHpGwxgKaHY68K+M4OyJDdwXrPcEkNYKRij25tBGR74fDH24HVuvRUeTxmRtAw
FL6nh4lA0QWUREG03DbtrZHEPJ0M1CJLj2N2ber5XCftuavYH80XP09a1lL5oWNk07cwW3Y+106
IF62t7ed3/R/AYM7OHPXdlDmfQdrxWMZKjImHQQ1wl2lzbYX9dW9OG5TkVHMfWJAObjdHkefU60V
/n4EQCrshRwFqBgMph1/2ZgCSGZco1OwLskQ3n/ADouYEwcQ9mYAujeppemFF+e00jttD1VHbzQM
gydraoNZHDrhmWiTU7yRqMtNCC826yeKLT7Uz543qBhm84RoMtS0s2iZPrM5xDtrUdLzPDb1to3o
L1tb22v/Ivn5YmPlSXRcmJbjpZrSyVgUw6hdTZRG/pUkEBOoQYiyXLsoMhG4LbrHSPVS43MhrajO
X/5n+sqimNVNjTQRHK9TsicbeFBUcbKic95L0QmSuMGGeYJcwNFqAyk0oAlSyX8IU5Heb9iq2qvY
VZT6uhobSxrWXHdEYDHWS1ODtBlsYqb3Patr2Bpt9iRwjKZr6ecWRhVnbKYKG6oPUFb+ty6lxbdb
bX/qg88C+bxqJgg9+yeJug7HMUuerqtQVRssFuBwKRWBwMBvjlgpLPqdqHFOskjHZe4nsygnlCR
NOqTg6kY4oAdBqyuZlLPZuwMgBVblRNk4rxmYEYE1NaqVOIPZSos1p+27zb8vRWTv5mii78jAivQ
j+xjn1MBcpWXWlsWE9bPPEjtZ0NPNnGCWB1Zfqt5pOsccl40tz/t77S5omfV1sc59zV+/UVrmh9
jEs+nlywSJpkO/Dq1imxcOhCpi7divzcYcdg401IbUWlM0HR8oMy2+RgMzzexUoWyRlmbT17cWex
E+CpteZUP/TWcWPbSOWTThFU41px0qB0dHGNejr5e9f9nBgo3KOR6jqjVGypAPuyi6oxFysW+Xxw
eqAkW0XhY2tYJIDp9SR1hW6zBokaSITw4ap5TvZ6bX+f12F7Rgfd2swht8o6jDKQFCgBbsx183VT
n0vNZlgnl1UA+xSVkcZ3XUItYDp7g0nimv09gmU/tWsQ5etpl7W4zV5nXDDgJhaNadmVKnGccYwd
HZrOyH5OioClvY07VqHT/Jo+ClwnEinmlSjTxXx1IndMXsZmZVB9PD280IR86BZQeKMNQGanC7n9
pabG6erZZN429hYcvxExvotPZqC9bXz7r4l/7q7aTGT6aruFQ7TtgP8YNU4qUtG3K0ARFPNXZOL
ZtNCEQPteyNQNKC9brl7r7Nz2TOAnApBMnMGMagQDlMUQ4AV64ZOGYcKbdsNUOpbdL+VBuU3uiV9g
emU8lY92simCljNSvYcfQxltz5e5fV9kw5NYCOqSyBx0HJzHlmgZ0vz+X95StlQNfbXIiMdsJeKS
rhX1rho0211IL19bWMnYt/U/jqbiHykhPmGqer0TiO6FRyfbFdE80aYuMxHZdhNB38tVG4sCA7It
JbS7nMVEWdbM7u99MDnbdqiU3rmqXYlFuzlU5ZN2/GSLJhGiedCytYoLE3GjRSlycFZalj6uGh82
z41vJ8X2Qu7ytZzLCcnCBdTQXHHiw5QPzNfU05JODz+QsP3BvLAQ22MGV1jm3Gtt/98fy334L1db
eufvr3pEU0+LqirBzkMDpwco2dOslcVdvTLMRkvhQZnGqyPEMijNEarq0Ns3Vqui0SU4BBWUy7ki
BkbrHFLoLshh8GClrbE861LTiaumgrGF35XjDoIOxEwRLHWTcco+EJO7LToQ5FFnnMonPWqDqDbR
hBD/HZobI5y99Ycuf3N2AyrLT2VzKlMJmKvOyGMANWXnjxElZ32h/8y1YX5er+8jn/17qBL+dsC
WhIcxXZMKmqRaec9mWr4271+kMJUcuknXdZmL6us+3sh+7ShC4zBxifakd9eH2N8W+HALqumKPRg
ooP/TlBx9BtwfPERDKIEHINR2CWSSzF/jakVCTITlKQyichfBsOeN1xfbtDKUCix26nCmT0/b1tU
m4pymGdqejj6/mM42txie3NanCFLv4/2PvTcAlu8oy0W/tvf/5zKdOTakKmRgCkSAiaZWHAQUFAe
mrtdtKK2lxtPjXro/bV1kdoIV4VVGj1Ua/QTM0cpjATQgIJYUyKJIQKJkknQ+nzqkz/ePea627vm
GtvfZ/Dmr3xRTR8yflVNU5p/5h7/VN7/d+79doTMMfvfL/Trfv+raxPiwfe5/4Njst7PnZgUuHcZ
VjTvlSy+T+hPuqacJ9SiRRYJrKKxwT3veKBkoq/4gP5xzdspQiaEZgUpNS5rRTB9Pan6/TCkZcPJ
XVsN2Skgg7uNBAGbyghfTJNXDwL+YsZRCq+0cwOwCzbrSeJ0zSuscCbQmnDHXSdYfZVv6a2ehcH
X0xtJJ2iSA0jO4+kPrYvpLXzvwS9t3fdtYH7aPR7/wZb1R1nxOXphlT9o3oxEZpUXF4YQGcQDv6k
SN9EQ05eGIWj40C4sc+VxTO4ZuKpJ4h4YYR0i2V40agVLUB8UaFQ0YSfYu3a6jogPupUkYqMJVjd
i/xaiNNS/WyVmzxX3XqRlqIY1667T6g96rqlGPFxfDZigKPUhB0e3B+vH7Xbo+Ipe03PSOafnBB4
7WV7r5lc98+r+Z377r28b6sH1c/h//4K680fntwpg+K/TzomUSGMmpFmdwSCzITRB/1ERWHFh3Ka
hy6S22gdCoaboFzfXxx1BNEHuztszczpaiCaHhLokVhkulPeG0kypcy9cLZ6UCWRwGpV1BN2miT8u
HQpbgbpxbpvTVCFEahcQz7mKGvznwEq8cegP7qKoFZCFTRPhz3HF+77S6UML3Isvq1L3nJT2+nw9
vG+vB9tGZ2vFVD9uHcKk00RFTihxGlwvqgYSTFNgmSHLIUICNihIuMnQnaU8OoqWJFGsS8HgNTaT
Q/jIIInuFVWYrIOOaGEzyY4rqISwfb/Xzuou2I01vlUnJU75+ElKW/ytWjQYuu5VFV48uwfraGh
E5cMkwlRnCqkULPRf6zJ07A8vFD1HLC94FRPO1MwZnVLnzgE5+AmjPceq3xPf3e8D0/9x9+evvsbR
vrw/Nx8QtflZlWr/ktDbc6MkDeMnNoRC29rGtz03d+HBDSlhaxUxP4n6vO2ZsloiRGV8VQMG6WLqs
MNGmuJhc74PCnPyCL3GNFakltBFhUumOoNSE84dzWpShGwmGGLy6Vak6Tq3ztzEpZPnqb9q7i+it
QOnQH211lf7OGDsH76uDPodYrIzMpy0ds9z7ve+0E4ePAeADY6zvkkqft8L1hZWxN9z/3cT6rtO7
9trA/P+vXNxl0ae/v47CLltEys6FV0ZB6f7AGw+EQ4yOT5C2nrRYXF7tU07qUGcd0SCoGV2tgBM
av4fgaciHw07phoXGSKZE5NUQdYVcfa2Qp1evOvp2DwGXHLs0ejlZpsL0YdGHx2LdgbXkJzhw/Tu
+BFC3w3+GelbVlOH3/nbBx+rRLtXMWAlNAO3k+95Vb4V3vfz/JktIKD2fGw2Fezwejl50+deZvXv
7yl2wb7LaxPjwfnZW826WmG4iu6q6LdMMhaTThTGuz2XYHve+i3DpxbsFFQuN+IUhENWm9QSAStn
swMgNygg0vL87d8+CS4+HGKmkjbp9Z4g89YO/o4I+jutZlGnJDWy4VhfQ7RH54eR9d8LZY8fg+H

33w/r6untOnH0dwgBR37OLcPrwIRdVz8LQ/Xxan2ABNfczh46fhde+/m/c2xxCq9mm94j/BiN3bz
hCg/2P9x889qbf/i+/vn00/wkP5b3z9u0753HfO//4Zuiufh/WpUhUULhyERFXl5LigHiSpzDRnI
Jaysr3mMomKdeenHrNDNiuLbk011XxB2tncFsWjNYRae4Tl5d2qKZNFx17DCa5aDjo9WEFDRn38e
DwAC54dj/Xd+1z371ub30AQ2fwuLu1M9VyaW2LxNHWTp3BqXmYnJuCxxQT2lOTsDJK4a8+8VW4//
gJaE9MwvT0LDRwN471xdLIScbYOk3VyBnxB2bnpl7y4Q9/ant0bttYH16PB9/9p/85H2z8ReKiD3
NunVHQyvI6GV+Oi5Vd0jvhakBccEULrly6i0JmWiciuR6RXZyIyVdXYIDyoTiB476HKA83Ttyijt
MxBniq5vTiEqxjf9Wl1ttqll1iNnWD2Lo3WaXrfnnAAaKqbYrXad9KKwpk5HBTQ7bZicnaTu7NpoBG
/5ymE4fHqR0t9OZ5IMlVJ2TJ8tOyGawaXn1jox5ub53Tuedc01N2wvVf42j+151u9GD1rLHsh0S6
PGrslDJbxuOANx6SWuY8T6FFJYcelob3UNptod0Bs4caOJg4vTNouMjiP3F9doDLs4FJDTz1ZaH2
mdSfUHztA0DNzPI2B0dn0Nuog6u9p4AwEsF/lw21WjDrBiQ8V6E40MsefRxpB2yqZ6BFPNJ523rr
n3cXZg4H23HYLDi8tIMaSMQBtDSHaourDelvRenEfql2hTy3d8axnPvWpn73uxm0B8G1jfbgUrp
MzauMyhlOqp0BISXQ/Xm43ncRNiXCAqang74zsrNr0KK9bCOXHhuaccUtkL1BgfUv9Da6pCSRd3
PoFUTElSahclzV2OujPKmGDRQyQ8YUqkwMNCwhVxlphy79rcptYuIyaSAUieS6URRgqSigXA0E
XKqXmkZ9c34OPfOgtHlpZ4cgi4B0sAVoH7313qq3huF9tN+KB0XHbUDmF4wflY6oEffszT/t0113
/hQ9sHYTsN/q5/PPdp939anbn3WUVRpLgXvI6tGJwwdXVp7gwLDzouPh+4+nJpZQiFM7AU02Hqw1
oi9vedkaaN0mlE33BRERFeWsuISK57vh6OuSHohDwpGuhRLsoWLRpiLWuggftf3dlAo0Wjr2dMOx
zhwICrOd0zw4yrV3HsDldGH1/rwU3H1+Hg4oqM2jEJg3rAigFfJHFg7xfNltUSWY6GfhaPIcrbuG
yg021tuLT5/73kURf9ztve+h69fSK2jfw78nHoRPFs4uDXvqQOfXka1QQR9bX5gA56isPi9TbkvX
UXKUcE1KysrruUeATrLi1d6xY0a4qRbJRRqk0xacYoiJgRzaG6enKEjCP3nGgc2C9FZHLKvyhnOO
16jWRmDBk3R8Amyru4YzJ0Pzdwbvlnmy27WrUVhNy9x4OrXfh+iPrcOzsEjOqRH+JJGiy0lhBDF
gBhJ8JyQSOHDFd35093rudV0lm9/cbDaee8uBbyxvn4zt1s131eNj19/UybLkY0U2NY06RnSsM1
Tuz0iqpdCKaj8KQsAD6VODouyab8Oe+SbsnFIw2zLQrhlnYNpFTESGpxJLhtYHBIXciuzq4K7JYc
P93h+suxS6Dx2Xz3ZwegfNW3mOceJtnUHBC5Z7CFzhILuL2LgnFne1Ht7owacFPasnlS9S6usNEI
OAKyGNAJWvx+HBR+Dwd5J9xF7sEiVdK1wYv/MHnnz5j2yfju2a9bvm8cu/8ouXPfKyp3/iSZdfsR
86s5B15qDoLpMaPs2yOuMi2iFKqCARUXMURVWGB066tRvQdMY4qLsI2Ea6YJPQ3o2eqyVd1F130R
YXYTVQqLu00zzA4uKqAWNXzxYk4W0ZcELdJST7o9EiMmxQmXAEiEt1WhloZQ1nwbZOdAfwyfsXYR
n7q7QVLwOelikNkaJ7FFkpsr3qVYVcgem6WiyFmmQzsEMkUX1HNZopHa54vmDV3zfE97hf4VXz
1wW/7E7/0eHL+9wP19312ePbU0OdKcaD140SUXb7z1f7wr3zbW7cc/y+P1b/iT2jWfuvY1o6F52U
WX6mncslxrTzjm4fmYMMzkTOwVh2GG9jKzMMk6qTDLNAemeE6gk1Ae3HQeDGVTLmMScAoF11nJj
OYANdorC0f1YgYkTuDxC3qg5F2EZOnaDemJJAxrLyE9EckRGMkLUIjUNLE322xBeZWKNYUcWu05Qz
0JGy4dx1SzoqNiqrUQCSRO1qibEvzDmxGChHVHvFGiHdVJ1l6Bg0f5ZGjfyYVjPqDCVcx/7Krb1
9w2WMe/Teuzn25JsllyyPShiS62yv0+vLtoyd+7+UPNRLOD+utxnWfv+FL/W1j3X78/3589gmfvG
9845u+d32192F3eHeNRk9W91exwWlirw4aLeJlme5TLBpGTrwtWyx/pEREf0KBYHI1DMW1XN6L0ij
ENQm4NDZTXoBiiGFtB8sKoQpGY1P2eE1MpSxvu65gYp7zMWWMKrgMk4CNqi70z7s9DZzRtlyLPu/
eGbRt8POgM9VMPnIL1gUuLM1ZLRAOVpBaURgk0pXL5lqTBHFKjmlXxv7WiM4Xps3vdNk7rqLr7vJ
oie6JUaqzaC1a/EhWn2DHw83H17UoEEmy0Cy5Vv0oP842n/dCTb0trtVdc/7mb7to21u3H/9bjD1
75u5fddMNX3pZr4tIU16li9HAnbX5hJ+2FcSkdrCctUozIUBrU5tBydWIXaoBCEj5uWEUk14jKoc
5cbYnaww0aVcNNbSNnZAMX9YyzFkx5G9YZYGYAk+XVft9F2pGrYy303fdRUqaHHGDK2BtLLKoGG
dbLmoutDvQdoY6ct//1loPbjy0Dgt9XJaVsOCAImoYGH3UDAjw208qq1/RKLklpAKEQsaKdp9y6k
9RnuzS8pZ41H8TQ1XgRwYzUreglZNJhpG+5XxTy+T6GbowX33aD15xV7PT+uUX/LsX3P5rv/qbdt
tYtx//4OMv/+p1za9//bbHHTty4g2D/ugJzhwmMooqipZFYdTctXM3HeDUGUFn9z5YvOda2HfBI1
x0dcZQcwnfRIO1ltBKXW2HYuCYrpoc1SQSml1FquCg13NGhiGtRTtSabrGYDpZwLp7HaQVY344co
badRbaM2QE7nmVqEQA6QVPuzR0j1LfzBmlhVvXNBw4tg499/wpLZ1i8CrFlwa1NWgkdWoMMsXpcO
UhKbShTQQ11khGaiSRMyzFajRoNFaM2rR7B6MrKmXYhFpGmFH4eti5kjRxztD9+fuHvcEXPvSeD3
7zhus/9x/e976PHNw21u1HeFzz2U+oL37hvp0333zsz7vr/Ve6g9hBTTJ3AFvu4KWkpStK+HnhU1
NnfPvP30d9T1QYrE3PQD+d5QNN4E1CLRPbxBlX5vxmONaGfVVnpNqlxGiUGplMpuYir3Ypr2UEuU
ipBuwh9x/31dg69Fy63EVjUJjaahTtpwF21JLB7XFznSbm4uZ1FlnPuih626gN964UsNFdccc6kxm
oSfFEVccqL3EPFFJWMKFFbGmZKDsPwvobXw/gpIQekcvjVRKdkF2JIRBwDq1FcJ6NOFD09WuZcS3
jzOg7N02C9oVC01u5XbbprcMXSqeVbnv2sp7137/69v/Xwt75nY9tY/5U+PvSRq9RNN924+45bv/
kLw2H+6y4V66BxukNjQrtFCdQBxsOHLRVsjeBIuIuMO/dccB0dBqkWYo2Jy4rbFz8BVk7cDMLL00
jALHOG1JydANvLeT8OMZM8952XVCFvGfLGCEVZlxKitlPfIAj1InGmiBY4cNF2lKKiRJPsaqVcSo
urJJ0BzzRc2tvpQNPVzbj68Vh/CPem58FyvQaLizcQ4ktGgkaAEZkM1fD2dlSdSnIab0HGqn0VX
2EpX9rmQ2FEHxKJmL4YqgBPcbrxdcCt+3xYeXPlDLPo2SM5N4uE21l1dd4Fm3AqzW8EW1DOMZByjf
Gp9zTo7KUPPNdkhT/6nB/5L0/6/svfceWVf2a3jfVfyem3f/cv83fedudPDfrF77nTMEVS2pYzwj
ops+ARTRXtjMG6EmFVTZ6+IGOlKIJ9iAsv1jSQ12lglNhx6RPh7Jn7YA61efXQfTuBBmoptThy1F
1q3Ede8CBxL2FdCpzyBvSBjFiY3vHRSJlDK1Aml+tSUGrIbUuYuE2OmlxadS6q2QN7Jhtutq4SZ
Fq6KzkaJ7BsYWnuHplAm6//p3OJlKZU+XPZckArERGLwqT0HvDFoxSNjCVfPpLS64gkRZPSs4Kw7

O21LghMA1/Dkf7GGwqwnMBCnruaislFMHxNVEwDh0LzfCqMo0PEVo2GKDRcrRHbeVRmia1XQCDN3
 3xpq+89EU/8bznf+jDH1/fNtZ/oY9ff8WvTN77rYPPHQ3NnzijmHWnoUVpLtgAsHCw4U1qpDZqOS
 XjNgUfJuTM4gHde/4lZMjDoXuSBh5+Wo8Ine/5Yegf+iJM1PFgujTXRcp60qBh9Hqa8VrIdN2lzo
 7nGwWBPoOR+x5uiBspSmOxrVMj6iAAMnZHWBcOeq5wXYMJ9+cdMy0CkGqoQ+ye70xPw73JPAwvfi
 rsaHfg+o/8NfQGXCW/k5DBcVdUBQM11qe7VqKoDVHVCrBEyC11DlrAJKmP0Xm5z2Yx+ueajB2ZTw
 bF40wSjN6DUGhxdF1J0bHOW+F9dFYsMffI7lt0hMhJpuycermKaJPoLFxa7DIE9ZSlpeW7n/vsZ/
 2fu/bt+dRb3vwOu22s/wIer3zV77RvvfUbP9Jbz19TFKPzrIuguTUtOSmSxgEBH5aQSV5zgVMuBB
 wp3nZuE498sj4SGu2u3bv5hiBjydWhCAKrhjPAuQVYObYAbbtI0Qppg7j9vJ65elLlzsCcYbqfbX
 YH0K31oWFxgxtAo5lCo++Mf2Sg3zPUs+z312Hd/VI6h4W2gpmdu6BVy4jcP+jnsOZe9/5RBks7L4
 fOBU+CXa06fPyqv4V1lBVVnK4aiarGOySJeIWoFEc2TmcJGKJ02VJqS71VBVChutpUnJgmMTbs7Y
 L8e47NqURwieVYF7v3je0kgZswM6ZBAbrA+H6NUDDwZ13tjpwN3HGL00lIKqHt7NiNLSzejY2N94
 0OHfnU85777J//+CeugWwb68Pw8ZorX9k8cMvXntLdGf437vwdTmaE2DtW6GHmbdcCNji3oQK
 0zAaleF0ZTKKw2gv76CgwyBJUK6GDbpbsGk+5uX7BzHtoTLfcadUqJcxd9e/2Ri6YDuDNZAHvZM+
 Di/ec7J2Dh89ddB0cfuFnI95YcEWcHzDIC+qQmrPXgjAIE5FFCLfQxFXe8JlRrxkbKNSZHWk3Lm0
 fOAjsh0tIVRScLNT5QzZrQNvZW00ZCPVZseVE6HDC7iICsLQ8BzmYwld2gaixr295AshMuIj8Ih
 fZ73nR85/7Ex/66CcObBvrw+Dx2te9pv6VL33liRvrwle6yPcki0ii0S1MZ7m3WdCcJx3AhP2+8R
 164FSRzgtppqjCIQj5cUFFtCkqHGfU0jAAjrx8Qic+b0YHFVX3m7NT0N350OgsHXBGZorAWGNlKZ
 LxsZeKgFonBWNtlaQ9l/bWp6Ez2YZpVJZANpCacCnjBMcgi+l2Dusrq7Cy3oVvdFswfMTTYNeFT4
 C5+S1CYA8eOQZ33PTE0IJRNPUdVE8HbNSG8Qisldqy5ABXdkKsFw5PwtcIEXfGg4anVACAQp1KLr
 r/OuTgyvYPzdWmTPwnI5SrN44yU1zH/bXUm8KBemZw+YkgzH4Sen38GYXTwvvdtnMC5/3o380s2
 P69W9723vttrF+lz3e9va/z2688aZLjx058ftFbp7p6tAJxHBcCovYjSgacDlqpDai8S7qMVB7iJ
 QuWSONfDFiav5bnnzxhxj/zMLExEt5o8DELxOCAR1EZ4jNvefDqaUTsLs4AikusaL+Y0LjanhiqV
 WBGkwu4qcwDZn7XTvDvDnZlH+1uLjKJhS58l4f1j6cm+yhUNTPwC7nv5DMOXs7Vozgf7GAE6f6c
 KNV73epcZrET1QUf1HgWFWakekZ3BOKimvrRhpTH4ojdmIIVqhHKqQMmtdyO/4bwtBnRX9GVUpkN
 GE3KgaySnIxjtrizBiR0JyCVMnY14ykS88zGW5zja0kV3TeB9tI1DMusJRhlGazLkvX7m60X3OM5
 /x9FdlyL5Ik5NjKvVox14KvaymRldf/Wm7bawPVYr7R7+XHLj19gtHg+J3TV48f5jrjjetALVQu8N
 6fNHehjChkgOL58T/PwsGzmpKUbxiQScruMCiKJUIrBqGDcaqafXYQroASnUBTZEBIymUZzBZh
 c8Hs7cuwwLtQ0A3HdKNacmXiOlKmSQG2mDisgB7t0WzP3Fv+cu3c2HBo4tDuG+7LEW9Zxnnwg/sXo
 A+torccwlGOXGir//g38ORI/eGNnfAzZRx804Sv+WsWTNZhRcKoch7USCFYAWzlxk9Bon++xEu
 JLiDJfaaUYFKrXM2ji2pAkLXo2DYZaOgYF45KJ/j0aoUHiRJHS/LURkUz40lq5MsNq/dxSon+We6
 6njptVLR3WCHrRj1EJkjcHz37ms+52z/iOWi390t795x1545vel021lu/g47bbv6re9c7/ef4d37
 j3V/KRfqq7iGSg7lfd6YRRXwfUbw6va9LfSuBvm411UsJ7lDFCRU5GHTArcxuJlFqiOkrJJUWA6
 ZlJ0+ecAfpSPJAYLAuLi0EI0x9og1n5x8H9cVboK0y/n4djRvLxWZ0oKn+QgMmTma9fyTe9wvoDw
 pYXR3A/d0d0Hrai+HyffvpcJJxjDQTF7fGMEtX/os3Hnb55ik4NlMEfGeP7MV4wGWiJH+aAKqkv
 qSpRkb+qYhxZff2Rg9YYIzDf/8ELkHumLO8JGHjPWSVTGAZWmkj3jIMi1ETg+4B2yF+UQKFRqHIL
 hdhCwpTq/ZEFLEHGeSgDgLL1WptEVOGV9JkGqjZAWnGe6yiXqKe8+jww8cGj33x35kaWpq8vee82
 PP/swv/eLLVren9X/z8dGPfXDHDZ+//iV33XnPb7mahMgKLrLVOWISoZbXGBquSxVGPip6Xs0gJb
 FtZZKQ6rLdpqQBY6NfY6STOpX7kCVhHY0IR90sz7XwygqkGcphx6gSR6YsM8SR9YQBfD7svU7u3g
 dHTx+BXb37YGayTodH0+RMnYfJ8z4DXfmIDAhbQ/21LpzplmF44Y/A+Zf+GxeZUGeJR+26AyRIWV
 jeKODk4fvhsx9/s6SobEzG1BNnPqMgZDu1MtZualpViWxkcNpWjL1KN8wkNS5LCEzRK8YeQCgeSG
 +ioDlOFVXAL0sTcr6mwEg8G6oAe0SY5lIkt9U+bNhiQC+kpL0k99Yygk33gGpqxWl/wpmBu9+pS7
 mxaV1Hhz3SZm40Kt7+7ndelf/g+z98+pGPfuRln/zkJ3/5Z/79z5ttY/1HHm/4q9e2bj1w27OXT5
 35kzzXe3JtW85311FcjKIKHkRLVDSqk2jShDwx3tw6UlJVJZvPB0DabTj5jVMeckWu2rilS0aos
 Ml3MdJmYqHkZYirOUbTysVZdeUrk5suzoMv051KL0OI8dWCWsOVSOWPG3UYOqSJ8CtH/s6f08jOj
 Azl1cGuBzkBo7F+q3G/0qW7RT6EQW5gpZiGiWf+JEzM7qWDj1/DBcu5dq9b07C04Yx2eRne+6Yrwy
 y6gsIaQWm9sWmpNVXkVJQM0yThZ2NDiIfK40d1IN2DU4qcAdaiNGuLAFBiyMkh+NZsMvWRZGJs9T
 W9AfJEj0+zY4CLHbAfGrBjIJQH5SFELVRNcN6YIreHYOEaYjWloycnDu6PT0iz+8cequW1luD/m
 jwqM23Xnvbl79x5qlvfuvfXf4Sz++r/4a71trNHj/R94Z/bJT3768SePnL5ypEeE5LqL2KLULH
 JdqEQUAF4dIdFTaxulaUWosAKiKcASJV9iqB7AQBofhxYrm8pZzyFJoJaGYU/tI1CpddPtyY6dN
 AyORC+j8leXkkrl42j1W7D/h98ARz+1k3u8I6gjdviMG2j9Rnu8BbuIDcmYNQtYK0xG2af8Xww7u
 8+ZefWBwIrzkj7qLWm4Z1vei2sr5yS1LLk55a//Ne1RFwdHXyzqTYcV4gIqPKWf1c0ReTTYhDeL9
 WrLqJ2Wm2Zl1X+8klpInRFWU3pDXPcCPG+4n3g7ychawjv21hJfctaVyt+fcqKuEHFeAC2enzENf
 wL+7dpUoMmhe0iXnOhTUsXxf4in++57cA3T73wBT/+Uld/5GM3/6s21tu/cYv69Gc+cf5NN375Nz
 fWev/eRQZXh+qWxetMqKkh8EeFOtMSWhv+bKQepcPsvWvOzX8EHQhxTEMkCa0H2iljmLAOUp9l3B
 ukelWYStJUEA5tSXinyElkghTazgB9hWbG0jQDY6Cje92J+V2gH/UUuPf2a+GSXUymoBbFMCf1Ql
 Ra0Psvg/nHP8M5j5SUHag/alHyxUCf0G2XIruf/8wH/h603HegbK2A3iIdNWKw+P6l0MAGxiW8xo

1ynMjv/xwGzyWlLWh/DxpdE9guLc3q4nUhHrT0rUuHoMie2ZhmaEt/KE7FVOrfipKFj6R4/6g1ZE
NkJbTB8pCCluf0hognP75/3JKA00H0NTokbCYEAX+u+PvA7ucv6A8H1z7rWc/430VPuOzFf/Hnf9
X7V2Ws7/vAO2au/fQ1P3n88KlXunpnKjd6goAiAVg8XI+lm6e7+1SVjNOvVuRd35yW4oGXQWRGL1
hxLxgO9kiT6mVD9QXPTSVSayJkgW+TBvpUm6M90/JmpiYlPc6YoC49XH908F1miaXPgVTDrfVZae
08dC96Ctz+5f8Jj3/846DemaD3gthW9tjHQ33vY8TDaZSct+rlmhbmJqPFA+YrgwK++ZXR4LOfel
8E7ChxWr4nWkh00iG1ZQO1khZr6a2aAdrF0XPCuMnXkACbFuPhfioJtikeQEdCPgqvdtQdijoiCp
OpxresLQV282wsOk8l151fg66cfI4szNySolG180CHjA6B0HrhPytRhPQstIQ2CKRQx0XU7mdzwi
f65IeHZrQpA6mQP4g+aaabD/vxO26/8+6f/qkX/cT7rvrQgX/Rxvqlm7/QvO76a57+1a/c+prhxv
BRwyJv2cLWDR1KK+ADDycXUKaaFAHF2+HFL0xe9t7o6HF6588hnwXFN5naLbbS1Fc0/6kpTWXFP
UOjxWwwnt8qm9UGiIrz2mW4Ajup5menqQoLvoKZY/V820t723t9QtY7Ro4c2YF8hGS9KdBPfpH4f
7Dd8Lefa62W9gFrYufBKP6BE2a0K5kYlIpUha0hl1EG0MNI4fuhre98c/JaXFmoYKRUp0H7PTYUO
OswkSpZBrSS5+Kxgy6/rWY4+ujK4JyCLCxYvra6zrpha8i7EwQk760EYek+3x0Hto5pROpOlSfCd
CEjor/DuF5guH6Vh057IQIFMhhpmhuJLWebEwjI0hZnmW2/0i8CDEru7ZmcQCDL4/07zDYe5c+
vD/UavfeZFL3rhyz/0oavf+y/KWO/8lu3p56+/7tFf/OKXX700ePapLmVC/jnyckv0UfOhQoi+rF
/koGkTeqW0PpGMJOXDKIcveH8KLimRxEOaes5FNe8hOzSiBZS3MrDRNEV//OUQgE8VGoDQGFk8T
C/GN/sRrMDnakOclW1l2oiMAOVBe8+uAifu05GOHDzzXD48BGi7s3M7YAf/bEXwFOf9mSoPe4K6D
cU3H1yCWqH1uCySlrOIaQEYcJQg3pp2LcdFqxxtHTqJPzVn73KpcvrtK3O2LIVRWCAR70sz91Spo
CMqQRkCByitNaGdhZe+ZgV/nbpcGlC7j0Ne+FrUGZjamqKQDv+eQgcX675oz62RC821jJqKrU55Y
17uxjZq0yrkqTB91vRVBL3lC0Llrv3Nip4PQlVfQcGCbWUuZrGmJSt8UIQMZbkcsippYkECf637r
nmNjZ6b/qpn/x381e9/4N/87A31g9e/a6Zz37mul8+fPDob7iLNOUuSSvXNmWgiEeqypqkWit5Ug
KnQmmg9tGWbrohRahrOIqaIJNpTTkVgzfRSMqHBK6GZEzUFvDkdn4emlNF0jjRAjPJhLWQz5ldE5
hKmg2kMzHj0lSukLHdI9lN0vc96Iz0DW98K1xz4y3O+DqwcN4+aO19IpWZ1WCqlCHave7iehe+eu
BbsLLWheOHD8Gu+Vm4arAGP/dTz4RLHrGHUN9aUkB3wi6p18/h797wh7CyeCK6fiaqHctsBGRant
5RwvOq1pS1KRtKWjHGrcClqtH6yJbS/UFwiQ21BTMzc7Rwi3qhpsx7QtpMzjb1cwhKwAPfWkXB2f
D7U6E9FFMcSxaV7/+qCnKMRJPM/ZcS/pAyvzvXoTwh2+sdBS4i4DvYbwvzc47CUiz2D5t0hNfi4
1Bv/H//OrLf/nI3/7NGz/6sDPWT15zdfKFG258/J133P26YW/4RGecWifWY2SSUkIbs11SB0SiA8
RlnqgM4MXV3hMriOkt3kBVqHGSCA01AmRAGNOiFoy/GYiMppmQAWQQ20p7gEiaj4vRhvKE5UnwB
calR24LYE1a6vdIX4sgyNMx8Qb/5HPHIBXv+4vYV3thAuv+BmAqT20qmI26cFFkxOwmDfhlumrcG
RWgh7oNqDhrsnF5z8R7nCBBqrtyD/zxm66Hpz79KfDzz9iHvAhKg7suIr/n7W+Co/d+0x3KnFdc2t
IB+frUUwSJ6SOfn3jMJqkoFCqZW9lSkNGfhRttA/PJCCmfHQTWqlif7tixwPusiBiaRHZQUflB/V
nhJAubjMkamsAg4m8pbuvEEX68XUMOmVAFRRNLDrNytXhifmW88c5oKZ2KMjvztFMry6htlcljh03
KfJflsIwBURbksb5T3plcWl974rne/7eIX/+wvdB8Wxvrq1/x+6+abD7x4Nche7Q7KlLtGLefJUq
bklV4PvZsOcgLgVFFCVqH0anxxcw55wrt1ZdyEhq1LcJXRn74J3uh1JUqzs8QD0RAQo/xZHpb0lF
MhnaugPE+thoSdCx/+hCU2kayW8XvD15hwkRV11/xhQjmWD13zdfjDP349XPh9LwS8zEuGm7Ao+
cUrJgWQHsnTDZ6sDA5B99cmoYTK0dhxqXSq+leuPvsKbhwrpJJI78PRv1VuG/NwpXXLSMvfd80zL
UBPv/JT8EXP/Nud9BGN5bLZ1EVHsJ+cF6JQYwQd2BjCwpWyjRxYwMlN0fG3Aa0FVYVqpKT0QiBE
bT2dlZrn+NCIxbG9pEIRsheVRnjC0fHUuDtHJ/0sa99pmTX9Hh/+7BJR71MxVeM9aLOOCOMEZ5PY
BUPkCcPdenPnAI4wlp0j4bS8uSwMu02kiyhv+ueHvCyMIwt7Of+9znXuGM9Y+/q431zW/5++zqD3
34vw4Go/9sdDLrPGSdboonKFiWxgxweDnGzOR18oCmPDMGPVce8VYlSiDgZPJJSVJoipAk3O3hn+t
1W+qBZ1gj8WK6LpMealMT9JHBdPfsnkbQ3kfKJh64JrUxFxU96r832RGA1IaTx9TuOwGvf8Bcwmn
wCnCwm4bK0B9neKbh/pQ/7Wz3IXZi89UwNGSP74YILz40lXY+EtjkL583UYVU/BpZ03gPzk3XY60
yBU+unYWbUhZ//1Bo8deooXPWW15UAlzHiCEuan89KvAHQQU44xcFPFCKoKudVfd1KURkR0yQt+7
GWe6EmsZuIEfj9iYkJWhKNdiJ+DzTPmjD5wCs+pDaWZTPBofia2YPvcR0aSp2QDuvAhNTWO1DWPq
ZZYp1XKKfEWMP9IvC7iM4Dt3sUatL59SCYVGVpOFuerkpOOjgXTecKv9cfDurdjcGv3XDTta972g
/9cPFdZ6w/8zM/MXpi+JnfdO/6Vxq15myBiC56M6k/YpaMkUkPAYoQAWi9BKa02SiNwfKBwAbNcD
seSJQ38SRuPE2+zzfeS4yji+cFJ0msKs8RhkbUUgGDhJdqZUyLAAzFOkMgol2eokj6t1nK6ZcVBo
77fmdyivaeYsQ6fPws/MGVr4WV06dg/ngTUJuahweHI7g8X4F9k5NwOu/A3o6GZ+6fhTuWZqG/vg
17GkM4XLTh90Jx2DE9CUvJAiyfPg17pyyc6dXhZJEHwZ45CUfveTtoXKYcDmMR0PHSSE1ELWRyPr
75GMoMiUp8bXzEwwq1hVrGui/eWWEExkqRKSgqOapjq4mv7MTcGAq1IjmouV6hvzdKGT6LPjkHWY
siqEd/8woqXWUn+c+VWjnKmN4K0kw9andm0MlzmquZMYZcZQKbCh4UwFIq6qOGc2oZhIwdP0QccV
yFQjUvQduKiS8YcYENGONokedTt93xjQuds777WzmId9189//9nXq+T/27F8/eWlpnm63+zvuDe
8aFXk9R8lMQWp9/y1mpNhNc4q+F8qynAyRJ6EpTg0TbaqMHG0qDJPSUJMKPY54wNpuavIHwERYXw
IQ7Nh7s0nlNT2aHEbjjAocVO6zKphdWKDIcfrsAP78r98FD953L9VNO8EqPGqCwsT03DrWXfYzB
AunkldhM3ga3cdhp11d0jn9kC/lSIVeyxcdMlFkDYU/OAuBft2Qfr3RXY0xzAar8G6cH3gR6u+O
XFpAeFPoMM1yqpXct0OKbrxdegKH8qI4PzTi9Lm+HQ8XgR6jFl9PdyuDzCCNzfvaHia/kFz1gD+q
jma9u47cM6hr5VJkML1Af3O3aSSfPCraUkyaLIyj9H/VpkYaV8vzEdHw40r6dE3WX3d3wvSN1EZ2
/pmuUyIVV2GGKj9dfLXz//c7RETDFJPwzikW/abA+kPlk/dN+h5/9DtvOQG+snP3zNld1B/0/dRC

ETWucPXxqoshF/00I1l1Qgflh8kxb2yRJRjXTHRQoTdGDW0QCB7G4Pko5fKHqeybFcKiUVK/mQBOVR
YsRYbvgrSI/65YXY+ACGItZdyC0BxBjMy77tzLkVu4IOF/CJ87vpPQsPdwDbq9TabcPfiEJr5El
yyMAX3rNRgcWUNnnXpDOy6+BJYWL2BZPUMHB004IZ7z8DJQ4dgMW/DgSPLMjN0YMe+C2C0ehIeu/
JRSP0zFNktI21cHohzxN9521xRAWSYUZSNsY6SCPvNuQ0is6dkREYmi5QKh1fNrvNwmRTUuZB87F
5oqQVV7PBMEaOBYShCJXLtlRmb6InxhmoU9LtgKYU13JLBPYPARbt8gB0Z0lbs2QcnQmsudahPmc
U0Nh2U1OUVtWgEm0Cn4I3WZ2jcs80ga2akBEKZhccssz55efvF3jbG+4PnPu0KP9Ct0oVojdkD0Qm
L+gppvElH1RMTLvzWJonIGCRC5CwUR8Sp744npQw9e7jOamFXL5VlHWkwJk7WB4Lc9aGh/hCkYZBq
9tmcoLq1QJ0MSHS0Mps8dZwN69e+C2g4vw7ne9g362VW/A3PQsLDQBpqn4d5lgPWzy3DFJTvgjJ
qHL37jKGQuajZm90JtsgPP3Kfgyd9zCUzMT8Cj2l0Y2jYcPPgALB68F9buvwW03/MFd/jcYXRpno
8cnsVFnkOollbBFnWl2pRdcPpfEhFig/ZMLpzXVGIEccnB1wajCa7u4Ajv68NRKD8Y19CBfF/yhE
XWNAwVMPC3eUDAD7rLDp0AABIDQH0rojK6+4tnd88NqlUYWb5V5HzWaa2J74tKtCx06ah96RPOjT
9zck38v42xjzg7QUBc7nYpkKaOb5mbzg4/81v+dvknNesr77yv6pBf/RWbe2E916Yy1P9YvliXa
6PfHeFD6Nb0IS2ixBBcCqkkTv61Jk1ntvqA+IalcsrApGsakeh9R6vVaeDaPZm5JUKemtXUraW
n/SPsF6yu2h3LGEkmPxoPNMnWD772b1+ED734/rK+cdTewCVMTEzA7tw+0i67pcBl2Tu2Gk7oG+t
AJuPi8XTA67xJX0x6HVv8knKnNwYOn3M/UTrmUEB8c03MYHjPZgNuX01Dc/XHIj18f5kR11K1421
4RKA4J1YXo+LyBbjLSirOTcTJxaLHeb3CGppQqzYs+NBsT0q4aUsuKgRsJCKtvG/DKSSbxJxESHU
ua+nuSBaWITbxf+lpBKC6rT7ADoKked+aQ/YUOrKzdfVaVV0XGfe+dSixdoTiGLiCev7Rs65HDMn
bTEi7/d0x9m806L6J2Qatg0Sj8c/rAA4fw8A3PqbHe/LUDj3dveh/m//gLD0tNgCCPqsaAD9ZS3D
ezm5KakMJqYRcpP9FiAnSe404W8cBExA7IZ7XGqEy/uAc25iPGPBMR5SZAheUEwt1NAtBBotVYV6
W+Vi1K0S7DYg5K5iyZ9aJJ7e/qj34WvwmNLxEqOtGehNmZBdi9x6WwE7vhqydyuQ7Cxfv3AdH9S
RM9bqgNnrQT2Zcabg0l88bODx5Hpw9dQimTj4AD4wyGC67P6+egPzYjTCK6ix/7eilpX5PoaxVAe
w/Mh0D5dpG7mEFo2RBOEv95ZQWL2MkcrXfsEukh3Z7Cmam23LvFIFJvpa3xpSDAT5RJ9JPBdh61Y
Y4U8/bDu8xKYknnj6IPW18YFnhSri+dreUghYV1FYBbAIECdwCFcocI8J45Ozw32gbAUuaHThEWB
bhFGURb7nPQIMG7ChIT1rr9PTimfY5N9ZhT/2aybFJmFTIBbQS0DBKhspzApwKcmgrPSoplMChT+
mxjni9ROurRo5WeykrYrftZWCsVlKSchXuAra2FIr2CgkhwITR2gdlK5M0bIw8pcPb2LjdkNCwtp
UsoIzoI5eq3fet06Bea0Or3YcPyTnYvXmv7Nq3B0a2BnvzGhwelWD/6iJcuvs8+NZ6G+Yncrh8Ss
NKuhPuPHUSzmv3obNnH5y85wDMOU99/30H4fzVT7vrM+DRvwg88oR8GDuQ44+tgYI2CJhZ7zCZMU
Iua9gf8QztYACD0ZAH6i0Q7xnr8pJsIUSVUFKYSI6lRG3VF1VaFTNQoqlkQnT1jnQc7W61WkLA0J
JpFJSmek60FwJQBmCrq8LdIT5DfjQSV5b5a5So0JLzKozjwxtpxshzQMhda47yHr2/WlrzWVnznK
bBf/b6K5W7GM911yPVesgFu2Hqn079bCD2H0vPNR5pQ2tFC6NEIKOY6E/FIFUZOXMln8qjFHyhI3
CqbMeUc454uHCBNwI89DPUU/OliAJrq6NrpQSLJQ1bT2PE3ip525TUpikK2TD87CVfhBWj2ySNOD
FGNcJmN6xC3bt2Qv79+2GYysDGN17DC56xMVwKpma/NQpePKj9se9q3W4+fbJv9OuQpFMwn3HDS
GEOQU6m4L84A2wf/BNyIdLAdzg9Y1FKXESpXrjxPuteqEMtqURQGcqmU6GqalzoN2NNVhbW6FDy8
oOTfe50s5Qd4mEDEvdmAioAWvHxtgE3ff16Ra0RX7PmomCiSyn8qLpnFuXvHD3WpixIBiJqCwz2m
w5eSUZheG1kiGTK0FMCMqUYLlaogo/WlnCJSLu/t944AksTwzhL6QZIRiExtrruvtLk1B8HmtKp6
PR6Nwa6+FDhybcm2kSdU3VxPP4CbKSm0kXUBBVdmSa6W4+LSxQwd5QHcrejrV1SGQMNYsSP9fG0p
aSGUuDpGmocqkLBDJTmoh4GXt4TJvanSaxiriPmHzboeoSeWT1B0p/aNSNUz2VspFTL9iUBsKqhn
zzcGazPTVDCoMLe/bA3vN2w/7d03Amz2DJGfPkyiKcv7MFZyb2wc13H4ULds1Bfd8joN49DrenPf
jK8Qb01+6F+sDVhr0TAKsHea+OliF6eXI6RA4nBktGTP1sF5Vbl7GlCYIgKg9lca3IbBaPXhjPYlu
xyRSNBZhJnEYUMRjiwIqM0FijbKrxEip0pr28kB5yIUZTgUkbTRCyelJSUPw/uYHrtfrdbdtL3R0
NXDogaCKHiEVAYb2ff9NkBggeHMYAx4A3fo+9eeCZbqFUV7bITUAKF2XnEDj9Gv9s1sXGLf3fvsQ
mIkKvGOUWDTx4/tc09itQpdlhC3UD0j1Ko9qZsROviw1I0YWJFOTL3YiliePUFndBOF5DkQ4GfQ0
1YgiWplwfUVFsu5K9sEhBEFH4EQ3CtYtKuVoTeXqiFBP4nkMkyx9Qj0OPpoyeA+1TSfx/TMEQEUR
Uf099dO+Zh/65Z2Dk3Bc1OG6ZqBnZPjNacJuGsM9gLW+4zL1zojCGBx06NoD6/G9a6G/BYV7cub2
joH7sB6ku3MJUwtKY0gR0+AYmlWsI6CQubiCKhFRon+dJD9aCSN1T/byddZoAlPxpqq9UhJDsoaQ
RDkkqDMqx75QEjus6+NSOzrft6m46sq03bsh9feloKasXg39EJ4rlnwy0F2WAMZCRnUys/i2czxf
pZHuX23/Of3f8cXS9hLTE6nob6lFslDUKA62mdVUCUGgx50hwnConE5NrOeLNSypH1OI+tGd3Aeyn
sqE/VLBbnLjY+iQAOhibyl0JtL5CZob6CRRwQBoFCGmkQkOCVYMOJqv/dEV9Lfb1lezEQCAHGAXbR
GE08jwbTxEVsUlpz/URhhJSRT9/e7UkF4nY0AFpaER+qhws3kCnYkGzEw5A3VGitvk5iYbMDs/Ax
vr98B118zB4bUBHLnrNtgzNQWfOXG5ZvphYXnd1YXpBFwvp+A85cqLjbtIh8nYcqkTZ3tFtTUS1d
9WyB0Vc4jkQX1UDf3LhItCFiU3Y+1z2bZAw/WlPpcTXnLVQqvw7KoHuqah1l0lFhOh1SiHnKgXwX
BW//AotAd18HmKnOVZJyfbZDBY2jATqarJNL5u0g90+Cg5nj21W439pUnJexZKK0Qjg/hjbLAZkX
bQMwfoaPFsdV1U9brRzsNhTKUNfY1G7dxyg90bmV7UkjUK2UGCe8ilZQ1NMNLGzO89oQPM46ICL
5BHtPIFRt0mmYhXek9nn68jSd12CvYTYworwXAFdf0pMjZHAzIu7eaouIQFBSyANGP18BW1hGCjN
55OVIxY3nmMovT7fD6NrstaKA10XIGWyc9n4WZNGL60JpqwOWP2u+uVR2+ft8xuOzx01wUrch6io
Hm6Qfg600H4OKZJZzhoMzASA8zKP4LFziRpQExSFO5BjaenkkCTZ/ir2/loIF5pYyxlgTt93Ep3Y
4d012EaIhz5V8021nYQMQUIUqYS6UJ/EofKXb3Lghsu0qgXyMmIIEDEkeLnPXGjHnc40x2kkNxSjUq

A8ED1Kp8KtJFOM2eCFR6vOa6uyJ0zvWgacJctUKO9wogojNw7V4+uhA0EADo0WpTtqIuNaq6fk5M6psboL7NJgk1qZ+0xC4zp8XzaL8aev0AoLqRndQfRkA55ycBHJe0vjgQju27GiO0ZlZxADF1fpUnbFX3AQkrlhAAo9m0oZWMKiHw8pekB8fQRLgi7vllEomqggWzVcsya8dY081S2pHp5KV8Pdou4m1pzht93vHWS0oD6uq20u2NWGiY45H+YesY8i+K0PrsIj5ifh9OFjrn69GParw9B16e/c5Kq7TmtYOMt+JSowFGh0pOJog+iXR+TzaUQpctBiImnPRBZs4UdMhMvKI2V+KVUVscfos2vXHrpWJuJmc9mSU5+XgTYVSA8lyUH22tiSrQRJ6bzL7EfLChJufVlpSyE9kCP6NBmH7+EamRwqe/Xe0QApPOK1yALJxm xJPYWD1ltkZ9pWtjH4tg/L1phKuoyrOmsuBR7mI3qv/v3WgUcsG80a1fzT01P5uTVW0POhTky8t6kxyTmLahI/uyjRCGQQG6dlpPlVclWtCTcxCHwJGJAgIJUWMRwvLJ8f9HzfSs3qRcxARU18xeCG1E dxFFPftp5AKawhLphKV1YuqZPdGOhuyhRtMKaYVZmnxR10q3GrVKCVu1RnIOdNTsPuRF8FiF+23gA s7NTi2eAqWmnvgUXASDh1fhObwqHu5s0I19GNcrF9M/T4biV0Lw4rBGE4nEPQQ7xL6xTw5wml/kD 5JEXfiWXYFM7PGtYcPt029TEv7fDgF1mFyhVtuEMj/VoigtMLPOWOJnJ/vsSZj4I8N/WL+nLmLUv j8kzgzQkWWHnuXULPGMDdxQFkmLouMtGAwsnS0PJZQcYxWG4Cu86Ui2dLzGx6ZGvdkK54WWYDlnO8 wHdHb6/T4Zq8c+UmeonfYkTSC1JjuDc2qs7gPupVgTQfR4CKjXanxaKNM0NhHJT8mboUT4PDSvxP Ma6fclcd2EmssUyXiMzdMAeYImCQaLVDSe8IjrfXp4vxglEldfhBQNXz+aqrcStnsiRgnflxviPC D1JUirfGTZgo4ghTtEtQb33xolASRCOn/b0TUYqDZod3Pn5jrwrR0nYL22Ey61p+HkMvdQs5WvQ1 5wnaaFlG9JjLSTURhgg6lq/aowJmbDtU+EminvPRVzQYQbp1IsR0RWu/e7Txk887VpuE5+YAINFq O9KVNVRVxuRnqDsUt/1YiIXaag5CATBCmjYpYXUSNTmLD6iNdrqJfLI27V8Tbr6f/KZzZJmLf1Uj wIjYoxylgbz7RtwjeYSJGUDDdbStmQmBoCSfWaze21k00ETrGvU3H6JmGAqtluwISrsRfmZvVEu9 U7p8bqbvIPuoudBq+Helzwz3keCBDjkuU+SrCyiboJrsiqep8ierLJiiK4Q6hOEY9pX6UANCGPur ehx1K1hZHxJraonokP+TjBnQXQbGU7nN/mHasCeF4wbnpj9LKq81wnqRRLG8d9FGu5FLjnDuCXHu jD/NwkJK6OPXr6NOjGLDxar8C9G7zKYebwh0mwu9CDMLIbIwbrheNUGHa09X3HARZ/mL1oWVhtkb DEqb9DSZhVKD+rr/28NjKCQgWR4AtSxfBorNQ1TCfESIo9cH+NEWj3EflpI56d9VGN5Hx0MBqaiH Gvg6kvTv4Qt7coAqYRn6VKHR7EyWfSqdFwJwZ00yOpWD8EjFFmEqSlAgxZVuFpuuB5BBtS9VFH6 xIwRHPG3UBpmBuZpZaXNjgmtu569waq/NYFyBBAKloPICLByAvc3928Ryh/E0PF8lWFjmByGUSME ELrX1No0TTqKB0FNNJXB0RczJaCwIH909LiThsbHGHjlofT3iVxIIss0ADbaKUHBVCJ6vJHBM85 oHTstLeoyOvBOCMUhwzbVSAcYILv1SBd6qKy/3IMLpg0sOkOdWzkDR1zmMIVaul9506z011zUKq Ihe0GCKd9rPRWzKDMUxayZmJAPSaTuoGzon9LoG24U99NPkSwqeLXGhEE8X8dX019qoeQB+IkdQ1 WQW6ahCkXC49QLVs3QKmFgTkqcMDWDawm4hxbBpEleqGyKSD0fSAmSni0oSq5pxG0cTKtsyCP8wm c8uoKaU7fBD2UI79dnJuzcIi6AlpsyKHx/sfoilh5YBk50Jqi9tbBzHuZnpvf96Nnp6eE5M9Zv3H lLlthaoqw0o8Hn9T7dAJLM9FIorGvD0YZ2w3jSufH6Oop3y9D2MCsSkybUnxQCV1//QiT7USoHYB 3h5ykrnNmX1sY4eBDR+VQa6Ejcl2iw+F51FM4yA4rUIPstvJb/dwiGILjQzFLp96UEbt12yh3IjQ 2oTTagX5+B5plFOOsMf3JuBvS1b4DV5QcpKkFEIdTGE8+TgHBaU2c5FV7yQM0xn/rblMG5LC31fv 0GAQ8A0i6dAvvB9Wjgwaf0QNkI1fHuXpG2lPYGOgoSoxWMIFFBuI5LB10RpyNKY1pSS0PLRFp8Pn qOXJ2KkQh7qUSWiWiUJrQHaGLMDWdjH3RCMOqk5LNjaPAvpaliOxBB1/5Bj1Tc09pEEb7QpsnZR mctJaF3jSePSTxT0zOuIg6DTOTE7yK093QTgdz7gCmO++8o4EEZdo+L2sgKEXzAQnHnGyNDjUBDy EK1Y2Zsn+1ImPivhTNOyZJMKS8YNNmLHbvZqSGx60hWT3h0k2/DMr3IcfnW0MdfVFPJNqVU1fflif zWjKPDGU1rYBocOMa0G2UoIFRGv+ouqjZdVG3UM7j50BocPdOHmruhXZfmKncI1vsjmF9YgNGBq2 Hlws+KmlcQQ6IWiInqe7+kMqyEwLOIbBrgGVSqt2RPTxjqFpYSDphTz5PuzQi6vQEdsJhT7aNkvD 4E60etfSulqq3Ly5NZMytNPHMoZbXktBZmiClaoTOWKO9TW9+SGg5ZhguBGab3FmNE14hVFC+eGl vmzGuHPMwVehUkRYxy4hqZkHKag8bTsbFtSxx8fz5wO4CLmtOM6pde49LuD+3JzqUSXU6LVfmzF Avn3ctgZnfsffcybpcf91105x3TT31zSOOWRsNC0JBvehZ6FvZcoyMhbJGxNstmUQ+hea9nx7E4Z aMnye11cFlTEe4rrL1lAfYUEdoHa03jFJz8FMmcd0qgwPKRqNZHsU0505SeZ8V+9Ig5eIhtDN38+ rNhjPWNr0/BjY+9AQti8PYW5+EjoultX+6izsOm8fLH/jRli9+R2iXWsrTbUlkcoGprpb1KY8xS LpZ8VAVXB2PvNg6mRGbbbURQKlJTMQtUa08Syt131OU4Q5WfwxPyOqZLOeIqekZdeql1Ki1HOixY 5Jg7KKJKhQ1BpPHqjBje1431BkrQikxU7K0wSV1NXE1OJauFSy9KLeRUrvTYLoXQJpGLb0WYCJ9r iSDBepkejxmK8AypR+MZuLlZ+TRjQaabtJ125mapp5y/iz2CpM7ejC/efbc2as99979PsLY+tt0Oa LMlOb63EHNIyE0TKOctCH9XTnk2i+DspImZzIkXc6/KpGeLBFYI+NXxLvGKQne7ER2a7JhGhb00n ZTHTPolVVxZMUBqjLK5bMRSaNB841spPNSnHmk0i/lgEuP5qdmYWpiEtqdGpzqAwx7tQENDY21rNO D08jrsowj7rymFYvvGvY5Y0SCwkrSOIoYhwkA820kFA+1sLVciJgkDY6DK/mgqqoWe3BDXYTRdki pB2YE401RCaO8oZGLTIYMDhd5U/9P9oHZHwG4DU0MLwmaEpKraAoaXaMBecwkvPaB5w7R6fn6hUr rEr+P5waEmVyl8FsruJKTkvJoLSgC2squQyvkROiOUS7WtTFGB4B4m4liPnx0v2xKXXLx+oxn6rw hC4e/TE5N6btfcXf+QLf2zG+tgmp886v6yEnb5wSjKotZR3itrCPDRVJraUW+LGUVm3WX+bVqpUZ SkL1731qOWNEUoSOWoHzovxyvI44kb38LwF9Rf+JDM2JJWRoZKjMI0bFBTAZEUbX4cgrPXDKEeE 0hp1IJtFtTMDU37+qXBky4CHv1LSuWUhgYcZdsMEXRu0F2JUvw9ev+VtQ+SqLuzpCK9n4bbQoyl ayl4RfjEgOoa63JUE/1N6pu07S/0wqusCq7HELWJdlrABpw4oN4d2KaDnpaelStsavn8AshAgxWO fZhEcWldboSUBSMT9ORTQ8GKKQHoq87yLqfIhSFYwB0vB35N4i2o8RmqNmOtbek+iepluM3fkzWo +coA4UTsrKFGtnZU1104CXvUkD8KTIcAlqRKJqZQmBAJl7rrXlFej2m3RPXSqsZ9u3/wvnzFif9+

PPTlZUusKShLrl3qfMGvphcT9vmHieKlHOSsDIt0asVyUcQ2KtcAUV8MAvb1jIWL4ji6RC1dgae8
uDYTaaPByfwggpccoRgBTsMD2sJRUGAT8L/gYERcuRSPUWEzpBPT7BumuKai+cDjmbW7jnZM95ZF
e7ujBwbCOBR8yswbFbPg2jY7cRgEMmEnNk/eibKC5411GsfhHPayS+5whJYPTQ+9UJry5My0NfFU
2XA5NlMmZmSvElT/fTw7KniHZdVEnzbCdpeFLGM5nyVbh36pYbJSkVFGjaTjsu2vccHV902QVsc
oHZTBKc/RUhwg2jIhXomY56bXVHEsS1dJh673s6PUoMW25tyWa7utsD0Q14gTw3yJjqWZorWGq0
LMEHp4PWqpyypXG4ELWSt6ULvmJ+59ZwZa299dJ77AE0WupclxUZf/SYGmVAG49iQZwR5hgg1jn
6tRTByU4nEPuWkQxx0YqtTff7fx6ADHSZTVfAD+PYb0OLn8rVYOS0iLSZlNs2DxuhyALEkjcJec6
M9DZNTbUKEv3JMw0qewmwTYL03hN0zLdg4+nVYvPmDldqswEipTYVjXPZAlSg+ZBGH3LdtxmZXJb
Im1GbavFgqviY0b5lloXazm2pFMVob6RaBEC6ScnhDiboDnQcOpIyCIlgF1XumFYnKOPOJv/bsXi
gXgOg++Eqc1Por6haJ7KCJJofw/cckER80QEdi7Cr6TKaIEUYgGvDYWxxN42xGNt+EbQeKyeTCc0
6+mGGG55635v7Ds0484WEPuoOhdtfh7nNmr05NvsL9NhFQNYLbG9aAFZ4w5vMj5zXxIBD7JSLmh8
2CgbZYAj028n5sKAlJbPDInTw/rfLbLOGylZoHowsXId0fDC7MrQu03eYuBhKi6QvOy7BaUNti5
+ugRB+p+1S8wYd2FuPdmHG1a3JKackozXXzsDijf8D9KhLDCUm5FMqIn3HCCQDLciY2qND7nIHLDM
DpTdzf4gkH8RhftEZEja104GENDX8xUt8q8ZM03F01ZdYU2m1IefQznVmJssIqaPg7te/QdmVKhX0
uzyyOqUwf9LszOzIWO71tUIVdISv6vN1bWkioHDkKEL/upfCAIhgnCTVaycaE6oRWXRmmUPrOsQJ
IlnRCG1D2lFtFgxCU8co6AUsYwN70dXJZWb9DQW8j9/MlZyQzPefYPu9ie/Ly7QqnvmbJnY9FrPw
3ii+xOBBp5IyAgRUNZ60n6GJr4xoZMhg6Xkd0sIXC4bxYl2IKRDPV20ihNqBRomme8X4qKRyArU
RtiCI7k965TtbG84thU0shRFapnRqtSWqJHF3DlM9Aw9V7p9cHMD/bhuPXvhPs4LRIaorYtVdwNF
EpgNdKFxUmFTOSlLQhOPYktWhaxh+sqOaqaieriuA5ZSWalZz5obrSWPOKthNYU1V/SKpURwLWIn
IER6sSgGGDzCkFHRpaFR36RGeKvmak5DHxpM4WHOWwQzewyJJK2qyiVJs3CfiB+ColljKKWtlWoo
zFSpoPiQBSvF2PxiWBucCk4ujOG95bRH7TGlNdm81GOeOKnQA0WCT6Z6nu9Qe9c2KsRW5+0f02yw
c9ZSYNTdqnrGNTlJMclA4UQ/pQm2VFpI8WdH91IIBDEoMGmpcZvTuDiKjhmFYjFrTsvLqml8R4
nyTlq0fpJUpkhSRhCVYm2oRJhAm+iTPkWT16+72nSqNQFpvQffPtx1hzODbncDZqYnYOnWTWoc+1
KIYGjIFE9tyWPmDeRp4Dr7AXr+LH6/awJ+70xwclD2DGkkUYRbxmt3mjd2ERNr6nFOs9aFrNmsrp
vw6xCNTWuN50jbyoB7AGVSAX+kVcN9U65VMbJiirh7194wJkkkkxhbkKGD8VZVmeoLVbKiwAghuo
YzIJ0E8A5MFCp5rDYN19xi8qqkguOBolTUjjjS5xgWbaMn7U/wJ6wS3vzjIy5t7buDLdBWSb2/I
krLeAca2PwpCdeor9yY33u85497y7kle5C1NmokJydcZtBaobS6bInGvT6MD1RJ5TMI79G+pOeRG
al95OKA9SSyOvBcJVsoUpHU+0BUfbjTqGZL69XoATRIlMKeqZ8Y0Ea+kW5UY36p4bRRWMrBI4thh
qCygt1WTsNWB0ALMMUrK2ccGl7Cp3+Igzuo5PC1rJazwElV4vwx6pKSuQFT1RIceqV8xmQbdXs
4MfPTnMssG0oEVWGW8RqWMp9MpjRRKsWwlw/9hUZMpxdhNWIYuYgMgg8mRyl+8hwaJzS1Lw1B5IR
RFnE7Blyd0OZ0ZvhacVGLRrmCNK27hpRXiQ3DaxoZeqorIEfGfy2F7kCkk/5maA2W+JidnnHGNjW
NvYWWG4CG0nZ3GATHTYxwG++J5mlMkKJcc26N016dd6pEgy0Wxo930f+cfs6jturH/6uv9Wd5nc9e
7CLEBMJ7JHMneoChJWVGGB9Wkrkroxb/fq537vZhEEW7j2wt1wQNTChDwvHTbjqz+cbxUgwfq+mg
5E8eoqCE1izZnUR/EaU2/fxMyk2Ev71IkPYxLS4zAlJK4lqfZsMZxQmZNM6tBsuXS3p+DEsZPuPa
XQdF52+YY3UtuG1ehFP0mBoOQmRAzcs10oifOeNkXXJgu6y35elWQzFM/o8v9piPKJgDT15zFkKN
jM92lpOcUCzLkW9XqfAuOIW0lKKDMgFhDjlpdv3+BgNnOnPTKMr5tTlpIjOIUSpvkAhi56L0zuoc
+Ck0mF9Sm4gFTODfIdxLz1ClQXQqE4nHxHNuBStsEOWzrzxgNifoiD3z/X4OVKD+6NQpDCKcEo5E
FbkhdFp1kt7ShErzGw0HyvLVhdwwK1oXD4HJcz12yGn3nDlShveciN9aYbvvaLzts+mi4YEUGYqa
tZXA5sqOcZ6H4yEufrlpG7YRkuOEKAJqp7qA4sePOWn7Sxpto3S6Vh7W1cyQJl4rZWxJ/TMFanx1
DlZTWW3ZILHHqrv8pWsdWlJFKgaqk2jFI5iNqrd4kqY/bTqzCci+Bxe0Usnuvhv6JuwKYxJFBxM
bEMYQJn6jXHNZN0sFFNfw6RYi0psI4Xyl9LHKf0U5SFUVZ9Pb4ZzxkYTZU82FGdUqqxQsboo0/2P
Q6+HtSkg+CPPhI3nMM3vl6EnuQTbk05MAHOe+amXZ1ahj8FnKJEWoivVZmS6aRychBBn0rIeb7NJ
mMUvFGwtCftlAZgyuKYXBK8UxsjGnQmsacyS2+RvXD+EnSZzBJ9JdJdhszxkETsjqm432YwLKnUY
OZZpOkSRvlzmh6ZuZVv/lbv/OFh9RY//tfvlaNhsVvu1NWJ+9H50tRUjYJSVblyi7UKQAeNirlCG1
XScRA3NPcTVdkRsqkOjQyK0+BMEGd3UGqSDhlfL9lKKmrH018VSV8mym7Zg/t2qe34zONW9bKJkG
5vsMp53vvPDOGQyzxmau7grx+CU7dfTaQCWtIENmgMKU1V/QyvjCATXyNTWi4aSX66xlMgVUzQj/
aolgPgnow+IjEvVHwIOr+ipoFEA3SctIQZnZnmvEh6a6W1iVFHxpxXaQ5TYGEDoxqqY8dofvIU
tqwvLi9NjHMeYrIho8NcWgktcHjwCCROmJMaVsvXoIe0NhWJCx4SV+LguR407CftN97r0DNmUf
DSUhljgQNdC0Gw6Tel2miElgqdbmQNorvoyAkvQDUCMQ5156LqO2kBa3JBnUBdu3a9a3de/f85T
/Fvr6jxvr5z9+4273bnUaQGJIVKVjBrpQb8a0FI4ykUu0eQaamqvNUjbCKVKmbUYJMY4bhRcr8v0
mUZxqpsDIjHE5TatGW4txlZOXVHFBR9fNzlp4dtXnEyo7L2ICXQ91s1InUmwDtZh0W151B1EcwMT
sJRz/9RkicoVghpicymG+80JrVZclNpA5BoWWomumDJki9MrCjghA5G6kOQ/HcQil7oEgXPL24CP
vOOy+i8xWcZiY8pBDPAZOCh6cxSlsktIBokiLlGWHWRDhdKu4Q+pnaBMvzToiz46TK8j/bbdagZ
BviORQ53YeLYBNeWUj+Bnhsjf9Z1jQJEDTVJKzCalK1QmbNVQA8kDwGs1NRu8IQ9JFlx2q/Lvyo
6pFyY0cunR4Aa2pWiJlib6qFHYpmnRsPk0qj/OzW5MtNov/Y3/67ftQ26sg27xc9qqFukgGeaIFj
raeapla43yQEm1RVJo5gOT9k7UWC85n0Voyag4YhnZZqv8rpRodw0R0iFsqktr7iYnmR4Xtw5URd
ojqaTYzDuWLQKQihib3a5S5u6nGHRMkC4CGZYUK3HC+tN6FJbUKnePXQbL2IJmSflaXEaDNo931Zj

RuUXmAyAM/yK5K0ypTh6pUT9+jU1yrRH//vnE+9MSp4zA/O0eHLWQLqMeMr51AKc/iEdGADOUKBj
Dft0S4tVxW4GbWLVJv+ZcosM57/TWh4njaBo8i6DKvv2DEZ7guzpuQkJL4nrMb68WVK78XyrCzd4u
0MJkwkeYDTys973WNKadOkKuEiZBzEAnz7kaiZ1ECCwPFlXnVGoFPWRMUPVirETeo0z4r86gaT+m
lEdN9Patle3j/wT7Wv75ixvvbPXqXcRX65i6Z1Lbs/SZBb9GFBVgEW0ue0ssnJ16Up7f1MKQWsqV
ZhEvHFsQVWtKtA9soP4Uah4WalWr+l/StSU7InqAySeymQ6gLnsi70xIxKjWqrO2A2pcZq837TKv
kgob01st6H/kYf5tsn4cTNV0lfmdUJS3aWiQmstM9eo906Mn1E7S/i/3JdSYfcg0eRjnFSyJxEoB
QTGk6ePE73AKdZfPbAMixQkdXZika/volPBQUNaa4sD4D7dga035UAjd+GMKJZ1UF/RGkkHuiKAK
i4btEmhmiiylaIJ9nxnJIK35XSZYgSiM5Ix21nNigsT1mc7vlOk/P9SWxAOybpooAJRYu4DOFz4
c6S3jec/f9wdCd5LQPzdYkGa+L0c5IjavFW65ubWgzLK79b394pX7IjfxWr9855dKoDnFEDUdIZS
MT4HVipTKLtSmT7IWBFOkY4co9BGo9JxhrrqFjtQXsFkNBm8f2veGN3EkXvONoxuCRLgNhjj1EMq5
M38b/lWijeexO3d2xFl3ZzLRvXrAlR8gBobAzgcZ1JWP3K37mkYVjuj5HPaoKh1qUCqXroceAZkU
NNBd1G3irv1jaBoxorKPjU0psrGuqpxZOE/j7qUY8JmQorB3pnA0GvmUALsJtWX8ptDKodtPFcKI
AgS40R9CISfyoTR5LS++fyGr8ox2k4pBGaWjo5XS6G8sh8JGBW2W+kcmZHJazFlevxctx3almk8
Mexyq8+B1GzHgTAaa8/OcaGTG2aEi7yv0d99Bi2js7M0WfCwfN0SE0ev3Rmum+93/Fxr5jxrq8uP
FEd+FboRdHq0gNb9kybGQhrUwgGHBIY6BcgGttrVxeJdvKgrKChQo9ML7YgWgu28XDKJQWFBjKZc
HIjMFUxM9oblqMvInVknFh8RvMx2/kOIFiM7kjImTgUAD2iOttKO76MKj8LMuxxiCUjKOZsEoyYi
MJkuprQD89k4iOEfeEkyozS5Wann9/6AjRUDfWluHCCy8mwIc0fseEC15YWOx3yehyYbUHYMI9h7
K04eFvVd3qXxAhoHR51KbychC44JaN6jm791MBJwV4iCC9GyV8ji+UpEMFznFxShqK0HoBY9fB4
j4xOMqIbEUjddZ8m0/r8iPlMJmC6mDDWfM3IOt0foQRptbpAeMihaYOXQH2ih4FuWaPf6d5wTY3
UX56XustQDVRA9cOKnQDTyIijUSYGVoeFtoMaTm9SqQHSGHijPGBhuVYNc4jBA+pyYZQoJuChS6
JEiXvnnv5LwiFB1M/XKOO7TmIDTDy5AC+Xqi7HHW/vxCN1PmMIjiZJROaEt95N5SdBLy+zMJwtAu
FBBYJ5UqK6fronkQkbqFIIPWGD9v5I/9AGDaOEd7SkpUNBxPfkSY6oCzt3w9TkDCOVJnJc0gPFWs
3EkdyUMS9Z3X1bGTUTIoZv9YIswMbWkhWZFc87Ho2YBTUyzO2mQWylgpqER2S9hIOOM7zsnJFTTO
9RpnRUhPiH7XJEnLOhdLC2JEKgy/Sfw97VETbpexJxSniPazT8jJO4TmFxsxxCLWMJUhrnLxGIH
VOMDmmx3lvwK/njHVU6JF7jeWH3Fhf9erfUbnNn+HeZiQqe/BUfnMbCTvbSFg6IdftQ6NDpArEjE
S8OEla7kONU0kFlWVCJc+z5P760ViTbG6leJ0gKyAVRiVM17NoNcP43tZxiI7TMBib2LAVLWH/c2
zIpZYQRNImVmYhpyZ4yJm2ZnsHEQ0PaC0Smf45Qr7JtWYSzVHSAHcCAWnm4KtCzRuAGffcg70eHD
9x1LKy6dlZ2LmwWEOtkYokOT7iUmjGF0wpzVmmkrBpv61/1JSnKACBWuFeUZvHirxPCVARKaZAHj
Cygmrc1+XdqWGAQdJXS19rTTIwGiAYQQAo/YfnVF/ad8BMJ5uUgwre4IMwn4KwtCx4+MRGAuEyyB
/9e61TQvxyxpFEuS2m69BgHDzIiXtGDLTQBqoULUmY4EJ01C8suyMI6nGXJ8CE31sXlpTrulaSWg6
jbl1Mxm+saOmiPb0bZe2Ej3ku24GAvKtQH8e6ATCraykY2os2mXapMpxvrwbbqsBFplaRikfQ3xf
tv4tQlmvIJ0deUwBhlCcZWBtU3ORkxQC3PnteEBeyYnyElCD+kHdL56H3g5ykk9U9FwQFvfqk8ZC
+RI2stEokaNJKmhRapyCF6R4CYwdnVZVg+s0RRftdcLMzvlKkcGSanHaV8CKl1FGlVkdngQtZGmm
gxsgsrIEuaSi6dtaw2b6P9N1aG8kh20nhDJRE0wyUGil3XMPvB70kaS10CoisCSycnKArmWjIRTz
kEXj1ZZjleTE8xywrRXlNdc0mApS6F3EgOLOqrgpAofGvI6/5yBK4RkadW01AzNcA5KVOWSiWesS
Gyk4omtDIFIzsgMKrImae/sbGBGYGutWr6ITfWYjBsu5SSjg8ieXQjgVsJlCpoXpJcRknjW/FktM
QIScq6ixA87KuZMdJDKjfc2Aoxu+LV0YgzU7ztjF+5mFb0h+KeaSI3zj+3S1XFSdftsxlDQkvvgf
ucXrE9TpGnyIiwWFanUnOfdYVjCGcmQttMWHKJOBefdnAvCpUmDEoGS4gRfGfTBUNEA/W4uIipb
9YE05Nz8Kci6oxGu4dn4ZhqKFN4CEX3E6jGpJJEUW4nqWTSWTtip4btoqRUSLQC6PiaWnu5FmEL+S
H2DtZ2Jg/yMJSxuM/dbLe4vVOU4n9pxJRIwn4RRACriNPKiEKXBoM2Q+Sy4ZQUVLk9LisV0V101
Q1TD0o5ksqPL8k20paVDn9QqmiFm8IoyavWylqOWMeOaLDXV7HkvyvLXH8jhjr8QdPLuCWuBhype
RNm7BzU8mYftavynvgLCmTTckYiFIOIUlZpQejGcV0S5QuRDbfM8UDXiSUcvFFRynOFHirCllVsr
BvJGjZaiM1SrQpLqpNkaTtB6tD6wbSUGolKBBAZc2DHDMGQLQw3kJerlzk1FUHGruJzJnway1PFQ
FXduVwAegKC4kxORumZnDB8dISR1PUqZ2dnacI5dszYbInAs+sLlHfknrH1EL87Aiq4D0euRwUs4
uAEQgg5gE+T9qoKAEWnCOAjgzVGSZGenRkuRgAoTfGq11Vr/AaTbZUyORrgQZetnbyQstrq6jVJY
R72nJfCymwdmel5p2rLUcyY6Apfv9JtJnBs5dIQ4o42SnVr7o/lAGEkSTQGprYg62nMBw4Y812ps
7vJA+psf7qr/3SfG9QXO8+yIRHAYFMwuhwiL0HC2p2WK+KdjBtHMeBg3VXlSkBcWlUrS7jcX5pVY
0xFOqf/Z6YqGaiBvt1012blEqFNurLxhulSaVCNp2jq457uP45/U3C1Cuulz1BXlHB7ac9eBjZqK
1pibzgShhdiqmYEHrATDqgVba/lvD4mPI7alRM2IgoYFJutvOqjUbSylOnF52H33CRQUP0vQmXZi
IK7jOLII+iIZoPVZUFwkFtn8AUTamwz0oIGfV9SrnfKilLjHK5sGxOVyyUrXOOziTZog2JdSPIHs
g9puvIqcVr5Z2J8mCfSptwJE8pe7PU41SlcgYGgCyrOvBQdeK5yyglDgAR3nrLIJMPGNamm9pxyd
jaR5q+QY1REUTz5xedOt1n0ZZCx9aopTT4b0ixpA7dQQGTUx0Mu80HzFgfPHjyI+7D7KXemR5f+q
TCOJffOKasj6qadGw0zWpGCuiGD06BazDSXIwzAa8d6HulrJInKW5EDCeIHedibRbaGyA9OrzJhR
qFHatIqcHXrm81cyrSKypVlVWTW018slyNb5nYcJB9xOd0UiISKmNQ1PDjXp7skQq51grYkUbjee
XwQcynLtsKWUj96PDJ8iNsY+BjZo7VE9EgqB8bbXmPa0ntta9siQfEKXKJdFdlakiPyZoqc8j/nC
ggq8ohbErImH6kLGWafn5+HHJc10bbydrQpjpUPIZqFZSUWRsrxbBRmVHK9xbmBlEGBUklRnutbKo
iUH3/j0qSQdZ0151LEgtqihROysaSSgZG91GrUaF4VXwodDg+XsyJEU5Z2UV82sxMTzfquvKd93U

NirG9/198lrlZ9JCuVJwEtjCdBtGeXyJKgMJkiS5OUyFsUsmrBETpXDkkzhMdqTMroiMVUvhaVxW
LOuP/EFQWbEEpMgQtbBPEsK4hlIeTseBA5ECaS6uRn4lFj0e0pB5dTadMARbReli5VVSISukleHE
6bsk4Nv0MMqnl+sQ0qFzDWFkorgIoKNd3KyjKsbaxTutlpTdTlpS3gFJEKGRHjaluunTCbVf6Eu0
s4gLIi8lpsWYaa9DZxeRcKhHlRMasyAopGxcBFzsmASKNfwbExzKimkTVFOkU4pdIQI7EyVmdZb1
n0myhrs7KAWcbpODvxI3mqgroTUOYBRbnO5UA/S5zy73lYaUK1tkReWvWSSJ9e88Y5zFjwfPhsEM
uphnNQozo6YqQXDMkcarsi43QUNdNid2nqObSbEK9rNenlyavY3vnbgs3/x/U/8Af3PbqvwefHLzF
v//r05ekda7UepSA7xIt4K+OPRW18fYeViY3TKL5plUKFGec8PTDNPM6mMNIHIuBBNjzaK8cbyme
m5MvqIPCj2MYOH9emeHDh8zVSU6RPxuuMspcD1hXL79lYtnSAnArKJDMpN3qF9BKZCkQt6wyIOzY
7B0wvVmMiZ1y2OEGmRF0EFwOXlZXIK5+3dD02RFglrIHQhYErBQxaRQiA3g2zgPXObRYWLYOHrsi
/HJ5VBbifKJDz6rhrVrC/MCsixQRblXzjtK8bpPuhSdVJlW9M3Y4LDoqfzWBZ9NyfQMq/4XXLZ41p
QsikND4QgPwcH6LMGjzyRtiudVD6WP69UhVdDQKsE3xFV4UAK/hlsTyFH6uVWPk7j3vtHtQ22EDr
BLrZzBKGCucg/gvpfCzMwU1PESTBAhp9Pd6OlWl37qIUmdG+3sw92lwX/Ca8TlDE/W+ZqBARoGYc
hJHzUFFfHcWiuRuFxtj143oX4arcCAUp2f2zAy6A38PAiVFygCXYxFSlMl5JdSoFyLFVDtkXIKt/
UQhBqbcY3TYRsvdvZaPqo0RrxhlP6B3XLMLp6D9Uh3DGpAtBmAwKMw+Mxm0+3lYNVfVARp/NB4rO
ZoovdDaToZfS7lKZR6HGhglLWMoh52wnEtKcnvtCEeOcmKt5nzfLKBDMcTs9KJ0aIpoj7ymfBtn1
wU+7FOzQLvlvKiUrCWhCitqr7qOhBKS+g57MlNFKg1SlJcLB5UW5Fpz4ngYpF2JJuo21lYURSUv
dywigh64HtF5ZcTVlayztrMLUlVsxkCuM5V2lQ2eD5lxFNEiEKP22m3PdW6HpOTk/DSBdY3yw8JM
b6tre/qd7vDf8P98EJotU5i3BXGEiUfhr0YzWYTZu8PPwNwj6KjYEuV3QPkKJPdxuTZXtTibtFhu
KpNxmQ7+9FXF8rbY1aUqZNYGjyJXHP9lGbduzQaDdHosQSMEF1lRwwkDTPL13CJ0TqGUCw3wrpXE
Xgl9fPtaRWn/LhHqO/BfaWSorQYWF9fRU2el3Yu/s8Ojy0TFq2wJNaRrRhjUaLLWixSkRvWYUeb9
BwtgziaFmCnMiyaSVKCibUyFH9jBmMsoHioQKs40odGRIHIJHC9w3lncL4nhaHgk4i5XJDKaYiEP
Eel2dgSq55XSQAov6rmmLw0aO/gaFmStlivB/tVgNARu882IQ9YWO3HnUstYBj0XdNRovKfKXB3H
XMzGglhrv2qP+M/27HwhwMBznpBmNplusOyIDx+TZ667C+sQqtdg0mp6YfGiL/m974tte5m78jTF
6IkDdGVm+AJMotiC2LDpj3pNxoHRPhx4XMYrL40Ac4Jkqg98qd98KLh1KPyhf9UHpyJgvyYAPUzv1
sRmmhEtC2JdIDMWGSrTJUw2didJxW0fAlfScmAvRcMQ5obcm5lXpUjHSNXoCyNV4IJk0SmrLfi9+
BHvLA9gS0ZNNp9Lu1lFb4kIOSBJ1th7Xh1ixj1LjWJ0iQZ2xsje2oEvPKzt4+mI3eCMVUUHPL7b
EsGoFEkEWJmgdeCzzkbVdPUyaVlJsRWFUCWEU/tQICaR6fk3ZOSiVTTWZOZYFWQUIzlbNiI3E8iq
Y00zoMLRvfnt1Kt0vJrlrf4vGIbx25vqmM9ynOXngbQULIdj50n8OdQVy2hhnO7PQkdLtdqCdtF0
kHtLkvUXXcVIFnQE9Pz57+ZzfWZ/7bZ0y68P6zaepKa+S8jggQZ0KBYXi9vFDMBMLNDhk2npNyZM
lHVn+QqiPeI5XN6oU0vq4ggn5OdEXybpob0lhPcAdBhb2o/iDx6kUdRNI500oBmqmsJY1AdUygCr
YE9f8oJTBvXZMRoGJBpGwTI6IrIq0RuziXsDEl1X2d2k3UBFy9dUFkFXFTP6zkufXjzJ9enOfa5caE
lEEw2ppBTBNmPtKJ5o4e6skpG7sHJEDNA7iJRaHVpaJaaqfBG21acBgWV9PM39TGr15NRDpbaFoP
K8/a5wxjoTtypUqYoap6l1kIwMr3iktSlJuWaD1lJAeY20tKqMoLqblmDK5vSw0jGJCBN2DN0vVF
Dm92BSLMtKQ+fOWXQaPCoHLprSZrysJNDYX200cFucK0nqOJkzhPX/j7dr540sqcJVdW93u3vssX
dm7RlvcNnA34ALwmeQNoEIfEPkAj4CSD+ARLBagMCMn4ACdmSEBAREhIRgxfxot+3uvvdWUec7j6
pqzwy7knelkWZ3Z+x2960653zne9x3EO+zL/Ke3tXl2dnZ1bMf1vxR/WCa0jEFB5MnKh6SSZz0uk
4+XNWRzgxUkFZ3bNtkrbpWUBWNmSqRt8RiOGOJMGcVBHixA6E3fQFmSHK64uWoh0EujygV1rWWLt
4ZL9ZAJbGMKNK0hMoHQU+VT0etaoipyhj1ZpViUR+Vi7yBSSLdKnL2ZCYucyxXU1OJ5L+zvl+769
sbd/qSWUh0630159UEgdOHKQBReMKJzZRgpxyq1DUOmniYnWfLwp2m2r9KfZMrcX9SNlVsQJ70xR
RNpxUKEQIYX3AWEeYXTcTnT0eQ69Wh2CKVWlpOnNnce7BiDUzSnB5UdUsWdSuxlqIQeXrybtP0+9F
kaR17naRyG4N5iGJ9QCDawjSp3nHO0KHyqaDXDRmpHyxecGjfrMbvOejZCWC50+GfNf45aYsjs5s
t97uR+/+Gr8/TshzW3H99eLl90+IFHNjkmboCqH4oqi+cv1FaJKXhXTJ9+aDonDFJd6YYi9UYCD1
HfxqIRpVt7GEvEWULmw2aZba14q4xRwj2/3k7ke504KbZodqqqYXQV0cIchQ7sS6F9nDNIqWCFzc
9S6UMtdUsNKcGLDab2x1FmsJubGyDeF+dvQXCg/R0f6mR72FT5DJmxXCoaWP2yGldi8FmoHRQPNb
rRVirqnBETeUA4OlKd2Tmhs8roZWcMt45K0Fcc7TuxnuHAMvgkpYiOCXGLYv0J3Y6sW1B183hBB5
QOoFASWcNwxMhUmbbFfhgVQHfQWHcamLAPR82R977DrjDbbHwT8XyNLG/JYnQkQkd+fUfcIVJnh9
QDR3tVdoVYLcjSZQGizzB0ap97+/VvfPPt4TBNF+4f+73aUJUYB9FM0rCYrYFTRV5PEqvwepDk
v+EARIgDe4WrrrgdlPbLLWwyxaBcPRblO4UEzlwHfkmNAVUD2pMgQwkrzF9zWzn2ZPALC15hIX3
a5qc13LZVY2Dgxlj/r1KdAMXtcelKdmgtPdn+a8F12loJECsMsCgZA8yl93beXF+5ocwgcVc3Ds7
Zf5IdKzhdzsCBR0+iiOrzafMbA2ijrNV956mqCdx5yksY7SmQID028McwaDTCCQTwq6SJ/JtolMO
gzudPT10Y0UL8rPgiTVczaLhatcIzi50/EGjQ/I/nAULWC9NCFBhfgDB02QI+pliG5qRBHYkf+f
ZSYsPqaGmXuK4SdUVDlVSJKNS+h3w48drGmRu7Ec8jEVKOlyt7nold+ZLSzXcjHBDpR1wczf7y85
/94qtR3Xz88bf+9vfp//GY+rCMwlyKFTG9tprkbUQQd0MvKGLxhUfF9AXQSMJqARc3f9qLRYlKZI
DPo2JxEnZplxVg0YV6nEr84aEtaC2949c6CkQfmhnRveobDTSC0Vss1dU16LNYrRAZIA5FSVotJA
pxn78uAugOnP9rcf3mYevu13e57T0Desq7U42XZNCHZXIJLhFTGqyy1txfvbhozhqrKlda91jS1C
0VPjaOkPyWEOo72QptSml+LTv8sRsghBEC3Nlo43nlwa+ZH/ootK4oMjcCkHQnmir+thIR6D2gdR
UBi7V2VQGnGexG8qxJnPB8wlCrami9YgNGS+lsU8BewYOhvwi6Jg8lVySBfHA99thRL4WevmdgID

```

FX/D2RbbYDYj/oe5Nh2hzMzBl5ND2enKw++7Jn7r0O6w9/9P1PNrePnw0xvurHGZMJYjLkkaWP05
OVh7OcFy/OA/DUQwsUdVaUbBvegTlD7IproEjIzCNpsN0t3YqNbakIprl3lpbuKgICSBVFXyHKHp
n7ojePoig2A17I9LH6OpZzIxYdFi62bzxK3ICQIVY8YrlonBKN0zFGVG1N9RiXdleYaVx+fYtdw
3mWazxE1wQMzPDJHu0hG4cVM+hUDQu9HOuhJDZdckSCgx8oksT2atoDeRyKvtGrERISTYIZiBgDZ
QIwHm7M97dAljysCHFJRqE4SXewnRQe2kto4gn6KEfQU7xqJ5EyaQHH4LvFfH22y06MFD18v8nSe
V+ZHUQvf/q4KCBUMV3qZeozkP0N6AdVjR+HIZGfGGXHX30+9HNFlxRSVgQxMQA9NF+XkzTVERoSM
QHiRx2+8fcbh+DPudg03aglc1if/r6g389+2H93ne++8nDZvOn7Ti8AuWKPgyKqxcSPCDvyER2Iy
eof44vbJmYilqkrkxAaiepGhAYVulh8DJOxVaQJBcAOcmp4oZR0Y7BJB8tFEmrM5ziw1RsNOGIMN
lBYpDGgTIX1DYzsWOCHvyk+0vfmyM/VTe6YVUIbbrYeODRpDacElCsraihTSXXl3fgjL4MlfUWd
AQrlCMxDD/i9wv7qQ9D0LST4bmJokMa4Xl114qqrX4XjuYZIABOpgaiY7JCBfs4pDgfABWUEIAoI
7eFzVnI0XLnGNFdi9LmbQ4/IHF7RPZlXp2vFftL2Xg0rqH3ofH9T2+N3UVdGFsNmujlBh88U4q/
yQOCxfYE9T+9K4HRgoUmKldwejTXGkZP5SaXKTu/nlgOqJrkwtDOgNpssSjd5LgThERlHB8Pw8G
LJxSowYSPkcXGWq27uDr/+uWvN896WH/6kx/TPfuHYZpeYfjXuSby7ssADqfG0NXM4zpZdYg+ED
etN2pZ/eZya0mzyMhrELFuUVKEInn0oBU2y4S9lu4NngQyGQLsK6K+sG6eRDkqCUHmlS764h4QKq
+eKFURl0fQyF7jOVus4wEC3Lwfwj5iFFYgStn/Xd/dufl2A6PthdQH+Dql24vJddnTkrsc06ijrH
Bq36pYPIwaV8hUWFZl+pyzsJonVjfllyTgEnvl4nXI50Kmsg4gYURlpUPHZgyc8UoPNy4JGWEi5I
d9Ywe77vbu5m7N9NETNhb59/TL/q7FOoED6QZI7+Kc7iude6HQvtgqjlWCMUoXOBULjA2xHNOg7
y8uEfV+366UOglscFbb2lxlPYpTEhOWKb//3l7ASfCSHhUDZGeEzt9w/xj+/T0X6pwzrsp1kXZi
9IvUEu7OSdQw8n0anyFGLx8crTMarV/GE9qmx+UztzafJPrnG7YHBjACrDudCZarlWwtSX7JAke
jmelATucpMsi5pLVucLwbczk1PjMLrdOzAG3qW6HmNp4jMspE9cYq+0ZFahunYOlbu14ECJmrcnU
RZQhakNI9dXV+DaXN++TUs32OdGgfthN7zoplowJ7NwpJZBIL0GDou6tQXfz3NFoWkL0u3xUsLZ
Yg5oYJBjcJlZnLXE/jBM2dJBf07JYPH6LIwB3ji/y5MJsrFEUPWesIlqDuhnebjVvfc+G5vHjjlr
mKIV4lFwgGeY5M1vhO8zrfpqfX+/6x+n1K9TMgSXqQOzoD+jj2cTTfJuiZqYvc7RpVEmEaz8cnON
xkq0rdBEgUPae976hlz98jn6F9nm//OyHNb+wOc/XM3eab7u7XAGGcW8WF1TZeGnMQJIXAXSgG9
Srj+8ks12lK5VZFwB50mL2qDhDRXZPT/yXonGG89+j4FphmyghwzBaOXRO8l4avmzjrMhAR7HwbO
duTl/zT4J2awkVCw6KmfET5374T0VBginSv+jnfHjYuNv1xp2fv8FhBVAWfbFsdawyoUwXL+0zrW
3q2ZjXsAPEk4AK367QddNitr2Xe/Udd5VuTdQ0anpaymVqA0mj0zsoeUkH1c+Y47B9QIARowwD
ip6jKjJMcCamWS7MUNAwtoBiURwuNui2cKPKb54rt88xE0JjSuI6PDmHf73mxknuiLjfbzBJIhYUQ
7BzU6t35aJ5J0kpftDuaQTsIvHE2w0qG0XwQZdIEiOk/3x6YsVdle09chj4udm2kd8XqsXR/v85/
797IclhWGeL4guisyNZg+6DR93DzgMlKXJmj2Rv4VOBL3iag8kgVUBxHTCGKu3v8D/saGzdZUjfw
gOjPLMo1RiVWnLnpO/nx0WKFWmAnSlCmQqIlW2PaVDG3wT3mRZOF+QdaPQv1VxZPukp5am6CZUKJ
3nydyVXNl42b+6PKSfw6hQyaJIIyCnncakxtWl9RNPFGxorxVC42Dh+OUES5g7wffjX4k1Nw6B6
fuHsLsGls09hTLeAMeuIY8J6VYJvNXggsIXQqR5WtUQB0YobFOtcN8Tw/8Xa6kZi1Kh/hoeeQuPj
zPVWlu6xPqtujZAGfEP2ex/dHqfRjfmWKBnQ/MJ/CM4cDRIUir5YEnETMUCaLqkLUI81jEe2D672
A55edl9zi5TX7PVvmbnRyfoguJy2oYJvfw+IBQqtVq9Z/f/PZ3u2c/rIvj+X6/2U5TLPA3/SCkmR
zme3AgaccEqBugSZIAiTWfzGZx5JQ+1T0gzX2hov09Ky3ci7+w+TQJiV2WhUBKiYgXt69dqmyFti
alI0ZzdKipfkYxq6H+agXTxi8kyzppb/nAkSzNyig21evhYev+d/XfXG1W7vLtOb9vQXaYUY9dLF
raSkfrKr5vE3Q8OQNYjPs8sQ0oHlydfSfe/wamHJavUSqpa6iTvvgWNUh3kJGGVVGaJlePNGyc53
n+dsWPmBwT6CASYLS+uUWLSd3Z+au34NDS36Uq67HXXwhTqzIcVw+Zw2R5gDpFJF8uXdkIhCapQB
4ph9iZih0pjzL0vM1trDNwzYnLouOWnYKm0FWwWwJwv7v1BOqtK23E/Gjlbm7v8rkASLY/Xq4+de
/5z5errLmmkwmzBedW3FhS4RMioLaSAY2SzSJTdpf5xSP03AsTyGaH1sVUE/b133HLTWXXGkztz5
XRrFmszZFg3Cocqo6gUqJErHbChx845qwgLn9aoXR+rTx+dVZNjd1oQX1bYIYP2HWeTW/Wt6AMXl
xcsA2Jk0ofmJFUbLu6qmVldig1Bbx1fHCiX4QbhpeAL5kzObSp6I2ZkNBVlTgYaScPQbevzbr98d
j13DExLjC9s9sIVVaQvn72mEqWJLB9uEfrC/ZPrqgfnL3mAY72o1j95M9WP99UGYmbprlRUgl9Nb
L/VnGccEaEUAZU7Qndi45YP0tttTXFvRdRUC++VooG0/uO/T4yiga0ywQkIcZ0xxTQs7Mz9/iwpu
fmcTlf/PV9D+v/BRgAn4GX9JL8IrgAAAAASUVORK5CYII=);
background-repeat: no-repeat;
background-size: cover;
background-position: center;
margin: auto;
margin-left: 35%;
margin-top: 15px;
font-size: 2.5em !important;
}
.loading {
background-image:
url(data:image/png;base64,iVBORw0KGgoAAAANSUHEUgAAAOEAAADhCAMAAAAJbSJIAAAAgV

```

BMVEX///+hpqibi2meo6WYh2P7+/qUg135+PaXhmLu7em6r5iip6nCuKTGvq2j1Hbn49q0qZGmmH
r19PGtsbPt7u6+wcKqnYKej22zt7ipra/f4eHW2Njr6OLNz9DR09Ta3N3c1svTzcDox7jEx8fc2N
C2q5TGvavOyb6QfVS9tJ7p6eigABEKAAANLklEQVR4n0ld65qiOBBtDBERr9xFURS7Z+X9H3BBW0
GohFwwYH+eX7szdrqOqVTqlpqvrw8++OCDQcM2G+hbpG5h/ASLGoK/RdGYTUY1jP8aQ/3D8M3xYf
j++DB8f3wYvj/+BENMk5ib4RDpn6Y7819iTobH1fAozoMRUSr7PF/XCY70S0L8So7BZHAUT4XQKx
v4m93p2180+BUI1qtZgoEfOQbFYsOieLpti2/U/tw++SC5Cs3Lsr5YELwXGxTF013z11LWKePf9Xy
NoamrrODg+7X3ye2QnK8rBVgt8Ch7i+g9h7fNkb5gXGJPFv93jq0kePzQYihWCpW6ZR2Z+N44/v8
qaVH5qKOamSjDhtJAq+ebhd+N4uv7g8x8OgmKNYE7RNucLTn5XrBKc1KzuECieAEnb7CcJi1XjWu
nfogIEu0XfFF9OsGdFxf6GfxfFO2LAoIjPWh4PepgfisguAB9V2UUVyLXAhf6JZgrqv9iisGxX4
J59L5+KUX92DO/guIrd3GS9E2vgL3iEFnXeRzWYAA7WMBkoqiPfr6/+r5MV76/XjDxHArBr69dG0
VdD/zZMVnuTKMoBy/PyWm6nrSRXPRuZEqYVF9b16en864mrb08XtbUDEBwGg5B0sPJammDstq704
LG8Z9qGhTMyWKO/XM9PVUCm/MRWVfXA0li5Dg3s6G/0FvPkvlNpDiZkb8btbCnRF37ZpAxIQYok/
PrhWfCkbQLwYzP55fEU9zIv/YDKmuqr0+MKxBvm3HySsGZkZAIJsxLmKRAC/E6sdmBCTrGQTDXA0
KgOR7CSTwTzAyrit5gEsyx/yKpeQDHWpQFcXkT/qL6zGD8wvgPJLjiXiib1hmN2MzxK3ECv/sFVE
lsWwlUBl9gpW4B2pmJSNgDh9JB0rXEnNiBOuoLRQWwnvatpDPPREzD/YU2ks/Z/8b8kb0qaCzBX
qoi6RTgXlhg8dQ1MIDXTe9Z9vq9b7bFgqbP2gTdeUxlGGXMH4gJRWPzXeQsvjLp1/ZIRUY5mVaAd
QoE0j4kpCtCVaVX/j9ep1dBnoF0BbKFMMAJfPbkiXGr788lnB/U0Wc+UtXHWBDqTQuKRQbFEO5++
sdGPpS8c70HRhKeVnQ9TM0hnJ9BafhM9RZUqRkeOLgQTGcyi3/DgylKkbnN2AitkIz4w20dOT/dU
sjeR/O3oDhWoohudY2HIZ/32vT+dL5zyAl94fFUDyJ8fuWpG+G+rgC6NTIHETQ0FR/4X+89RAB4A
pAifREeG0bKvP4S/z009UCTI79CHum525Td11gB1kG8Tv/BKmEjOWSB5h1EM7hgv24fZdmwAL8Wk
yvMLSfKj6SPI6QUBMxxQJ7xyTjTXkY8AUmIhWGS6T9HsMvOEs9mojcyzb4Zcm5uV1gOQb11D+zj+
HGI/6GgM4BaulowZ1xu8DNGANoE57DkvFeiv/gZRZ9P13LYRKcZT5XhNCgKHSguwYmbOJoXkGR1K
Gq995rUmBH2MTgm1nDEhJBQiThRvT17IyHQCuMH8Im6ms2inhOigon4AKZtwnpF+7BCvf8RMgA3e
+bhCyXhrEitRhPgFIkPjgIIetAW9H2kIa01BXkA+BEYph7JG3eDTYDUvJJb3R8YTfeIC0H8mhxYm
RdP4PCqCuPj/JoRl+dacYC74gamqOmAXbkoSu/HBvKScTx76cQcrZuNyEz1HRypziaJsRvcneivX
y7VL8bnG3DB79c+JgsjbbvRys9t0k420qA9tNSDy7H+YqaAfZ5Tn71VA0McpY5W8sslD8mnbfv9oI
Ysby9/Io/0BJwe+NPTswdtJz+rtrEEZSVy722epC5AtDXYqX0SWaEnea8yPOnWg4W/ms2P590yOc
2mq3VAeSzzoHjXL6/BL7c1JOXLgA8jOUltqxcJ4zHvxmwKrSGS6oXQ1yFH8HVvSB/zbuqKV0idwu
IY0Lch5e08kGBZjTWAYlsFe2wYtnuFbRtXcxYDW9jiAvVI8NFDhilgY7ZZdIhTLwydHGEYeuk2yq
pXRflJGYIvnxTx89+BndE2joWuuNG4/tcG0gfNkbgTKFd9Z7iO9HGBTWQFisX9GhUE813MzyIGLg
xW0Dy8FpjEqKBTXB8/RcIMyTcnE0UFBG/NjJZwbHBKbrEzKiZ/3AdjbaUZ0jxYnOpNRRU2rzwk4
/JHx1kJVlA8AzEKerr4z/BaSCXpPkM8fFSOhJkaKWY0dOuRrHoxdglR7C3YrJO7IbWj+8JYbwX3U
LN8mSDp2eKN60yzmsw20+EPp5fb/bnR8XjelNGDi3x20JzqFkdFop+SfHxas0+LtG56pPve4xcnX
czvs+NcJvhIR8oGQEmLr11KWQ+Iw2Z1FNHhtlFfH+bUUYG3gFCjuWO/TX858+pyW/C/Xq+fhno
95N4/JH1sJj62kKbn13ywq8BDBTmarNSGc14Hsxm2xKkEcd0BQ/lr8MvOzSJiBYZ6PM0hf19NTAm
WoficQ3Hdw3wnBDijuVpSpXMay2XM4Sch6Y/rjY4cq+gvZdNsonleNdZ73fQmzNDKdEcwpvrChip
Xh1nuOdGTCQoji8+Kxo5xhHgWizb70smzZe7AGVKlIZZ6GkhjKMGwKpIsyEp/iw3YaxFY2zihl+
4PmVtkbEJn03Zx/ibdsFFUsLT6pipgmN5TL15UWAWGc3phenh2Xhke6/FvdvkCmK4+7tubDorpb
IxzMKH+Cjcu5FDEzaMoZILzg5ABaCC1I3Sygdk3TlOhtvqFiGHEk0gh1xtMSJPI300njZZ+pLkY2
izZpuQFVPLgkZESXk8/4lcioOXYcbGT00blrInhvKr4FoyaSpejh1kwqFLBc3a250ItXIzdBlk8
liNa4uW1KAVkzmYzjWa/ivznDPxHDDHK/bHpNDJJPyraN8zp+an4wBkoqgDwcSmWkLCsi2RQHM5
gODr2Ppg6XxTh3ZmtawZLTRpydT17VZ7ij2xYxsjBMXzfVqhnDUSQV7k7sGw1WBHP74jqUWgJS8wz
FY4iQRq8eg/FldGHREYStFsUKn28rPctSoqXHWIhIrDrAlfXHT/DVlhrMjSfBXRUXjuuBc7Ljb5ns
3j13qE+WrksRspgBWpC0jGJo9Bt5n6cdPY3RPuMOGMx9hBRdotj9AH8AHsk/EYAI5ZPHgCtXPAX7
pMW9MB2kNF1zwMMAssIRFUH5fs5ZME1MQmE6gaUPombC1KClG6MgEAhtAJzpLQqNagCyDlFJfG0
nqKnsHakip5AxDmYctwRdpdgUw+0nKOXYhh12LEYXkARhKydP9inIAfRApnQKNaacPo/gAMpTy/6
HWzb/FEHzNoCTshQEYlLLtg9vDzu2CAa3YI8NX3BbNffu8LcAim9yND+h9nzc+6NNI9aUNzWsDa3
9SDA9AO0aoyz81ASVwPcPE4EOhHmOLr6hJUMowQaeb/pD41cigp0sS4RN0DPuN8e8jkbhWgenEH1
0aQs+6uUHbXMH/eliJmH4HQpSkUPdQHCvNtQLt6L2D2EBH6a+AVGerOGRODRTOIXKIGKhZK/YRD
FhgIfPlU176BIW42D19ZxAlItYksBESenVuLeMlyoELvCGBogFJW5gWRRQxIBc2ow9LDIvRmBp
TobobFfJZ+kw57oalgaWBCvKl4lp6hDlU6WDqJnt4KDK91dioujBsprY2rgYmsBBZh8xLb04w9c
9aHLORDkwdUZbCRgW27suQVaKY6SmDurY2QkYMAft7YttLhgdU5msYX2sjlmiY9etSG+szdvUyiI
W3rI9t1Ka+n1NQSC0beoRoBsegPTr4XlVxQqgSbkBwMkaU2yWP71zQRGcb9hQhp1TMnCo/oDhd84
jGkeVt3fx/KZJqRcdWPbrDbrQnhdA35AEKjuJHGKo8L3x7TvmYsdZyMPPAx4u3h6zYTOxm0WGFhh
aV3/3YuffZbwIdq3Jw819cPpi5/16w1+eZpLVxwtDzvOLhmkV32qYd866NTno9LarEO2py/loK
GWz1kV+4Td2JicciYct24iJQbvNFF30HCP1ZkSwmNpAII9FmMoEJ+80yA4iB0DAJaERQgqz44yA+
r3ESC4HSxBxsc9bRjmGbwDrjxwAcKMcwXB9nKPSLDXcigjTOEHsSwbzAEfwRIGLdKggufJYq8QvT

```

VYX9X2D9EpJ/1W7DkA9kYzoc/OGR5Ak1cZN/FN1FT8vniLuyJXUonbQmrIrjLAhc563AsHwkh2QJ
QSgEO642vuwroxs6zrOB5w9tkb2BropTcyi3+MJDpst/sC28MhKxI80Hch3KmiDuDLJVhsSJ+twd
saAxrntQrZwdHfSsUVAFc/Jjj6W6m4AoDOFjEp0flgfRWIvE3tmTdlSlfdv7OcuOd/YZAJWfw8wo
rSA1adxlRk/g/Dt6Q3uNvKnDJaarBMeiAtjHt4kC6Osrmp7mv+GqZbBevNgO9VI2q8cK1ZiStW2V
rZiYiyZZeusUxQpttnw/uZWGkLX3okRe9Mb0PPvjggw8Gjf8BrNv4CMYPw6kAAAAASUVORK5CYI
I=);
background-repeat: no-repeat;
background-size: cover;
background-position: center;
margin: auto;
margin-left: 45%;
margin-top: 5px;
font-size: 2.5em !important;
width: 85px;
height: 85px;
}
.e-acrdn-content li {
height: 28px !important;
line-height: 28px !important;
font-size: 16px;
}
.e-acrdn-content li, .e-acrdn-content li .e-content-icon:hover {
cursor: pointer;
color: #e3165b;
}
</style>

```



Blazor Sidebar vs native sidebar

1. Simply customizable and responsive design.
2. Different sliding types provides the user to adjust the main content by pushing, overlaying or sliding the content.
3. Includes [docking](#) support.
4. Provided with touch friendly gestures for easy interaction in mobile devices.

5. Flexible option to initialize the sidebar to any HTML element as [target](#) other than the body element.

See Also

- [Getting Started with Syncfusion Blazor for Client-Side in .NET Core CLI](#)
- [Getting Started with Syncfusion Blazor for Client-Side in Visual Studio 2019](#)
- [Getting Started with Syncfusion Blazor for Server-Side in .NET Core CLI](#)

<!-- markdownlint-disable MD009 -->

Sidebar for specific content in Blazor Sidebar Component

By default, Sidebar initializes context to the body element. Using the `Target` property, set context element to initialize Sidebar inside any HTML element apart from the body element.

In the following sample, click the toggle button to expand or collapse the sidebar and add button in sidebar element.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<SfSidebar @ref="sidebarObj" Width="280px" Type=SidebarType.Push @bind-
IsOpen="SidebarToggle" Target="@Target">
<ChildContent>
<div style="text-align: center;" class="text-content">Sidebar</div>
</ChildContent>
</SfSidebar>
<div id="head">
<SfToolbar>
<ToolbarItems>
<ToolbarItem PrefixIcon="e-tbar-menu-icon tb-icons" TooltipText="Menu"
OnClick="Toggle"></ToolbarItem>
<ToolbarItem Align="@ItemAlign.Center">
<Template>
<div class="e-folder">
<div class="e-folder-name">Header</div>
</div>
</Template>
</ToolbarItem>
<ToolbarItem PrefixIcon="e-tbar-search-icon tb-icons" TooltipText="Search"
Align="@ItemAlign.Right">
</ToolbarItem>
<ToolbarItem PrefixIcon="e-tbar-settings-icon tb-icons" TooltipText="Popup"
Align="@ItemAlign.Right">
</ToolbarItem>
</ToolbarItems>
</SfToolbar>
</div>
<div class="maincontent text-content" style="text-align: center;">
Main Content
</div>
<div class="footer" style="height:35px; width:100%;text-align:center;line-
height: 35px;font-size:15px;">
<span style="float:right; margin-right:15px;">&#169; copyrights</span>
</div>
```

```

@code{
SfSidebar sidebarObj;
public string Target = ".maincontent";
public bool SidebarToggle = false;
public void Toggle()
{
SidebarToggle = !SidebarToggle;
}
}
<style>
/* Sidebar styles */
.e-sidebar {
background-color: #bbbbbb;
color: black;
}
.text-content {
font-size: 1.5rem;
padding: 3rem;
}
@@font-face {
font-family: 'Material_toolbar';
src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjltShMAAAEoAAAAVmNtYXDoMoJqAAACDAAAAHhnbHlmIuy
19QAAAswAACNMaGVhZA6okZMAAADQAAANmhoZWEIUQQkAAAArAAAACRobXR4jAAAAAAAAAYAAAC
MbG9jYyC0kUIAAAKEAAAAAG1heHABOwG8AAABCAAAACBuYW1lx/RZbQAAJhgAAAKRcG9zdJZeEVU
AACisAAACGAABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAAAIwABAAAAAQAAQsu/F8
PPPUACwQAAAAAANXLJlEAAAAA1csmUQAAAAAD9AP0AAAAACAACAAAAAAAAAAAAEAAAAjAbAADgAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnIQQAAAAAXAQAAAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAA
AAQAAAAEAAAABAAAAAQAAAAAAAACAAAAAwAAABQAAwABAAAAFAAEAGQAAAAEAAQAAQAA5yH//wA
A5wD//wAAAAEABAAAAAEAAgADAAQABQAGAAcACAAJAAoACwAMAA0ADgAPABAAEQASABMAFAAVABY
AFwAYABkAGgABAbBwAHQAeAB8AIAAhACIAAAAAADIAjgFwAfgCIAKYAxIDSAO2BRYFMAVcBnIGugb
2ByoHQgguCNYJRgn6CiQKiAQuCsgMFgzADoYNzg7WDvAQyBEyEaYABwAAAAAD9APzAAMABwAKAA4
AEgAVABkAADchnNSElITUhJTkBBSElITUhNSEFFxEnITUhDAPo/BgBtgIy/c7+SgG2AjL9zgIy/c7
+Svr6A+j8GAXefV67Pl19Xvr6AfScXgAAAAIAAAAA/QD9AAEAEGAACUhNxc3AREfDyE/DxEvDyE
PDgOF/PbDisP9gQEBAwQEBgYICAgJCgoLCwsDCgsLCwoKCQgICAYGBAQDAQEBAQMEBAYGCAGICQo
KCwsL/PYLCwsKCgkICAgGBgQEAWGz+qf6AYX89gsLCwoKCQgICAYGBAQDAQEBAQMEBAYGCAGICQo
KcwsLwAwLCwsKCgkICAgGBgQEAWEBAQEDBAQGBgICAKKCgsLAAACAAAAAAPzA/QAQAC/AAABFQ8
PLw8/Dx8OAQ8ELwErAQ8FFR8FBxcPaxUfBzsBNx8L0wI/Cx8B0wE/Bj0BLwQ/Aic/BC8HKwEHLws
rAg8FARIBAgUGBwkKDAwODxAQERITEhIREQ8PDgWMCgkHBgUCAQECEBQYHCQoMDA4PDxEREhITEhE
QEASODAwKCQCGBQL+zxUWFhUwfwUFBAUDBANqAgEBAGIDbgMDbwMCAQEBAMkDBAQEBQSEFBYWFxQ
CAGIDBAQEBcwfBAQEAWICAhQXFRYVgAQFQBQEAwNoAgEBAGIDcAEBAQNvAgIBAQEBA2gDBAQFBaw
DFBYWFxIBAgMDAwQFBcwfBAQDBAICAgAJCRIQEBAODGwLCgkHBgQDAQEDBAYHCQoLDA4OEBAQEhI
SEhAQEA4ODAsKCQCGBAMBAQMEBgcJCgsMDg4QEBASAc6ECwNDjIBAQICA7QEBQQFBAMEUjIyVgM
EBAQFBawAwICATMODQwLhAQEBAMCAGECAGIDBAMEhAsMDQ4yaQECAGOWBAQFBAUEAwRSDAwAmLY
DBAQEBQQFSAACAGeZDg0MC4QEAWQDAGICAgICAwQDAAAAAAMAAAAA/MD2AAyADUAAQAAJRUfDTs
BPw4lLwgPBwMhAScXAQ8GHQEfBQEfBjsBPwYBPwYvBwEDFgIDBAQGBgcICQkKCgsLCwsLCwoKCQg
ICAYGBAQDAQEDBAcMCQoLFCMTFQoJCQCfBHV96gEL04X+4gYFBAQCAGICAgIEBAUBNwCHBwgHCAc
IBwgHCAgHBwYBOAUEAwMCAQEBAQIDAwQFBv4PlwsLCwoKCQkIBWYGBAQDAGIDBAQGBgcICQkKCgs
LCwcPEBAYEBAPHck3HRAQEBAQEAEIAQrThf7iBgCIBwgHCAgICAgHbWch/skGBQQEAWIBAQIDBAQ
FBgE3BwCHBwgICAgHCAgHCAcGAFEBQAAAAAD9APzAAMABwALAA8AEwAANYe1ITChNSEnITUhNyE
1ISchNSEMA+j8GN4CLP3U3gPo/BjeAiZ91N4D6PwYDF6AW5xefVqAXgAAAAEAAAAAAP0A/QACQA
TABCAWwAAAQcVMzcXMzUjLwEjFTMbATM1IwELESERBxEfDyE/DxEvDyEPDgFro8ObnnROxOpOnZv
Qti+j8AGW/NREAQEDBAQGBgICAKKCgsLCwMKCwsLCgoJCAgIBgYEBAMBAQEBAwQEBgYICAgJCgo
LCwv89gsLCwoKCQgICAYGBAQDAQENAYowliO4BSUBK/7VJQFSffzUAYwR/PYLCwsKCgkICAgGBgQ

```

EAWEBAQEDBAQGBggICAKKCgsLCwMKCwsLCgoJCAgIBgYEBAMBAQEBAwQEBgYICAgJCgoLCwAAAAA
CAAAAAAOWA/QAAwBpAAA3ITUHEXUfHTSBPx01ESMRDw8vDxEjagMs/NRKAgiDAwUFBgcHCAkJCgs
LCwwNDQ0ODw4PEA8QERAREREREBEQDxAPDg8ODQ0NDAsLCwoJCQgHBWYFBQMDAgKLAQMFbgKCww
ODxARERMTFBQTExEREASODAsFCQcGBAKLDH0BsBEREREQEASQDg8ODg0MDQsLCwoJCQgHBWYFBQQ
DAgEBAgMEBQUGBggICQkKCgsMDAwNDg4ODw8PEBAQERERAb7+RRQTEhIREASNDQsKCAYFAWEBAwU
GCAoLDQ0PCBASEhMTAcUABQAAAAAD9APXAAIABQANABcAGgAAJTcjASM3ATM3MxczAyMFIQEVITU
hATUhJTMnAgJx4wG/v1/+Fot+ila3FD9RgEg/t4Bov7UAST+aAF/6XQobQET//47eHgCM07+XD5
NAaQ/UHMAAAAAQAAAAAD9ALoAF8AABMhJz8PHxo3Lx8PDycMabWYDQ0ODg8PDxAQEBERERIREhA
QEBAQDw8PDw4ODg0NDQwMFxYTEhAHBgYGBXUHBwgJCQoLCwwNDQ0PDg8QEBERERITEhMUExQVFB
VFRgYFxcXFhYVfHqUFBMTehGwARi6CwsJCgGICAYGBgQEAWIBAQEBAgIDBAQFBQYHBwgICQkKFRY
YGHsODg8PDygUFBMTehISERARDw8PDg0NDAsLCgoICAgGBgQEAWMBAQECAwQFBgcJCQoLDA0ODg+
6AAYAAAAA/MD9AA/AGsAqWDrAO8BMwAAARUfDTsBPw09AS8ODw4lHwk7AT8IPQEvByMnByMPByU
fDz8PLw8PDiUfDz8OPQEvDSsBDw01ESERBxEfDyE/DxEvDyEPDgHhAgMFBQYHCAkKCgsLDA0NDA0
MCwsKCgkIBWYFBQMCAGMFBQYHCAkKCgsLDA0MDQ0MCwsKCgkIBWYFBQMC/scBAQEFBwgKCwYGBWY
GBgWKCACfAQEBAQUHCAoMBgYGBWYGCwoIBWUBAQHzAQECBAQEBAgYGCACICQkJCgoJCQgJBWgGBgY
EBAMDAQEBAQMDBAQGBgYIBWkICQkKCgkJCQgHCAYGBgQEBAIB/qgBAQMEBAYGBWgICQoKCgsLCws
LCgkJCQcHBWUFAwMCAgMDBQUHBWcJCQkKCwsLCwsKCgoJCAgHBgYEBAMBAlD81F4BAQMDBAUGBgC
HCAkJCQkKAYYKQKJCQgHBWYGBQQAeAgEBAQEBAQFBgYHBwgJCQkJCvzaCgkJCQkIBWcGBgUEAwM
BAWQNDAwMCwoKCQgHBgUFAwICAwUFBgcICQoKCwwMDA0NDALCwsJCQgHBWUEAwIBAQIDBAUHBwg
JCQsLCwwMMQYGBgsKCQcEAgEBAgQHCQoLBgYGBWYGCwoJBgUCAQEBCQYJCgsGBvMJCgkICAgHBWY
FBQQDAWEBAQEDAwQFBQYHBwgICAKKCQoJCQkICACGBWUFBAMCAQEBAQIDBAUFBwYHCAgJCQkGCws
KCgoJCAgHBgYEBAMBAQEBAwQEBgYHCAgJCgoKCwsLCwsKCQkJBwCHBQUdAwICAwMFBQcHBWkJCQo
LC9/81AMsA/zaCgkJCQkIBWcGBgUEBAIBAQEBAgQEBQUHBWcICQkJCQoDjGoJCQkJCAChBWUFBAQ
CAQEBAQIEBAUFBwCHCAkJCQkAAAAAaGAAAAADTQP0AAMACgAANYe1IQEjCQEjESFKA2z81AEG7AG
cAZzs/qAMfQIK/mQBnAfHAAYAAAAA/QD8wADAACwAPABIAfGAANYe1ISUhNSE1ITUhNSE1KQE
RNwMhNSEMA+j8GAG2AjL9zgIy/c4CMv30/kr6+gPo/BgMXn1efV19Xv4M+gGWXgAFAAAAAAPzA/M
AJQBpAKgArADwAAABFT8bIw8GBR8PNSMvDT8CJw8OHw8TByMPDBc/AzsBHWUzHwYzLxcjJREhEQc
RHw8hPw8RLw8hDw4CKg8QDw8ODg4NDQwMDAwKCwoKCQgJDw0KCQQDAgLxBAUGBgJC/7WDQ0NDg4
PDw8PEA8QEBAQEAKCQkJCQgIBgQEBAUDAQEDBKsKCQgIBWYGBQUdAwMBAQEBAQEDAwQEBQYHBwg
ICgoL9g4OHA4ODQ4NDg4NDQwNDKkLDA8HCAkJCAgICA4DEAUFBQMEAU4EBWkKDQ8REwoKCwsMDAw
NDQ4ODg4PDy8KAZP81F4BAQMDBAUGBgCHCAkJCQkKAYYKQKJCQgHBWYGBQQAeAgEBAQEBAQFBgY
HBwgJCQkJCvzaCgkJCQkIBWcGBgUEAwMBAZz0AgMDBAQFBQYHBwgICQkJCgsLCwsYGHsbDw4PDwo
KCQgIBWUDAsLCgkIBWcGBQQAeAgIBAfEBAwMEBQYIBQcGBw8PEA8ODa0NDQ0ODg4ODw8PDw8PEA8
PEA8PEA8ODw8ODg0ODQ0MAj8BBAMDBAQFBgCHBwkJCgoGBQQAQICBAMJEACHCAgJCh0dHRSaGRc
XCgoKCQgICACGBgYEBQMDBj781AMsA/zaCgkJCQkIBWcGBgUEAwMBAQEBAgQEBQYGBWcICQkJCQo
DJgoJCQkJCAChBgYFBAQCAQEBAQIEBAUFBwCHCAkJCQkAAgAAAAAD8wPrAB8AMwAAEW8HHw/BBU
hNSEBNwKBpWcvcDYKCAcGBQMCAQECAwUGBwgKrwgJCgoKCgkINQKQ/Xv+20EBPAGOCgkHBgUDAgE
BAgMFBgcJCtMBQsMDQ0NDg4ODg0ODQ0NDAUvBgUDAQEBCAY0I14BJEH+xAGRCwwMDQ0ODg4ODg4
NDQwMC9QABgAAAAAD9AP0AAMADwATAB0AIQAnAA1ITUhIzMVixUzFSMVMzUjNyE1ISMzBxUzNSM
3NSM3ITUhJzVMVzUjAQYC7v0S+n0/P328vPoC7v0S+nh4vHh4vPoC7v0S+j4/fWpeID4gPvrBXXw
/P3w/u14gvPoAAAAABQAAAAAD9APbAAIABQANABcAGgAAJTcjAyM3ATM3MxczAyMFIQEVITUhATU
hJzMnAgJx4x6+Xv76Wi39LF3fTWfLAST+3AGk/tIBJP5mw+10JXMBGP/+N3h4AjRQ/lo+TQGpPk5
zAAABAAAAAAsA/QACwAAATMDixUHNsMTmZUHXGd88gCPJ3zyP3EAX79xNBWajzWAAAGAAAAAP
0A9QAABDAEAcAhwCLAMsAACUhNSEHFR8OPw49AS8ODw4TITUhBxUfDTsBPw09AS8ODw4TITUhBxU
fDj8OPQEvDg8OAQYC7v0S+gIBAwMEBQUFBgcGCACICAgICACHBgYGBQQAeAwMCAQECAwMEBAUGBgY
HBwgICAgIBWgGBWYFBQUEAwMBAvoC7v0S+gIBAwMEBQUFBgcGCACICAgICACHBgYGBQQAeAwMCAQE
CAwMEBAUGBgYHBwgICAgIBWgGBWYFBQUEAwMBAvoC7v0S+gIBAwMEBQUFBgcGCACICAgICACHBgY
GBQQAeAwMCAQECAwMEBAUGBgYHBwgICAgIBWgGBWYFBQUEAwMBAkpeLwgIBWCHWYFBQUdBAICAQE
BAQICBAMFAGUGBwCHBwgICAgIBWcGBgYFBAQDAwIBAQEBAgMBAQFBgYGBWcICAFgXS4ICAgHBWY
GBgUEBAMDAgEBAQFBAQFBgYGBWcICAgICACHBgYFBAQDAwIBAQEBAgMBAQFBgYGBWcICAFgXS4ICAgHBWY
dLggICACHBgYGBQQAeAwMCAQEBAQIDAwQEBQYGBgCHCAgICAgHBWCHBgUFBQMEAgIBAQEBAgIEAwU
FBQYHBWCHCAAAAwAAAAADmQP0AAcAKACNAABFSE1MxEhESUHFQ8GLwC/Bx8GJysBDw0VERUfDTM
hMz8NNRE1Lw0rAS8OKwEPDQEdAcZb/YQBBAEDBAYHBWkJCQkHBWYEAWEBAWQGBWcJCQkJBwCGBAO
svwkJCQgICACGBgYEBAMCAgICAwQEBgYGBWgICAKJCQJ8CQkJCAgIBWYGBgQEAWICAgIDBAQGBgY
HCAgICQkJvWMBQYGBWgICQkJCgoKCwsLCwoKCgkJCQgIBWYGBQUdPoiI/SkClY4FBQgIBWUEAwE
BAWQFBWgICgkICACFBQIBAQIFBQcICCCQAgMEBAYGBgCICAgJCQn9KQkJCQgICACGBgUFBAMCAgI
CAWQFBQYGBWgICAKJCQLXCQkJCAgIBWYGBgQEAWICCgkJCAgIBWYGBQQAeAwICAgIDBAQFBgYHCAg

ICQkAAQAAAAAD9ALoAGAAAAExLw8PHxc/Gh8PByERA0QREhMTFBQUFRYWFhcXFxgYFRUVFBUEExQ
TEhMSEREREBAPDg8NDQ0MCwsKCQkIBwd1BQYGBgcQEhMWFwwMDQ0NDg4ODw8PDxAQEBAQEhESERE
QERAQDw8PDg4NDbIBtQIUdW4ODQwLCgkJBwYFBAMCAQEBAwMEBAYGCAgICgoLCwwNDQ4PDw8REBE
SEhITEExQUKA8PDw4OGxoYFhUKCQkICAcHBgUFBAQDAgIBAQEBAgMEBAYGBggICAoJCwu6AdAAAAA
OAAAAAP0A/MAAgAFAAgACwAQABQAFwAbAB4AIQAPAC0AMQB1AAABETclFzUXNyMFNyETFQUhEQE
hJRMlMycFMSEnBzcnBxcRBRMDBSUDEy0BEQMLIwUDEQCRRHw8hPw8RLw8hDw4CGcj+ZaG3MJb+wM7
+4jQBCv6EAy7+ggEKdP1S3JkBCWejWemWlvrIATJ0dP7n/up3dwEWAZhy/vQ0/vZyXgEBawQEBgY
ICAgJCgoLCwsDCgsLCwoKCQgICAYGBAQDAQEBAQMEBAYGCAgICQoKCwsL/PYLCwsKCgkICAgGBgQ
EAWEBxv7fWSQ730Bky8v+9QNxAyH+f28BHx2ZmcukmTgJyWEeP/7n/ud3dweZAR13Bv5xAR1ycv7
yAYAR/PYLCwsKCgkICAcHBQUEAwEBAQEDBAUFBwcICAKKCgsLCwMKCwsLCgoJCAGHBwUFBAMBAQE
BAwQEBgYIBwkJCgoLCwAAAAFAAAAAAP0A/MAAwAHAASADwATAAA3ITUhJSE1ISUhNSE1ITUhJSE
1IQwD6PwYAVGcKp1w/qgD6PwYAVGcKp1w/qgD6PwYDF6AW5xefV19XgAAAAAKAAAAAP0A/MAAwA
HAASADwATABcAGwAFACMARwAAARUjNSMVIzUjFSM1ARUjNSMVIzUjFSM1JRUjNSMVIzUjFSM1JxE
fByE/BxEvByEPBGOw+j7bP9oDLPo+2z/aAyz6Pts/214BAwUGAwgJCgOJCgkJBwYDBAIBAwUGAwg
JCvx3CgkJBwYFAwElvb27u7u7ARrb29vb29v6vLy8vLy8hvyCCwoJBwQGBAIBAwUHBwUJCgOECwo
JBwQGBAIBAwUGCAKAAAAAAUAAAAA/QD8wADAACACwAPABMAADchNSE1ITUhNSE1ITUhNSE1ITU
hDAPo/BgCkP1wA+j8GAKQ/XAD6PwYDF6BV59efVqAXgAAAAADAAAAAP0A00AAwAHAASAADchNSE
1ITUhNSE1IQwD6PwYA+j8GAPo/Bizb6Zwpm8AAAAABQAAAAAD9AP0AD8AXwCfAKQBIGAAJQ8PLw8
/Dx8OExUPBSsBLwU9AT8FOWefBQMPDy8PPw8fEAE1IwUVHw8zPwMXBy8FDw8fDz8PNS8DNwEzNQE
/BS8PDw4BOAEBawMEBQYGBwgICQkKCgoKCgoJCQgIBwYGBQQDAwEBAQEDAwQFBgYHCAGJCQoKCgo
KCgkJCAGHBgYFBAMDAeICAgMDBQUFBQUFAwMCAgICAwMFBQUFBQUdAwIC4QEBAwMEBQYGBwgICQk
KCgoKCgoJCQgIBwYGBQQDAwEBAQEDAwQFBgYHCAGJCQoKCgoKCgkJCAGHBgYFBAMDAftkAV6W/K4
BAwUHCAoMDQ4PERETExQUCwsVFBn2dgkKCgoVfHQUEXMRQ8ODQwKCACFAwEBawUHCAoMDQ4PERE
TExQUFBQTEExERDw4NDAoIBwUDAQEEBgd2AV6W/ZYFBAMCAwEBawUHCAoMDQ4PERETExQUFBQTEExE
RDw4NDAoIBwUD1AoKCGkJCAGHBgYFBAMDAQEBAQMDBAUGBgICAKJCgoKCgoKCQkICAcGBgUEAwM
BAQEBAwMEBQYGBwgICQkKCgEiBQUFAwMCAgICAwMFBQUFBQUdAwICAgIDAwUFAScKCGoJCQgIBwY
GBQQDAwEBAQEDAwQFBgYHCAGJCQoKCgoKCgkJCAGHBgYFBAMDAQEBAQMDBAUGBgICAKJCgqgZAF
eMpYKChQTEExERDw4NDAoIBwUDAQEEBgd2dgUEAwIDAQEDBQcICGwNDg8RERMTFBQUFBMTEREPDg0
MCgghBQMBAMQMBwBgKDA0ODxERExMUFAsLFRQTdv6iMgJqCQoKChUWFBQTEExERDw4NDAoIBwUDAQE
DLQcICGwNDg8RERMTFAADAAAAAANXA7UAIGBFAJMAAAEzHw4PDIsBNRMzHw4PDIsBNQMhPxEvDz8
PBxghAkGKCgkJCAGHBwYGBAQEAgEBAQEDAwQFBgYHBwgJCAkKCEdACgoJCQgIBwCGBgQEBAIBAQE
BAGQEBAYGBwcICAKJCgrAwAHDDQwMDBcFWRMSEQ8NDAoHBgQBAQIDBAYHBwkKCgsNDA4ODwsLCgo
KCAgIBgYFBQMDAQEBAQECAwQEBAUGDA8QEhQVFgwmDA0NDQ0N/nABogICAwQEBgYGBwgICQkKCQo
KCQgJBwGGBgUFBAMCArsBdwICAwQEBgYGCACICQkKCQoKCQkIBwGGBgYEBAMCArv9MQEBAQIGCAo
MDg8REhQUFhcYGBERERAQEA4ODgwmDAoJCQcICQkKCgoLDAsMDAwMDQwNDQwNDQwMCwwLCxQUERA
ODQoFAwQDAgEBAQAABQAAAAAD9APzAAMABwALAA8AEwAANYe1ITUhNSE1ITUhNSE1ITUhNSEMA+j
8GAPo/BgD6PwYA+j8GAPo/BgMXn1enF59XX1eAAAAAEAAAAA9QD1ADUAAATHx8/DxcRIRcPDy8
fPx8fDz8MvHw8eKwECAwQFBggICQoMDA0ODhAQERISEXQUFRUWFhcXGBgYGBgXFxcWFhUVFBQTEhI
REIr+ZrsMDA0ODg4PEBAQEBESERISEhIREhEQEQ8QDw8ODg0NDALCgoJCQgHBgYEBAQCAQEBAQI
EBAQGBgICQkKCgsMDA0NDg4PDxAPERAREhESEhwcGxoAGBgWFRQSEQ8OCwp7BQYHCAGJCQoLCww
NDQ4ODg8QEBERERISEhMTFBMUFRQYGBgXFxYWFUUFMBSEhEQEA4ODQwMCgkICAYFBAMCAgAYGBc
XFxYWFUUFMBSEhEQEA4ODQ0LCgoICAYFBAMCAQECAwQFBggICgoLDQ0ODhCKAZq7DAsLCgkJCAc
HBQUEAwMBAQEBAgQEBAYGBwgICgkLCwwMDQ0ODg8PDxAREBESERISEhIREhEQERAPDw8ODg0NDAw
LCwkKCAgHBgYEBAQCAQECAwUICQsNDxASEXUWFxgaExITERIREBAQDw8ODg0NDAsLCgoJCAcHBgY
EBAMCAQEBAgMEBQYICAoKCw0NDg4QEBESEhMUFBUVfHYXfxcYAAAAgAAAAAD8gP0AGcA7gAAARU
PGC8YPQE/FzSBHxcFHx8/DxcVATcBIyc/Dj0BLx0rAQ8dAoABAgIDAwQFBQUndXATEXyLCwwMDAw
NDQ0NDQ0NDQwNDAsMCxUUEhAPDQUFBQQDAwMBAQEBAwMDBAUFBQ0PEBIUFQsMCwwNDA0NDQ0NDQ0
NDAwMDAsLFhMTEA8NBQUFBAMDAgIB/Y0BAQMDBAUGBgICQkLCwsNDA4ODg8QEBARERISEhMTExE
REBEQEBAQDw8ODg4ODA0OAR1W/uMuDgoKCQkIBwYGBgQEAWMCAQICAwQFBgCHCAKCGsMDA0NDg8
PDxAREREREhMSEhMTExMSEhIRERAQEA8ODg4MDQsLCwkJCAgGBgUEAwMBAoIODQ0MDQwMDAsLFRQ
SEQ4NBgUEBAQDAgEBAQEBAQIDBAQEBAQYNDhESFBULCwwMDA0MDQ0ODQ0NDQwMDAwLCxUUEhEODQY
FBQQDAwICAQECAgMDBAUFBg0OERIUFQsLDAwMDA0NDQ0UEhMSEhIRERAQEA8ODg4NDAsLCwkJCAg
GBgUEBAIBAQEBAgIEBAUFBgCHCAgJCgoSLf7jVgefDg0NDQ4ODg8PDxQEBERERITExISEhIRERA
QEA8ODg4NDALCgoICQcHBQUEBAICAgIEBAUFBwcJCAoKCwwMDQ4ODg8QEBARERISEhITAAAAgA
AAAADtQP0AAMACgAANYe1IRMzESERMwFKA2z81A/zAWjz/1kMfQHN/p0BYwGeAAAAAAUAAAAA/Q
D9AA/AH8AvwD/Aa8AAEPDisBLw4/Dx8OBQ8OKwEvDj8PHw4lFQ8OLw49AT8OHw4FFQ8OLw49AT8
OHw4BHx8zPw09AS8MPQE/DjsBPx01Lx8PHgOFAQECAgQEBAQUGBgCHCAgJCAkJCAcIBgcGBQUEAwM

1046

```

}
.e-toolbar .e-icons {
font-size: 20px;
}
.e-tbar-menu-icon:before {
content: "\e718";
}
.e-tbar-search-icon:before {
content: "\e71d";
}
.e-tbar-settings-icon:before {
content: "\e702";
}
.maincontent {
height: 400px;
background-color: rgb(244, 246, 249);
}
.e-toolbar .e-toolbar-items, .footer, .e-toolbar .e-toolbar-item .e-tbar-
btn.e-btn {
background-color: midnightblue;
color: white;
font-size: 16px;
}
.main > div {
padding-left: 0px !important;
padding-right: 0px !important;
}
</style>

```



<!-- markdownlint-disable MD009 -->

Responsive Sidebar in Blazor Sidebar Component

Sidebar often behaves differently on a mobile versus a desktop display. It has an effective feature that offers to set it in opened or closed state corresponding to the specified resolution. This is achieved

through `MediaQuery` property that allows to set the Sidebar in an expanded state or collapsed state only in user-defined resolution.

In the following sample, `mediaQuery` has been used for specific resolution to close and open sidebar.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<div id="header" style="height:45px;color:white;background-color:midnightblue;font-size:1.2rem;line-height:45px;">
  <span style="position:absolute; left:10px; font-size:25px;">#9776;</span>
  <span style="margin-left:45%;">Header</span>
</div>
<SfSidebar Width="250px" MediaQuery="(min-width: 600px)">
  <ChildContent>
    <div style="text-align: center;" class="text-content"> Sidebar </div>
  </ChildContent>
</SfSidebar>
<div class="text-content" style="text-align: center;">Main content</div>
<style>
.e-sidebar {
background-color: #f8f8f8;
color: black;
}
.text-content {
font-size: 1.5rem;
padding: 3rem;
}
.main > div {
padding: 0px !important;
}
</style>
```



<!-- markdownlint-disable MD009 -->

Dock in Blazor Sidebar Component

Dock state of the Sidebar reserves some space on the page that always remains in a visible state when the Sidebar is collapsed. It is used to show the short term of a content like icons alone instead of lengthy text.

In the following sample, the list item has icon with text representation. On dock state only the icon listed out to interact. It can be achieved by using `EnableDock` property

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<div id="header" style="height:55px;text-align:
center;color:white;background-color: midnightblue;font-size:1.5rem;line-
height:55px;">
Header
</div>
<SfSidebar @ref="sidebarObj" Width="220px" DockSize="72px" EnableDock=true
@bind-IsOpen="SidebarToggle">
<ChildContent>
<div class="dock">
<ul>
<li class="sidebar-item" id="toggle" @onclick="Toggle">
<span class="e-icons expand"></span>
<span class="e-text" title="menu">Menu</span>
</li>
<li class="sidebar-item">
<span class="e-icons home"></span>
<span class="e-text" title="home">Home</span>
</li>
<li class="sidebar-item">
<span class="e-icons profile"></span>
<span class="e-text" title="profile">Profile</span>
</li>
<li class="sidebar-item">
<span class="e-icons info"></span>
<span class="e-text" title="info">Info</span>
</li>
<li class="sidebar-item">
<span class="e-icons settings"></span>
<span class="e-text" title="settings">Settings</span>
</li>
</ul>
</div>
</ChildContent>
</SfSidebar>
<div id="main-content container-fluid col-md-12" style="margin-top: 10%;">
<div class="title default" style="text-align: center">Main content</div>
</div>
@code{
SfSidebar sidebarObj;
public bool SidebarToggle = false;
public void Toggle()
{
SidebarToggle = !SidebarToggle;
}
}
```

```
<style>
/* Content area styles */
.title {
font-size: 20px;
}
/* Sidebar styles */
.e-sidebar .e-icons::before {
font-size: 25px;
}
/* dockbar icon Style */
.e-sidebar .home::before {
content: '\e102';
}
.e-sidebar .profile::before {
content: '\e10c';
}
.e-sidebar .info::before {
content: '\e11b';
}
.e-sidebar .settings::before {
content: '\e10b';
}
.e-sidebar .expand::before,
.e-sidebar.e-right.e-open .expand::before {
content: '\e10f';
}
.e-sidebar.e-open .expand::before,
.e-sidebar.e-right .expand::before {
content: '\e10e';
}
/* end of dockbar icon Style */
.e-sidebar.e-dock.e-close span.e-text {
display: none;
}
.e-sidebar.e-dock.e-open span.e-text {
display: inline-block;
}
.e-sidebar li {
list-style-type: none;
cursor: pointer;
}
.e-sidebar ul {
padding: 0px;
}
.e-sidebar span.e-icons {
color: #c0c2c5;
line-height: 2
}
.e-open .e-icons {
margin-right: 16px;
}
.e-open .e-text {
overflow: hidden;
font-size: 15px;
}
.sidebar-item {
text-align: center;
```

```

border-bottom: 1px #e5e5e5 solid;
}
.e-sidebar.e-open .sidebar-item {
text-align: left;
padding-left: 15px;
color: #c0c2c5;
}
.e-sidebar {
background: #2d323e;
overflow: hidden;
}
@@font-face {
font-family: 'e-icons';
src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgTlMvMjciQ6oAAAEoAAAVmNtYXBH1Ec8AAABsAAAAHJnbHlmKcX
fOQAAAKAAAAg4aGVhZBLt+DYAADQAAAAANmhoZWEHogNsAAAArAAAACRobXR4LvgAAAAAYAAAAA
wbG9jYQYkCgIAAAIkAAAAAGmlheHABGQEOAAABCAAAACBuYW1lR4040wAACngAAAJtcG9zdEFgIbw
AAAZoAAAArAABAAADUv9qAFoEAAAA//UD8wABAAAAAAAAAAAAAAAAADAABAAAAQAAlbrm7l8
PPPUACwPoAAAAANfuWa8AAAAA1+5ZrwAAAAAD8wPzAAAACAACAAAAAAAAAAAAEAAAAAQIAAwAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPqAZAABQAAANoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA4QLhkANS/2oAWgPzAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAAAAAgAAAAAUAAMAAQA
AABQABABeAAAAADgAIAAIABuEC4QnhD+ES4RvhkP//AADhAuEJ4QvhEuEa4ZD//wAAAAAAAAAAAA
AAAABAA4ADgAOABYAFgAYAAAAAQACAAYABAADAAGABwAKAAkABQALAAAAAAAAAB4AQABaAQYB5gJ
kAnoCjgKwA8oEHAaaaaIAaaaa+oDlQAEAAoAAAEFESERCQEVCQE1AgcBZv0mAXQB5P4c/g4Cw/D
+lwFpAcP+s24BTf6qbgAAAAEAAAAAA+oD6gALAAATCQEXCQEHQCEnCQF4AYgBiGP+eAGIY/54/nh
jAYj+eAPr/ngBiGP+eP54YwGI/nhjAYgBiAAAAwAAAAAD6gOkAAMABwALAAA3IRUHESEVIREhFSE
VA9b8KgPW/CoDlVwq6I0B64wB640AAAAEAAAAAA+oD4QCaAAABMx8aHQEPDjEPah8bIT8bNS8SPxs
CAA0aGhgMDAsLCwoKCgkJCQgHBWYGBgUEBAMCAgECAwUFBggICQoLCwMDg0GAgEBAgIDBAMIBiI
dHh0cHB0ZFhUSEAcFBgQDAwEB/CoBAQMDBAUGBw8SFRYYGhsbHB0cHwsJBQQAeWIBAQMEDg0NDAs
LCQkJBwYGBAMCAQEBAgIDBAQFBQYGBWgICAKJCgoKCwsLDawMGRoD4gMEBwQFBQYGBWgICAKKCgs
LDawNDQ4ODxAQEBEWFxYWFhYVFRQUExIRERAOFxMLCggIBgYFBgQMDAwNDg4QDxERERIJCQkKCQk
JFRQJCQoJCQgJEHERERAPDw4NDQsMBWgFBgYICQkKDAwODw8RERMTEUUFhUWFxYWFxEQEBApDg4
NDQwMCwsKCgkICAgHBgYFBQQAEBQAAAAAAwAAAAAD8wPzAEEAZQDFAAABMx8FFREzHwYdAg8GIS8
GPQI/BjM1KwEvBT0CPwUzNzMfBR0CDwUrAi8FPQI/BTMnDw8fFz8XLxcPBgI+BQQDAwMCAT8EBAM
DAWIBAQIDAWEbP7cBAQDAwMCAQECaWMDBAQ/PwQEAWMDAgEBAgMDAwQE0AUEAWMDAgEBAgMDAwQ
FfaUEAWMDAgEBAgMDAwQFvRsbGRcWFRMREA4LCQgFAwEBAwUHCgsOEBETFRYXGRochR4eHyAgISI
iISAgHx4eHRSbGRcWFRMREA4LCQgFAwEBAwUHCgsOEBETFRYXGRsbHR4eHyAgISIiISAgHx4eAqY
BAgIDBAQE/rMBAQEDAwQEBGgEBAQDAgIBAQEBAgIDBAQEaAQEBAMDAQEBAECaWMDBAV0BAQDAwM
CAeUBAgIEAwQEaAUEAWMDAgEBAgMDAwQFaAQEAwQCAgElERMVfhcZGhwdHh4fICAhIiIhICAfHh4
dGxsZFxYVExEQDgsJCAUDAQEDBQcKCw4QERMVfhcZGxsDhH4fICAhIiIhICAfHh4dHBoZFxYVExE
QDgsKBwUDAQEDBQcKCw4AAAAIAAAAAA9MD6QALAE8AAAE0AQcuASc+ATceAQEHBgcNjgYPAQYWHWE
GFBChDgEfAR4BPWEWHwEeATsBMjY/ATY3Fxy2Pwe2Ji8BNjQnNz4BLwEuAQ8Bji8BLgErASIGaps
BY0tKYwICY0pLY/7WEy4nfAkRBWQEAWdqAwNqBwMEZAURCXwnLhMBDgnICg4BEy4mfQkRBGQFAwh
pAwNpCAMFZAQSCH0mLhMBDgrICQ4B9UpjAgJjSkpjAgJjAZWEFB4yBAYIrggSBlIYMhSBhIIrgg
FAZiFe4QJDAwJhBQeMgQGCK4IEgZSGDIYUgYSCK4IBQMyHxOECQwMAAEAAAAAAwED6gAFAAAJAic
JAQEbAef+FhoBzf4zA+v+Ff4VHwHMac0AAAAAAQAAAAADAQPqAAUAAAEXCQEHAQLlHf4zAc0a/hY
D6x7+M/40HwHrAAEAAAAAA/MD8wALAAATCQEXCQE3CQEnCQENAY7+cmQBjwGPZP5yAY5k/nH+cQO
P/nH+cWQBjv5yZAGPAY9k/nEBjwAAwAAAAAD8wPzAEEAgQEBAAlDw4rAS8dPQE/DgUVDw4BPw4
7AR8dBRUfHTsBPx09AS8dKwEPHQL1DQ00Dg4PDw8QEBAQERERERUUFbQTExITEREREBAPDw00DAw
LCwkJCACGBgQEAgIBAgIEAwUFBgYHBwkICQoCygECAGQDBQUGBgCHCQgJCv3QDQ00Dg4PDw8QEBA
QERERERUUFbQTExITEREREBAPDw00DAwLCwkJCACGBgQEAgL8fgIDBQUHCAkKCwNDg8PERESExQ
UFRYWFhgXGBkZGRoaGRkZGBcyFhYVFRQUExIREQ8PDg0MCwoJCAcFBQMCAGMFBQcICQoLDA00Dw8
RERITFBQVfHYWGBcyYGRkZGhoZGRkYFhgWfHYVFBQTEhERDw8ODQwLCgkIBwUFAwLFCgkICQcHBgY
FBQMEAgIBAgIEBAYGBWgJCQsLDAwODQ8PEBARERETehMTFBQUFREREREQEBAQDw8PDg4ODQ31ERE
RERAQEBApDw8ODg4NDQIwCgkICQcHBgYFBQMEAgIBAgIEBAYGBWgJCQsLDAwODQ8PEBARERETehM
TFBQUFRoZGRkYFhgWfHYVFBQTEhERDw8ODQwLCgkIBwUFAwICAwUFBWgJCgsMDQ4PDxEREhMUFBu

```

```

WFhYYFxfGZGRkaGhkZGRGxGBYWFhUUFBMSEREPdW4NDAsKCQgHBQUDAgIDBQUHCAkKCwWNDg8PERE
SExQUFRYWFhgXGBkZGQAAAQAAAAAD6gPqAEMAABMhHw8RDw8hLw8RPw6aAswNDgwMDAsKCggIBwU
FAwIBAQIDBQUHCAgKCgsMDAwODf00DQ4MDAwLCgoICAcFBQMCAQECaWUFBwgICgoLDAwMDgPrAQI
DBQUHCAgKCgsLDA0NDv00Dg0NDAsLCgoICAcFBQMCAQECaWUFBwgICgoLCwWNDQ4CzA4NDQwLCwo
KCAGHBQUDAgAAABIA3gABAAAAAAAAAAAAAABAAAAAAAAABAA0AAQABAAAAAAAAACAACADgABAAAAAA
DAA0AFQABAAAAAAAEAA0AIgABAAAAAAFAAsALwABAAAAAAAGAA0AOgABAAAAAAAKACwArWABAAA
AAAAALABIAcWADAAEEECQAAAAIAhQADAAEEECQABABOAhWADAAEEECQACAA4AoQADAAEEECQADABOArWA
DAAEEECQAEABOAYQADAAEEECQAFABYA4wADAAEEECQAGABOa+QADAAEEECQAKAFgBEWADAAEEECQALACQ
BayB1LW1jb25zLW1ldHJvUmVndWxhcmUtaWNvbnMtbWV0cm91LW1jb25zLW1ldHJvVmVyc2lubiA
xLjB1LW1jb25zLW1ldHJvRm9udCBnZW51cmF0ZWQgdXNpbmcgU3luY2Z1c2lubiBNZXRYbyBTdHV
kaW93d3cuc3luY2Z1c2lubi5jb20AIABlAC0AaQBjAG8AbgBzAC0AbQBlAHQAcgBvAFIAZQBnAHU
AbABhAHIAZQAtAGkAYwBvAG4AcwAtAG0AZQB0AHIAbwBlAC0AaQBjAG8AbgBzAC0AbQBlAHQAcgB
vAFYAZQBvAHMAaQBvAG4AIAAxA4AMABlAC0AaQBjAG8AbgBzAC0AbQBlAHQAcgBvAEYAbwBuAHQ
AIAbnAGUAbgBlAHIAIAYQB0AGUAZAAGAHUAcwBpAG4AZwAgAFMAeQBvAGMAZgBlAHMAaQBvAG4AIA
NAGUAdABYAG8AIABTahQAdQBkAGkAbwB3AHcAdwAuAHMAeQBvAGMAZgBlAHMAaQBvAG4ALgBjAG8
AbQAAAAACAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAAa
BCwEMAQ0AB2hvbWUtMDELQ2xvc2UtaWNvbnMHbWVudS0wMQR1c2VyB0JUX2luZm8PU2V0dGluZ19
BbmRyb2lkDWN0ZXZyb24tcmlnaHQMY2hldnJvb1lsZWZ0CE1UX0NsZWZyDE1UX0p1bmmttYWlscwR
zdG9wAAA=) format('trueType');
font-weight: normal;
font-style: normal;
}
.main > div {
padding: 0px !important;
}
/* end of sidebar styles */
</style>

```



Styles and Appearance in Blazor Sidebar Component

The following content provides the exact CSS structure that can be used to modify the component's appearance based on the user's preference.

Customizing the sidebar

Use the below CSS to customize the sidebar root element.

CSS

```
.e-sidebar {  
  background: #898b2b  
}
```

Customizing the sidebar based on the positions

Use the below CSS to customize the left positioned sidebar.

CSS

```
.e-sidebar.e-left {  
  border-right: 2px solid red;  
}
```

Use the below CSS to customize the right positioned sidebar.

CSS

```
.e-sidebar.e-right {  
  border-left: 2px solid red;  
}
```

Customizing the sidebar based on the active state

Use the below CSS to customize the open state of the left positioned sidebar.

CSS

```
.e-sidebar.e-left.e-open {  
  transition: transform 2.5s ease;  
}
```

Use the below CSS to customize the open state of the right positioned sidebar.

CSS

```
.e-sidebar.e-right.e-open {  
  transition: transform 2.5s ease;  
}
```

Use the below CSS to customize the closed state of the left positioned sidebar.

CSS

```
.e-sidebar.e-left.e-transition.e-close {  
  transition: transform 2.5s ease, visibility 1200ms;  
}
```

Use the below CSS to customize the closed state of the right positioned sidebar.

CSS

```
.e-sidebar.e-right.e-transition.e-close {  
  transition: transform 2.5s ease, visibility 1200ms;  
}
```

Customizing the sidebar with dock state

When the Dock support is enabled, the "e-dock" class will be added to the root element. Based on that class, all the above stated customization can also be done. Use the following CSS to customize the sidebar element with a dock state.

CSS

```
.e-sidebar.e-dock {  
  background: #2d323e;  
}
```

Customizing the different types of sidebar

Use the below CSS to customize the auto type sidebar.

CSS

```
.e-sidebar.e-left.e-auto {  
  background-color: pink;  
}
```

Use the below CSS to customize the push type sidebar.

CSS

```
.e-sidebar.e-left.e-push {  
  background-color: beige;  
}
```

Use the below CSS to customize the over type sidebar.

CSS

```
.e-sidebar.e-left.e-over {  
  background-color: aqua;  
}
```

Use the below CSS to customize the slide type sidebar.

CSS

```
.e-sidebar.e-left.e-slide {  
  background-color: green;  
}
```

Customizing the backdrop of the sidebar

Use the below CSS to customize the backdrop of the sidebar.

CSS

```
.e-sidebar-overlay {
```

```
background-color: aqua;
}
```

Customizing the sidebar in the RTL direction

When the RTL (right to left direction) support is enabled, the "e-rtl" class will be added to the root element. Based on that class, all the above stated customization can also be done. Use the following CSS to customize the sidebar element in the RTL (right to left direction) mode.

CSS

```
.e-sidebar.e-left.e-rtl {
background-color: antiquewhite;
}
```

How To

<!-- markdownlint-disable MD009 -->

Initialize the Blazor Sidebar with ListView

Any HTML element can be placed in the Sidebar content area. Sidebar supports all types of HTML structures like `TreeView`, `ListView`, etc.

In the following example, the Sidebar is rendered with `ListView` component in its content area. Add the HTML div tag with its id attribute as `default` in the `index.html` file to initialize the Sidebar.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
@using Syncfusion.Blazor.Lists
@using Syncfusion.Blazor.Buttons
<SfSidebar @ref="sidebarObj" Type="@Type" Width="100%"
HtmlAttributes="@htmlAttribute" @bind-IsOpen="SidebarToggle">
<ChildContent>
<div class="title1"> Menu </div>
<div class="closebtn">
<SfButton ID="close" @onclick="@Close" CssClass="e-btn close-btn">
<ChildContent>
<span id="innerclose" class="e-icons close-icon"></span>
</ChildContent>
</SfButton>
</div>
<div id="listcontainer">
<!-- ListView element declaration -->
<SfListView DataSource="@Data" ID="list">
<ListViewFieldSettings TValue="ListViewData" Id="Id"
Text="Text"></ListViewFieldSettings>
</SfListView>
</div>
<div class="sub-title">
ListView component is placed inside the sidebar content area.
</div>
</ChildContent>
</SfSidebar>
<!-- end of sidebar element -->
<!-- main content declaration -->
<div>
```

```

<div class="title2">Main content</div>
<div class="sub-title"> Click the button to open/close the Sidebar.</div>
<div style="padding:20px" class="center-align">
<SfButton ID="toggle" @onclick="@Toggle" IsToggle="true" CssClass="e-btn e-
info">Toggle Sidebar</SfButton>
</div>
</div>
@code{
SfSidebar sidebarObj;
public SidebarType Type = SidebarType.Over;
Dictionary<string, object> htmlAttribute = new Dictionary<string, object>()
{
{"class", "default_sidebar" },
};
List<ListViewData> Data = new List<ListViewData>()
{
new ListViewData{ Text = "Home", Id = "list-01"},
new ListViewData{ Text = "Offers", Id = "list-02"},
new ListViewData{ Text = "Support", Id = "list-03"},
new ListViewData{ Text = "Logout", Id = "list-04"}
};
class ListViewData
{
public string Id { get; set; }
public string Text { get; set; }
}
public bool SidebarToggle = false;
public void Close()
{
SidebarToggle = false;
}
public void Toggle()
{
SidebarToggle = !SidebarToggle;
}
}
<style>
/* Listview element styles */
#listcontainer {
width: 100%;
}
#list {
margin: 0 auto;
width: 30%;
}
.e-listview .e-list-item {
text-align: center;
font-size: 14px;
padding: 0;
}
/* Button element styles */
.e-btn.close-btn :hover { /* csslint allow: adjoining-classes*/
box-shadow: none;
background: transparent;
}
.close-btn, .e-listview .e-list-item, .default_sidebar {
background-color: rgb(20, 118, 210);
}

```

```

color: #ffffff;
}
.close-icon::before {
content: '\e109';
}
.close-btn {
box-shadow: none;
border: none;
}
.close-btn:hover {
color: #fafafa;
}
.e-icons.close-icon { /* csslint allow: adjoining-classes */
line-height: 2.2;
}
.closebtn {
top: 15px;
line-height: 36px;
height: 42px;
color: black;
position: absolute;
right: 10px;
}
/* Sample level styles */
.title1 {
text-align: center;
font-size: 20px;
padding: 15px;
}
.title2 {
text-align: center;
font-size: 20px;
padding: 15px;
}
.sub-title {
text-align: center;
font-size: 16px;
padding: 10px;
}
.center-align {
text-align: center;
padding: 20px;
}
body {
margin: 0;
}
@@font-face {
font-family: 'e-icons';
src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjciQ6oAAAEoAAAAVmNtYXBH1Ec8AAABsAAAAHJnbHlmKcX
fOQAAAKAAAAG4aGVhZBLt+DYAADQAAAAANmhoZWEHogNsAAAArAAAACRobXR4LvgAAAAAYAAAAA
wbG9jYQYUkCgIAAAIkAAAAGmlheHABGQEOAAABCAAAACBuYW1lR4040wAACngAAAJtcG9zdEFgIbw
AAAZoAAAArAABAAADUv9qAFoEAAAA//UD8wABAAAAAAAAAAAAAAAAADAABAAAAQAAlbrm7l8
PPPUACwPoAAAAANfuWa8AAAAA1+5ZrwAAAAAD8wPzAAAACAACAAAAAAAAAAAAEAAAAAQIAAwAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPqAZAABQAAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA4QLhkANS/2oAWgPzAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAAAAAgAAAAAUAAMAAQA

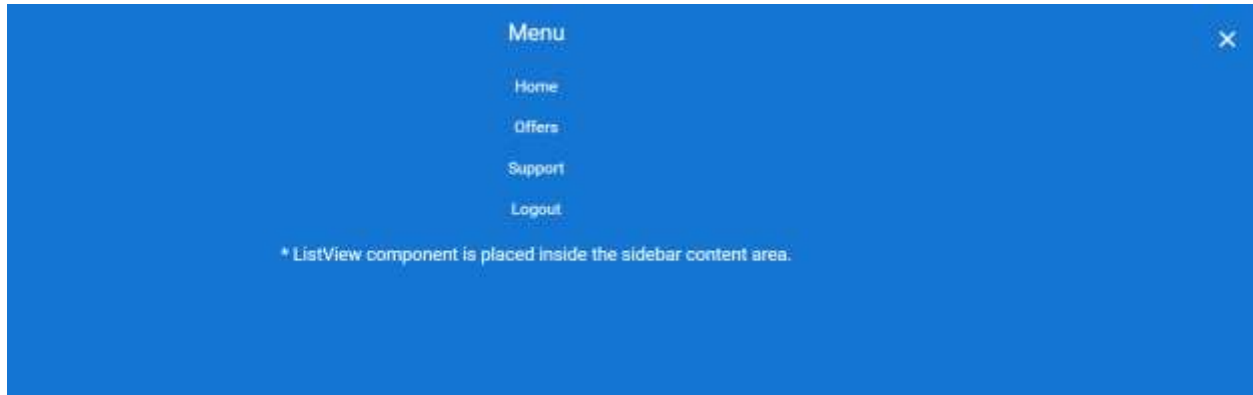
```

```

AABQABABeAAAAADgAIAAIIABuEC4QnhD+ES4RvhkP//AADhAuEJ4QvhuEuEa4ZD//wAAAAA
AAAAABAA4ADgAOABYAFgAYAAAAAQACAAYABAADAAgABwAKAAkABQALAAAAAAAAB4AQABaQYB5gJ
kAnoCjgKwA8oEHAIAAAAAIAAAAA+oDlQAEAAoAAAEFESERCQEVCQE1AgcBZv0mAXQB5P4c/g4Cw/D
+lwFpAcP+s24BTf6qbgAAAAEAAAAAAAA+oD6gALAAATCQEXCQEHCQEnCQF4AYgBiGP+eAGIY/54/nh
jAYj+eAPr/ngBiGP+eP54YwGI/nhjAYgBiAAAAwAAAAAD6gOkAAMABwALAAA3IRUHeSEVIREhFSE
VA9b8KgPW/CoDlvwq6I0B64wB640AAAEAAAAAAAA+oD4QCaAAABMx8aHQEPDjEPah8bIT8bNS8SPxs
CAA0aGhgMDAsLCwoKCgkJCQgHBwYGBgUEBAMCAgECAwUFBggICQoLCwwMDg0GAgEBAgIDBAMIBiI
dHh0cHBoZFhUSEAcFBgQDAwEB/CoBAQMDBAUGBw8SFRYYGhsbHB0cHwsJBQQEAWIBAQMEDg0NDAs
LCQkJBwYGBAMCAQEBAgIDBAQFBQYGBwgICakJCgoKCwsLDAwMGRoD4gMEBwQFBQYGBwgICakKCgs
LDAWNDQ40DxJCQEBEWfxYWFhYVFRQUEXIRERAOfxMLCggIBgYFBgQMDAwNDg4QDxERERIJCQkKCQh
JFRQJJCQoJCQgJEHERERAPDw4NDQsMBwGfYgICQkKDAwODw8RERMTExUUFhUWFxYWFxEQEBApDg4
NDQwMCwsKCgkICAgHBgYFBQQEBCQAAAAAAwAAAAAD8wPzAEEAZQDFAAABMx8FFREzHwYdAg8GIS8
GPQI/BjMlKwEvBT0CPwUzNzMfBR0CDwUrAi8FPQI/BTMnDw8ffz8XLxcPBGI+BQQDAwMCAT8EBAM
DAwIBAQIDAwMEBP7cBAQDAwMCAQECAwMDBAQ/PwQEAwMDAgEBAgMDAwQE0AUEAwMDAgEBAgMDAwQ
FfAUEAwMDAgEBAgMDAwQFvRsbGRcWFRMREA4LCQgFAwEBawUHCgsOEBETFRYXGRocHR4eHyAgISI
iISAgHx4eHRSbGRcWFRMREA4LCQgFAwEBawUHCgsOEBETFRYXGRsbHR4eHyAgISIiISAgHx4eAqY
BAgIDBAQE/rMBAQEDAwQEBGgEBAQDAgIBAQEBAgIDBAQEaAQEBAMDAQEBOAECawMDBAVoBAQDAwM
CAeUBAgIEAwQEaAUEAwMDAgEBAgMDAwQFaAQEAwQCAgElERMVFhcZGhwdHh4fICaHiIhICaFhH4
dGxsZFxYVExEQDgsJCAUDAQEDBQcKCw4QERMVFhcZGxsDhH4fICaHiIhICaFhH4dHBoZFxYVExE
QDgsKBwUDAQEDBQcKCw4AAAAIAAAAAA9MD6QALAE8AAAE0AQcuASc+ATceAQEHBgcnJgYPAQYWHwE
GFBCHDgEfAR4BPWEWHwEeAtSBMJY/ATY3Fxy2PwE2Ji8BNjQnNz4BLwEuAQ8Bji8BLGErASIGApS
BY0tKYwICYOpLY/7WEy4nfAkRBWQEAwdqAwNqBwMEZAURCXwnLhMBDgnICg4BEy4mfQkRBGQFAwh
pAwNpCAMFZAQSCH0mLhMBDgrICQ4B9UpjAgJjSkpjAgJjAZWEFB4yBAYIrggSBLiYmhhSBhIIrgg
FAzIfE4QJDAwJhBQeMgQGCK4IEgZSGDIYUgYSCK4IBQMjHxOECQwMAAEAAAAAAwED6gAFAAAJAic
JAQEbAef+FhoBzf4za+v+ff4VhWHMAC0AAAAAAQAAAAADAQPqAAUAAAEXCQEHAQLlHf4zaC0a/hY
D6x7+M/40HwHrAAEAAAAA/MD8wALAAATCQEXCQE3CQEnCQENAY7+cmQBjwGPZP5yAY5k/nH+cQO
P/nH+cWQbJv5yZAGPAY9k/nEBjwAAAAwAAAAAD8wPzAEEAgQEBAALDw4rAS8dPQE/DgUVdW4BPw4
7AR8dBRUfHTSBPx09AS8dKwEPHQL1DQ0ODg4PDw8QEBAQERERERUUFbQTExITEREREBAPDw0ODAw
LCwkJCACGBgQEAgIBAgIEAwUFBgYHBwkICQoCygECAGQDBQUGBgCHCQgJCv3QDQ0ODg4PDw8QEBA
QERERERUUFbQTExITEREREBAPDw0ODAwLCwkJCACGBgQEAgL8fgIDBQUHCakKCwwNDg8PERESExQ
UFRYWFhgXGBkZGRoaGrkZGBcYFhYWFRQUEXIREQ8PDg0MCwoJCACFBQMCAgMFBQcICQoLDA0ODw8
RERITFBQVFhYWGBcYGRkZGhoZGRkYFxfGWFhYVFBQTEhERDw8ODQwLCgkIBwUFAwLFCgkICQcHBgY
FBQMEAgIBAgIEBAYGBwgJCQsLDAwODQ8PEBARERETeHMTFBQUFREREREQEBAQDw8PDg4ODQ31ERE
RERAQEBApDw8ODg4NDQIwCgkICQcHBgYFBQMEAgIBAgIEBAYGBwgJCQsLDAwODQ8PEBARERETeHMT
FBQUFRoZGRkYFxfGWFhYVFBQTEhERDw8ODQwLCgkIBwUFAwICAwUFBwgJCgsMDQ4PDxEREHMFBU
WFhYYFxfGZGRkaGhkZGRgXGBYWFhUUFBMSEREPEdW4NDAsKCQgHBQUdAgIDBQUHCakKCwwNDg8PERE
SExQUFRYWFhgXGBkZGQAAAQAAAAAD6gPqAEMAABMhHw8RDw8hLw8RPw6aAswNDgwMDAsKCgIBwU
FAwIBAQIDBQUHCAGKCgsMDAwODf00DQ4MDAwLCgoICACFBQMCAQECAwUFBwgICgoLDAwMDgPrAQI
DBQUHCAGKCgsLDA0NDv00Dg0NDAsLCgoICACFBQMCAQECAwUFBwgICgoLCwwNDQ4Cza4NDQwLCwo
KCAgHBQUdAgAAABIA3gABAAAAAAAEAAAAABAAAAAABAA0AAQABAAAAAAACAAcAdgABAAAAAA
DAA0AFQABAAAAAAEA0AIgABAAAAAAFAAASALwABAAAAAAAGAA0AAGABAAAAAAAKCAwArwABAAA
AAAAABIAcWADAEEECQAAAAIAhQADAAEECQABABoAhhwADAEEECQAA0AA0AQAADAAEECQADABoArwA
DAAEECQAEABoAyQADAAEECQAFABYA4wADAEEECQAGABoA+QADAAEECQAKAFgBEwADAEECQALACQ
BayBlLWl1jb25zLWl1dHJvUmVndWxhcmUtaWNvbnMtbWV0cm91LWl1jb25zLWl1dHJvVmVyc2lvbiA
xLjBlLWl1jb25zLWl1dHJvRm9udCBnZW51cmF0ZWQgdXNpbmcgU3luY2Z1c2lvbiBNZXRybyBTdHV
kaW93d3cuc3luY2Z1c2lvbi5jb20AIABlAC0AaQBJAg8AbgBzAC0AbQBlAHQAcgBvAFIAZQBNAHU
AbABhAHIAZQAtAGkAYwBvAG4ACwAtAG0AZQB0AHIAbwBlAC0AaQBJAg8AbgBzAC0AbQBlAHQAcgB
vAFYAZQByAHMAaQBVAG4AIAAxAC4AMABlAC0AaQBJAg8AbgBzAC0AbQBlAHQAcgBvAEYABwBuAHQ
AIABnAGUAbgBlAHIAAYQB0AGUAZAAGAHUAcwBpAG4AZwAgAFMAeQBwAGMAZgBlAHMAaQBVAG4AIAB
NAGUAdABYAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBwAGMAZgBlAHMAaQBVAG4ALgBjAG8
AbQAAAAACAAAAAAAAoAAAAAIAAAAAAAAAAAAAAAAAAAAAAAAAAwBAGEDAQQBBQEGAQcBCAEJAQo
BCwEMAQ0AB2hvbWUtMDElQ2xvc2UtaWNvbnMhbnVudS0wMQR1c2VyB0JUX2luZm8PU2V0dGluZ19
BbmRyb2lkDWN0ZXZyb24tcmlnaHQMY2hldnJvbi1sZWZ0CE1UX0NsZWfYDE1UX0plbmttYWlscwR
zdG9wAAA=) format('truetype');
font-weight: normal;
font-style: normal;
}

```

```
.main > div {
padding: 0px !important;
}
</style>
```



```
<!-- markdownlint-disable MD009 -->
```

Open and close the Sidebar in Blazor Sidebar Component

Opening and closing the Sidebar can be achieved with `IsOpen` property.

In the following sample, `IsOpen` property has been used to show or hide the Sidebar on button click.

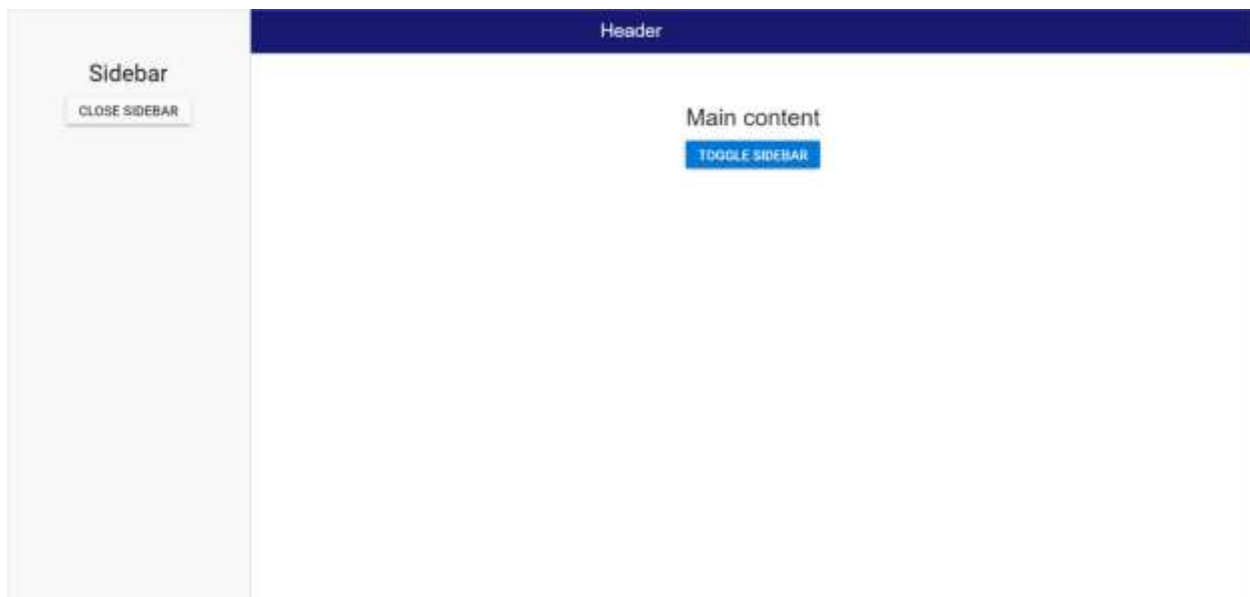
ASPX-CS

```
@using Syncfusion.Blazor.Navigations
@using Syncfusion.Blazor.Buttons
<div id="header" style="height:45px;text-align:
center;color:white;background-color:midnightblue;font-size:1.2rem;line-
height:45px;">
Header
</div>
<SfSidebar @ref="sidebarObj" Width="250px" @bind-IsOpen="SidebarToggle">
<ChildContent>
<div style="text-align: center;" class="text-content">
<span>Sidebar</span>
<span>
<SfButton @onclick="Close" CssClass="e-btn close-btn">Close
Sidebar</SfButton>
</span>
</div>
</ChildContent>
</SfSidebar>
<div class="text-content" style="text-align: center;">
<div>Main content</div>
<div>
<SfButton @onclick="Toggle" IsToggle="true" CssClass="e-btn e-info">Toggle
Sidebar</SfButton>
</div>
</div>
@code{
SfSidebar sidebarObj;
public bool SidebarToggle = false;
public void Close()
```

```

{
  SidebarToggle = false;
}
public void Toggle()
{
  SidebarToggle = !SidebarToggle;
}
}
<style>
.e-sidebar {
background-color: #f8f8f8;
color: black;
}
.text-content {
font-size: 1.5rem;
padding: 3rem;
}
.main > div {
padding: 0px !important;
}
</style>

```



<!-- markdownlint-disable MD009 -->

Multiple Sidebar in Blazor Sidebar Component

Two Sidebars can be initialized in a web page with same main content. Sidebars can be initialized on right side or left side of the main content using **Position** property.

The HTML element with class name **e-main-content** will be considered as the main content and both the Sidebars will behave as side content to this main content area of a web page.

In the following sample, more than one sidebar is rendered based on **Position** property.

ASPX-CS

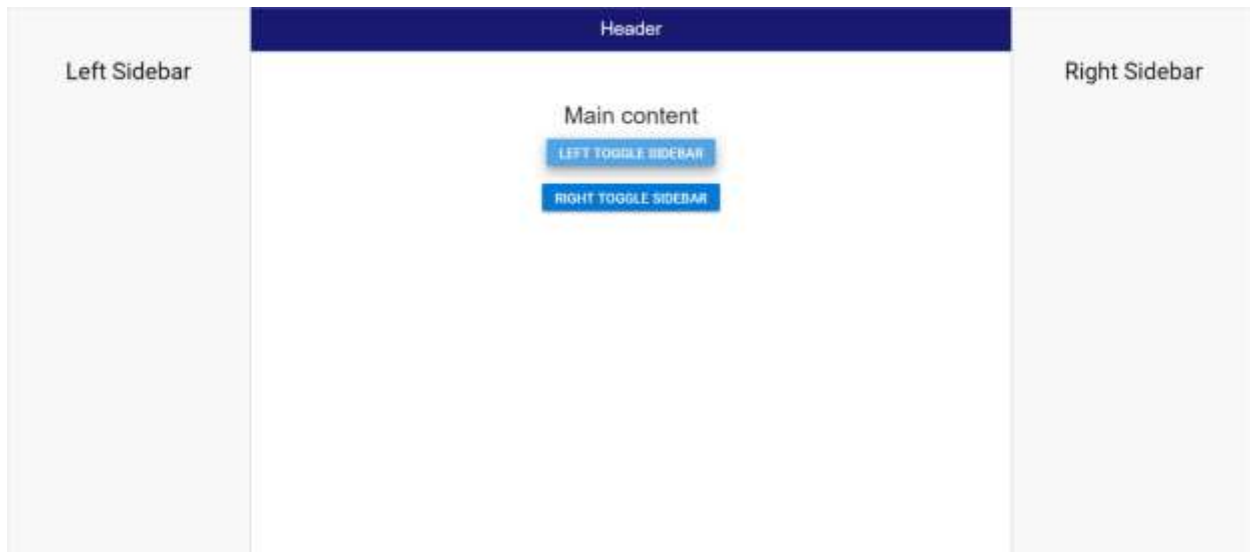
```
@using Syncfusion.Blazor.Navigations
```



```

@using Syncfusion.Blazor.Buttons
<div id="header" style="height:45px;text-align:
center;color:white;background-color:midnightblue;font-size:1.2rem;line-
height:45px;">
Header
</div>
<SfSidebar @ref="leftSidebarInstance" Type=SidebarType.Push Width="250px"
@bind-IsOpen="LeftToggle">
<ChildContent>
<div style="text-align: center;" class="text-content"> Left Sidebar</div>
</ChildContent>
</SfSidebar>
<SfSidebar @ref="rightSidebarInstance" Width="250px"
Position=SidebarPosition.Right @bind-IsOpen="RightToggle">
<ChildContent>
<div style="text-align: center;" class="text-content"> Right Sidebar</div>
</ChildContent>
</SfSidebar>
<div style="text-align:center" class="text-content e-main-content">
<div>Main content</div>
<div>
<SfButton @onclick="ToggleLeftSidebar" IsToggle="true" CssClass="e-btn e-
info">Left Toggle Sidebar</SfButton>
</div>
<div style="margin-top:10px">
<SfButton @onclick="ToggleRightSidebar" IsToggle="true" CssClass="e-btn e-
info">Right Toggle Sidebar</SfButton>
</div>
</div>
@code {
SfSidebar leftSidebarInstance;
SfSidebar rightSidebarInstance;
public bool LeftToggle = false;
public bool RightToggle = false;
public void ToggleLeftSidebar()
{
LeftToggle = !LeftToggle;
}
public void ToggleRightSidebar()
{
RightToggle = !RightToggle;
}
}
<style>
.e-sidebar {
background-color: #f8f8f8;
color: black;
}
.text-content {
font-size: 1.5rem;
padding: 3rem;
}
.main > div {
padding: 0px !important;
}
</style>

```



Signature

Getting Started with Blazor Signature Component

This section briefly explains about how to include [Blazor Signature](#) component in your Blazor Server App and Blazor WebAssembly App using Visual Studio.

Prerequisites

- [System requirements for Blazor components](#)

Create a new Blazor App in Visual Studio

You can create **Blazor Server App** or **Blazor WebAssembly App** using Visual Studio in one of the following ways,

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion Blazor Extension](#)

Install Syncfusion Blazor Inputs NuGet in the App

Syncfusion Blazor components are available in nuget.org. In order to use Syncfusion Blazor components in the application, add reference to the corresponding NuGet. Refer to [NuGet packages topic](#) for available NuGet packages list with component details.

To add Blazor Signature component in the app, open the NuGet package manager in Visual Studio (*Tools* → *NuGet Package Manager* → *Manage NuGet Packages for Solution*), search for [Syncfusion.Blazor.Inputs](#) and then install it.

Add Style Sheet

Checkout the [Blazor Themes topic](#) to learn different ways to refer themes in Blazor application, and to have the expected appearance for Syncfusion Blazor components. Here, the theme is referred using [Static Web Assets](#).

To add theme to the app, open the NuGet package manager in Visual Studio (*Tools* → *NuGet Package Manager* → *Manage NuGet Packages for Solution*), search for [Syncfusion.Blazor.Themes](#) and then install it. Then, the theme style sheet from NuGet can be referred as follows,

Blazor Server App

- For .NET 6 app, add the Syncfusion bootstrap5 theme in the `element` of the `~/Pages/_Layout.cshtml` page.

ASPX-CS

```
<head>
....
<link href="_content/Syncfusion.Blazor.Themes/bootstrap5.css"
rel="stylesheet" />
</head>
```

- For .NET 5 and .NET 3.X app, add the Syncfusion bootstrap5 theme in the `element` of the `~/Pages/_Host.cshtml` page.

ASPX-CS

```
<head>
....
<link href="_content/Syncfusion.Blazor.Themes/bootstrap5.css"
rel="stylesheet" />
</head>
```

Blazor WebAssembly App

The theme style sheet from NuGet can be referred inside the `<head>` element of `wwwroot/index.html` file of client web app.

HTML

```
<head>
...
<link href="_content/Syncfusion.Blazor.Themes/bootstrap5.css"
rel="stylesheet" />
</head>
```

Add Script Reference

Checkout [Adding Script Reference topic](#) to learn different ways to add script reference in Blazor Application. In this getting started walk-through, the required scripts are referenced automatically via javascript script isolation approach.

Syncfusion recommends to reference scripts using [Static Web Assets](#), [CDN](#) and [CRG](#) by [disabling JavaScript isolation](#) for better loading performance of the Blazor application.

Register Syncfusion Blazor Service

Open `~/_Imports.razor` file and import the `Syncfusion.Blazor` namespace.

C#

```
@using Syncfusion.Blazor
```

Blazor Server App

Now, register the Syncfusion Blazor Service in the Blazor Server App.

- For .NET 6 app, open the **~/Program.cs** file and register the Syncfusion Blazor Service.

C#

```
using Microsoft.AspNetCore.Components;  
using Microsoft.AspNetCore.Components.Web;  
using Syncfusion.Blazor;  
var builder = WebApplication.CreateBuilder(args);  
// Add services to the container.  
builder.Services.AddRazorPages();  
builder.Services.AddServerSideBlazor();  
builder.Services.AddSyncfusionBlazor();  
var app = builder.Build();  
....
```

- For .NET 5 and .NET 3.X app, open the **~/Startup.cs** file and register the Syncfusion Blazor Service.

C#

```
using Syncfusion.Blazor;  
namespace BlazorApplication  
{  
    public class Startup  
    {  
        ...  
        public void ConfigureServices(IServiceCollection services)  
        {  
            services.AddRazorPages();  
            services.AddServerSideBlazor();  
            services.AddSyncfusionBlazor();  
        }  
        ...  
    }  
}
```

Blazor WebAssembly App

Open **~/Program.cs** file and register the Syncfusion Blazor Service in the client web app.

- For .NET 6 app,

C#

```
using Microsoft.AspNetCore.Components.Web;  
using Microsoft.AspNetCore.Components.WebAssembly.Hosting;  
using Syncfusion.Blazor;  
var builder = WebAssemblyHostBuilder.CreateDefault(args);
```

```
builder.RootComponents.Add<App>("#app");
builder.RootComponents.Add<HeadOutlet>("head::after");
builder.Services.AddScoped(sp => new HttpClient { BaseAddress = new
Uri(builder.HostEnvironment.BaseAddress) });
builder.Services.AddSyncfusionBlazor();
await builder.Build().RunAsync();
....
```

- For .NET 5 and .NET 3.X app

C#

```
using Syncfusion.Blazor;
namespace WebApplication1
{
    public class Program
    {
        public static async Task Main(string[] args)
        {
            ....
            builder.Services.AddSyncfusionBlazor();
            await builder.Build().RunAsync();
        }
    }
}
```

Add Syncfusion Blazor Signature component

- Open ~/_Imports.razor file or any razor page under the ~/Pages folder where the component is to be added and import the Syncfusion.Blazor.Inputs namespace.

C#

```
@using Syncfusion.Blazor
@using Syncfusion.Blazor.Inputs
```

- Now, add the Syncfusion Signature component in razor file. Here, the Signature component is added in the ~/Pages/Index.razor page under the ~/Pages folder.

C#

```
<SfSignature></SfSignature>
```

- Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion Blazor Signature component will be rendered in the default web browser.



The Signature component will render default height and width of canvas (300 * 150), when the Signature height and width are not specified.

See Also

- [Getting Started with Syncfusion Blazor for client side in .NET Core CLI](#)
- [Getting Started with Syncfusion Blazor for server side in Visual Studio](#)
- [Getting Started with Syncfusion Blazor for server side in .NET Core CLI](#)

Smith Chart

Getting Started with Blazor Smith Chart Component

This section briefly explains how to include a Smith Chart component in the Blazor server-side application. Refer to the [Getting Started with Syncfusion Blazor for server-side in Visual Studio](#) page for introduction and configuring common specifications.

Importing Syncfusion Blazor Smith Chart component in the application

1. Install **Syncfusion.Blazor** NuGet package to the application by using the **NuGet Package Manager**.
2. Add the client-side resources through [CDN](#) or from [NuGet](#) package in the **HEAD** element of the `~/Pages/_Host.cshtml` page.

HTML

```
<head>
<link href="_content/Syncfusion.Blazor.Themes/bootstrap4.css"
rel="stylesheet" />
<!--CDN-->
@*<link href="https://cdn.syncfusion.com/blazor/{{ site.blazorversion
}}/styles/bootstrap4.css" rel="stylesheet" />*@
</head>
```

For the Internet Explorer 11, kindly refer to the polyfills. Refer to the [documentation](#) for more information.

HTML

```
<head>
<link href="https://cdn.syncfusion.com/blazor/{{ site.blazorversion
}}/styles/bootstrap4.css" rel="stylesheet" />
<script
src="https://github.com/Daddoon/Blazor.Polyfill/releases/download/3.0.1/blaz
or.polyfill.min.js"></script>
</head>
```

Adding component package to the application

Open the `~/_Imports.razor` file and include the **Syncfusion.Blazor.Charts** namespace.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
```

Adding SyncfusionBlazor service in Startup.cs

Open the **Startup.cs** file and add the services required by Syncfusion components using the **services.AddSyncfusionBlazor()** method. Add this method in the **ConfigureServices** function as follows.

CSHARP

```
using Syncfusion.Blazor;
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void ConfigureServices(IServiceCollection services)
        {
            ....
            ....
            services.AddSyncfusionBlazor();
        }
    }
}
```

To enable custom client-side source loading from CRG or CDN, please refer to the section about [custom resources in Blazor application](#).

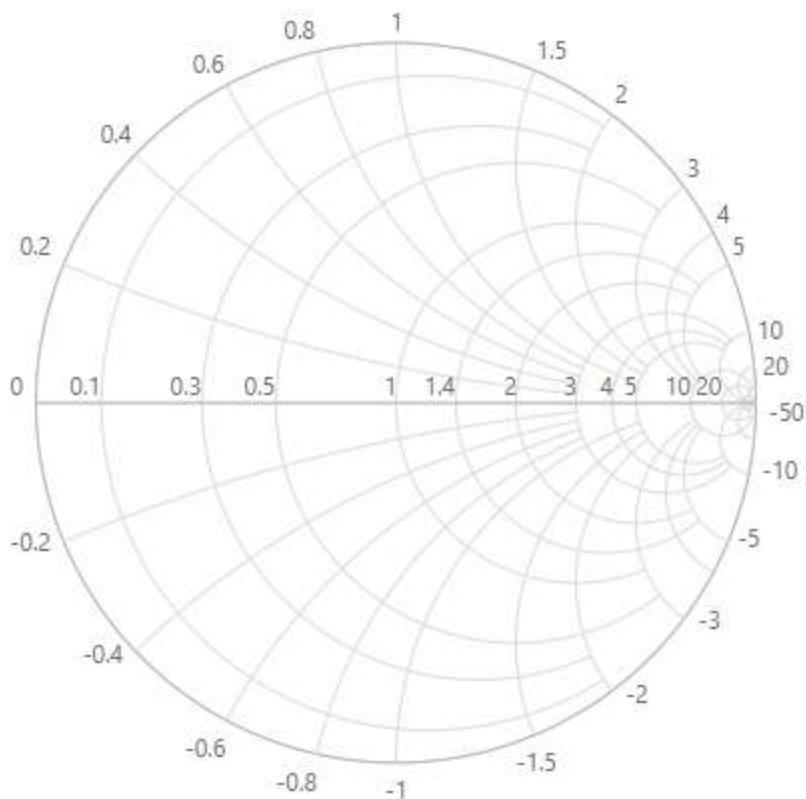
Adding Smith Chart component

To initialize the Smith Chart component, add the below code to the **Index.razor** view page under **~/Pages** folder. In a new application, if **Index.razor** page has any default content template, then those content can be completely removed and following code can be added.

ASPX-CS

```
@page "/"
<SfSmithchart>
</SfSmithchart>
```

On successful compilation of the application, the Syncfusion Blazor Smith Chart component will render in the web browser as following.



Adding series to Smith Chart

Smith Chart series can be added in two ways. Use either [Points](#) or [Datasource](#) in the [SmithChartSeries](#).

If you add using [Datasource](#) property, additionally you need to specify data source mapping fields using [Reactance](#) and [Resistance](#) properties.

If you are using [Points](#), you don't need to specify mapping fields as like in [DataSource](#). But the [Points](#) collection should be [SmithChartPoint](#) type and define [Resistance](#) and [Reactance](#) properties mandatorily.

The following sample demonstrates adding two series to Smith Chart in both ways.

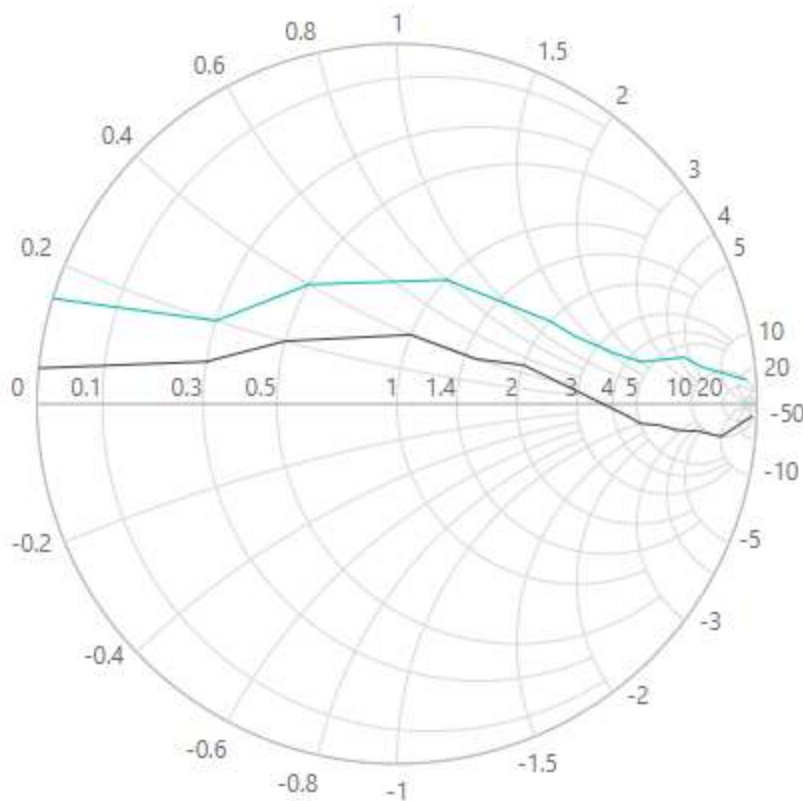
- First series [Transmission1](#) shows [DataSource](#) bound series.
- Second series [Transmission2](#) shows [Points](#) bound series.

ASPX-CS

```
<SfSmithChart>
  <SmithChartSeriesCollection>
    <SmithChartSeries Name="Transmission1"
      Reactance="Reactance"
      Resistance="Resistance"
      DataSource="@FirstTransmissionSeries">
    </SmithChartSeries>
    <SmithChartSeries Name="Transmission2"
      Points="@SecondTransmissionSeries">
    </SmithChartSeries>
```



```
</SmithChartSeriesCollection>
</SfSmithChart>
@code {
public class SmithDataSource
{
public double Resistance { get; set; }
public double Reactance { get; set; }
};
public List<SmithDataSource> FirstTransmissionSeries = new
List<SmithDataSource> {
new SmithDataSource { Resistance= 10, Reactance= 25 },
new SmithDataSource { Resistance= 8, Reactance= 6 },
new SmithDataSource { Resistance= 6, Reactance= 4.5 },
new SmithDataSource { Resistance= 4.5, Reactance= 2 },
new SmithDataSource { Resistance= 3.5, Reactance= 1.6 },
new SmithDataSource { Resistance= 2.5, Reactance= 1.3 },
new SmithDataSource { Resistance= 2, Reactance= 1.2 },
new SmithDataSource { Resistance= 1.5, Reactance= 1 },
new SmithDataSource { Resistance= 1, Reactance= 0.8 },
new SmithDataSource { Resistance= 0.5, Reactance= 0.4 },
new SmithDataSource { Resistance= 0.3, Reactance= 0.2 },
new SmithDataSource { Resistance= 0.001, Reactance= 0.15 }
};
public List<SmithChartPoint> SecondTransmissionSeries = new
List<SmithChartPoint> {
new SmithChartPoint { Resistance= 20, Reactance= -50 },
new SmithChartPoint { Resistance= 10, Reactance= -10 },
new SmithChartPoint { Resistance= 9, Reactance= -4.5 },
new SmithChartPoint { Resistance= 8, Reactance= -3.5 },
new SmithChartPoint { Resistance= 7, Reactance= -2.5 },
new SmithChartPoint { Resistance= 6, Reactance= -1.5 },
new SmithChartPoint { Resistance= 5, Reactance= -1 },
new SmithChartPoint { Resistance= 4.5, Reactance= -0.5 },
new SmithChartPoint { Resistance= 2, Reactance= 0.5 },
new SmithChartPoint { Resistance= 1.5, Reactance= 0.4 },
new SmithChartPoint { Resistance= 1, Reactance= 0.4 },
new SmithChartPoint { Resistance= 0.5, Reactance= 0.2 },
new SmithChartPoint { Resistance= 0.3, Reactance= 0.1 },
new SmithChartPoint { Resistance= 0.001, Reactance= 0.05 }
};
}
```



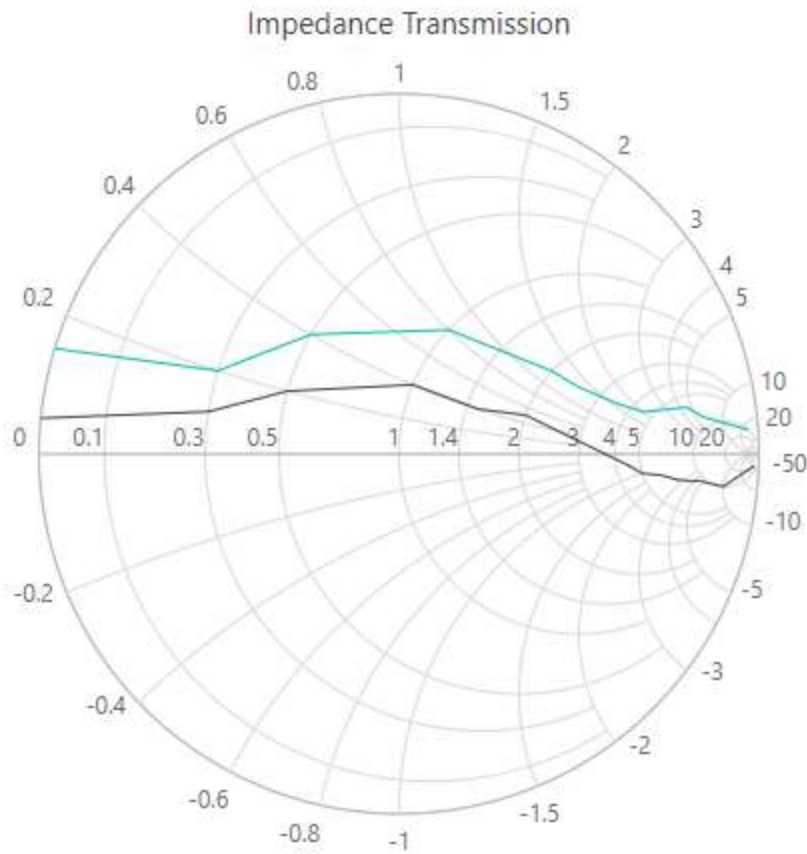
Adding Title

Title can be added to the Smith Chart to provide a quick information to the users about the context of the rendered component. Add a title by using the [Text](#) property in the [SmithChartTitle](#).

ASPX-CS

```
<SfSmithChart>
  <SmithChartTitle Text="Impedance Transmission">
  </SmithChartTitle>
  <SmithChartSeriesCollection>
    <SmithChartSeries Name="Transmission1"
      Reactance="Reactance"
      Resistance="Resistance"
      DataSource="@FirstTransmissionSeries">
    </SmithChartSeries>
    <SmithChartSeries Name="Transmission2"
      Points="@SecondTransmissionSeries">
    </SmithChartSeries>
  </SmithChartSeriesCollection>
</SfSmithChart>
```

Refer to the [code block](#) to know about the property value of `FirstTransmissionSeries` and `SecondTransmissionSeries`.



Enable Marker

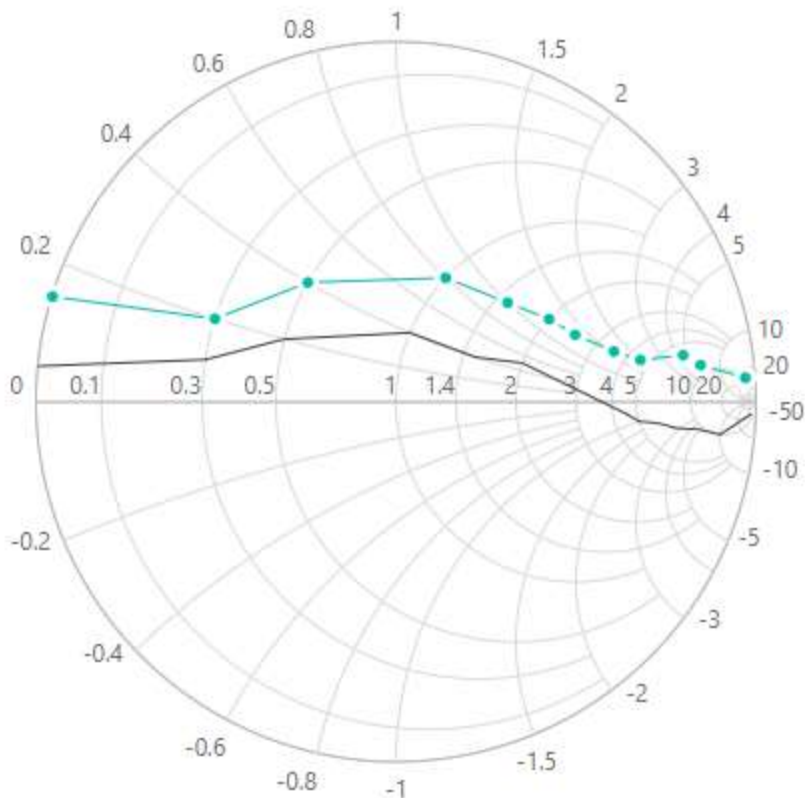
To display marker for particular series, set the [Visible](#) property to **true** in the [SmithChartSeriesMarker](#).

In the following example, marker is enabled for first series only.

ASPX-CS

```
<SfSmithChart>
  <SmithChartTitle Text="Impedance Transmission">
  </SmithChartTitle>
  <SmithChartSeriesCollection>
    <SmithChartSeries Name="Transmission1"
      Reactance="Reactance"
      Resistance="Resistance"
      DataSource="@FirstTransmissionSeries">
      <SmithChartSeriesMarker Visible="true"></SmithChartSeriesMarker>
    </SmithChartSeries>
    <SmithChartSeries Name="Transmission2"
      Points="@SecondTransmissionSeries">
    </SmithChartSeries>
  </SmithChartSeriesCollection>
</SfSmithChart>
```

Refer to the [code block](#) to know about the property value of `FirstTransmissionSeries` and `SecondTransmissionSeries`.



Enable Data Label

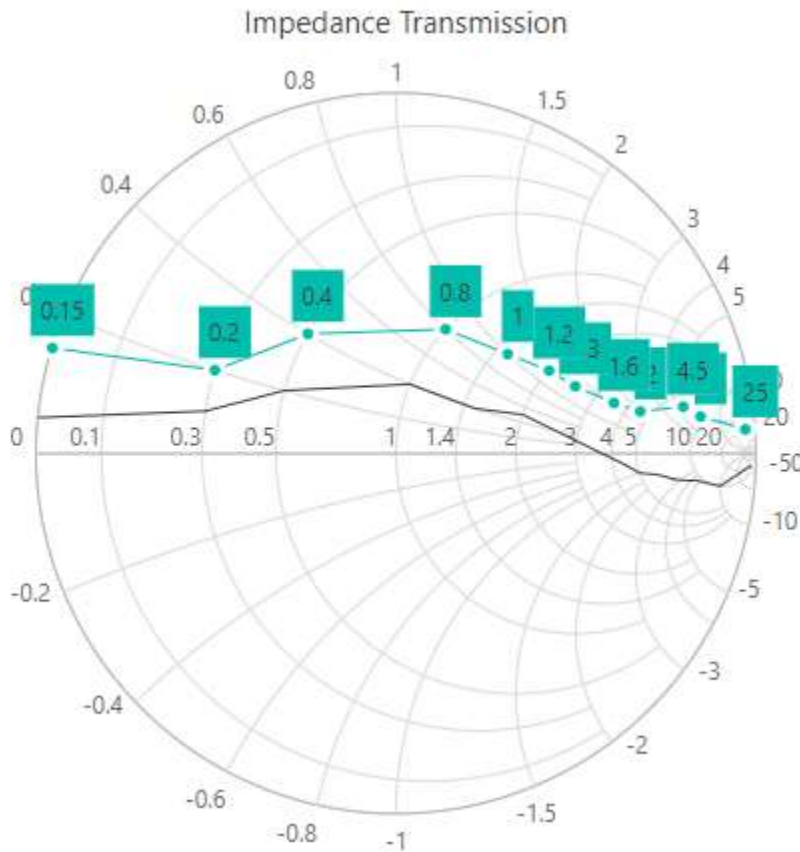
To display data label for particular marker series, set the [Visible](#) property to **true** in the [SmithChartSeriesDatalabel](#).

In the following example, data label is enabled for the first series only.

ASPX-CS

```
<SfSmithChart>
  <SmithChartTitle Text="Impedance Transmission">
  </SmithChartTitle>
  <SmithChartSeriesCollection>
    <SmithChartSeries Name="Transmission1"
      Reactance="Reactance"
      Resistance="Resistance"
      DataSource="@FirstTransmissionSeries">
      <SmithChartSeriesMarker Visible="true">
      <SmithChartSeriesDatalabel Visible="true">
      </SmithChartSeriesDatalabel>
      </SmithChartSeriesMarker>
      </SmithChartSeries>
    <SmithChartSeries Name="Transmission2" Points="@SecondTransmissionSeries">
    </SmithChartSeries>
  </SmithChartSeriesCollection>
</SfSmithChart>
```

Refer to the [code block](#) to know the property value of `FirstTransmissionSeries` and `SecondTransmissionSeries`.



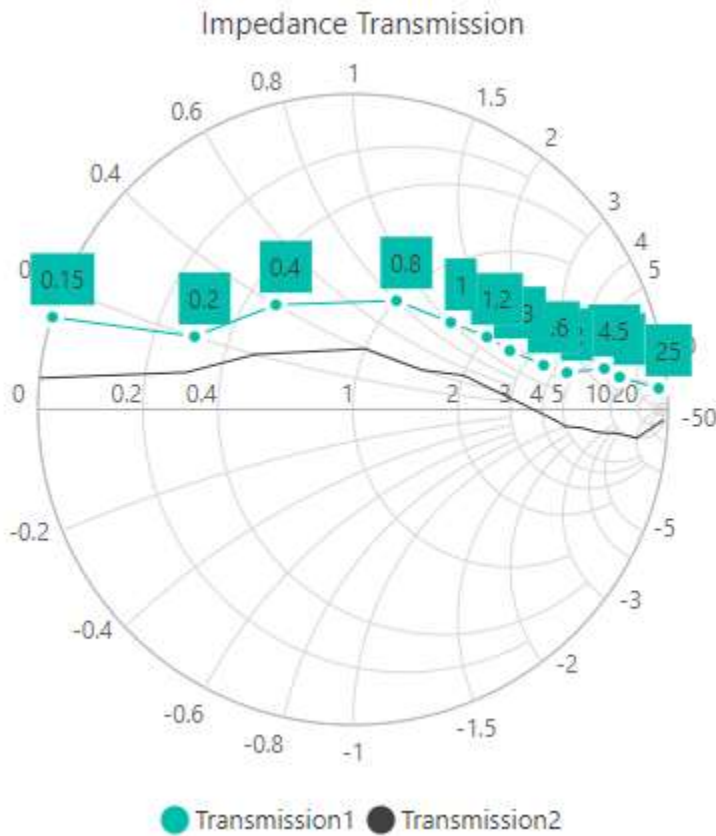
Enable Legend

Use legend for the Smith Chart by setting the `Visible` property to `true` in the [SmithChartLegendSettings](#). The legend name can be changed by using the `Name` property in the [SmithChartSeries](#).

ASPX-CS

```
<SfSmithChart>
  <SmithChartLegendSettings Visible="true"></SmithChartLegendSettings>
  <SmithChartTitle Text="Impedance Transmission"></SmithChartTitle>
  <SmithChartSeriesCollection>
    <SmithChartSeries Name="Transmission1"
      Reactance="Reactance"
      Resistance="Resistance"
      DataSource="@FirstTransmissionSeries">
      <SmithChartSeriesMarker Visible="true">
        <SmithChartSeriesDatalabel Visible="true">
        </SmithChartSeriesDatalabel>
      </SmithChartSeriesMarker>
    </SmithChartSeries>
    <SmithChartSeries Name="Transmission2" Points="@SecondTransmissionSeries">
    </SmithChartSeries>
  </SmithChartSeriesCollection>
</SfSmithChart>
```

Refer to the [code block](#) to know the property value of the `FirstTransmissionSeries` and the `SecondTransmissionSeries`.



Enable Tooltip

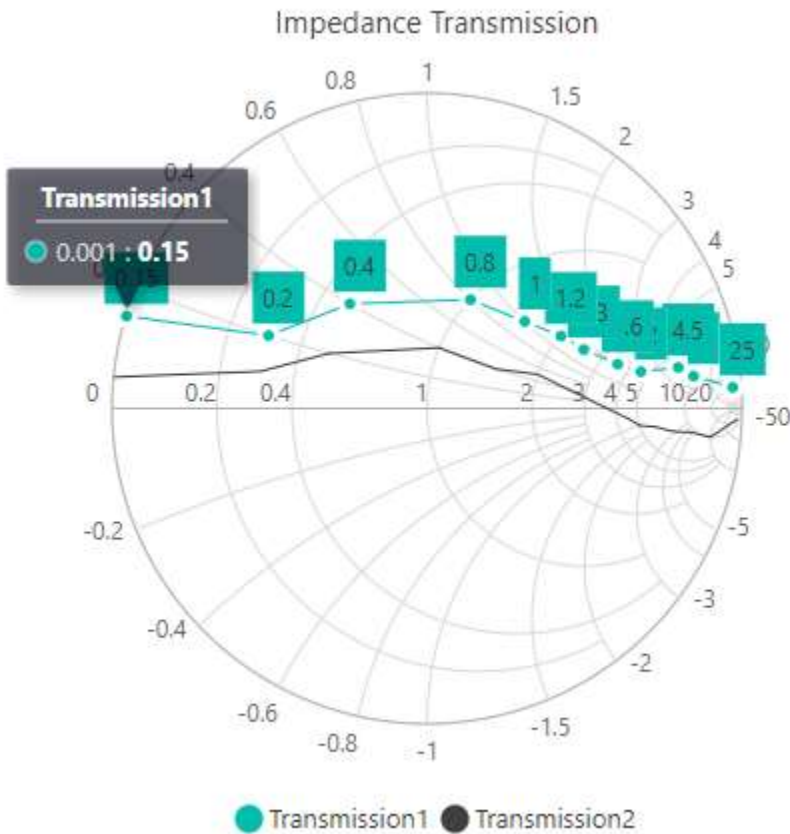
When space constraints prevents from displaying information using data labels, the tooltip comes in handy. The tooltip can be enabled by setting the [Visible](#) property to `true` in the [SmithChartSeriesTooltip](#).

ASPX-CS

```
<SfSmithChart>
  <SmithChartLegendSettings Visible="true"></SmithChartLegendSettings>
  <SmithChartTitle Text="Impedance Transmission"></SmithChartTitle>
  <SmithChartSeriesCollection>
    <SmithChartSeries Name="Transmission1"
      Reactance="Reactance"
      Resistance="Resistance"
      DataSource="@FirstTransmissionSeries">
      <SmithChartSeriesMarker Visible="true">
      <SmithChartSeriesDatalabel Visible="true">
      </SmithChartSeriesDatalabel>
      </SmithChartSeriesMarker>
      <SmithChartSeriesTooltip Visible="true">
      </SmithChartSeriesTooltip>
    </SmithChartSeries>
    <SmithChartSeries Name="Transmission2" Points="@SecondTransmissionSeries">
    </SmithChartSeries>
  </SmithChartSeriesCollection>
</SfSmithChart>
```

```
</SfSmithChart>
```

Refer to the [code block](#) to know about the property value of the `FirstTransmissionSeries` and the `SecondTransmissionSeries`.



See also

- [Getting Started with Syncfusion Blazor for Client-Side in .NET Core CLI](#)
- [Getting Started with Syncfusion Blazor for Server-Side in Visual Studio](#)
- [Getting Started with Syncfusion Blazor for Server-Side in .NET Core CLI](#)

Dimensions in Blazor Smith Chart Component

The dimensions of the Smith Chart can be modified in the following ways.

- Using CSS
- Using API

Using CSS

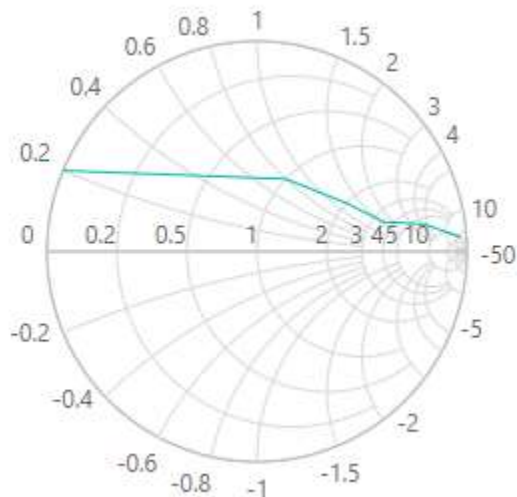
To set the size using CSS, add an [ID](#) to the `SfSmithChart` tag, and set the width and the height of the Smith Chart in the style tag as following.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfSmithChart ID="smChart">
  <SmithChartSeriesCollection>
    <SmithChartSeries DataSource='TransmissionData' Reactance="Reactance"
    Resistance="Resistance">
    </SmithChartSeries>
  </SmithChartSeriesCollection>
</SfSmithChart>
<style>
#smChart {
height: 300px !important;
width: 300px !important;
}
</style>
@code {
public class SmithChartData
{
public double? Resistance { get; set; }
public double? Reactance { get; set; }
};
public List<SmithChartData> TransmissionData = new List<SmithChartData> {
new SmithChartData { Resistance= 10, Reactance= 25 },
new SmithChartData { Resistance= 6, Reactance= 4.5 },
new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
new SmithChartData { Resistance= 2, Reactance= 1.2 },
new SmithChartData { Resistance= 1, Reactance= 0.8 },
new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
}

```



Using API

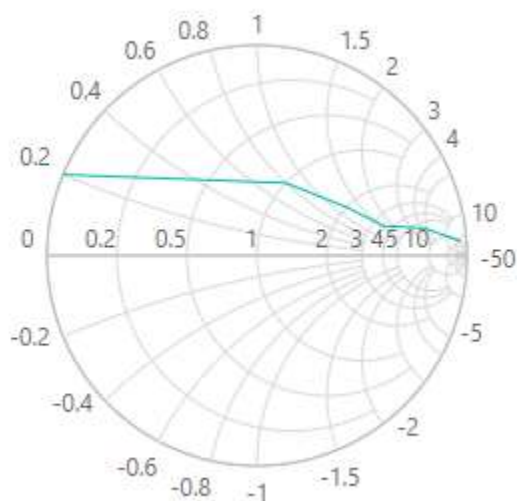
The width and the height of the Smith Chart can also be set directly using the [Width](#) and the [Height](#) properties respectively. It can be in pixel or in percentage.

In Pixel

The [Width](#) and the [Height](#) properties in the Smith Chart can be directly given in pixels, as following.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSmithChart Height="300px" Width="300px">
  <SmithChartSeriesCollection>
    <SmithChartSeries DataSource='TransmissionData' Reactance="Reactance"
      Resistance="Resistance">
    </SmithChartSeries>
  </SmithChartSeriesCollection>
</SfSmithChart>
@code {
  public class SmithChartData
  {
    public double? Resistance { get; set; }
    public double? Reactance { get; set; }
  };
  public List<SmithChartData> TransmissionData = new List<SmithChartData> {
    new SmithChartData { Resistance= 10, Reactance= 25 },
    new SmithChartData { Resistance= 6, Reactance= 4.5 },
    new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
    new SmithChartData { Resistance= 2, Reactance= 1.2 },
    new SmithChartData { Resistance= 1, Reactance= 0.8 },
    new SmithChartData { Resistance= 0, Reactance= 0.2 }
  };
}
```

*In percentage*

The Smith Chart's [Width](#) and [Height](#) properties can be directly given in percentage, as shown in the following. The component will be rendered as a percentage of its container size.

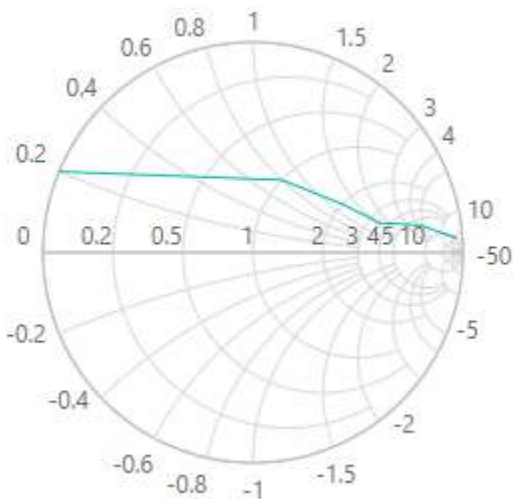
ASPX-CS

```
@using Syncfusion.Blazor.Charts
<div style="height:600px; width:600px">
```

```

<SfSmithChart Height="50%" Width="50%">
  <SmithChartSeriesCollection>
    <SmithChartSeries DataSource='FirstTransmissionData' Reactance="Reactance"
      Resistance="Resistance">
    </SmithChartSeries>
  </SmithChartSeriesCollection>
</SfSmithChart>
</div>
@code {
public class SmithChartData
{
public double? Resistance { get; set; }
public double? Reactance { get; set; }
};
public List<SmithChartData> FirstTransmissionData = new List<SmithChartData>
{
new SmithChartData { Resistance= 10, Reactance= 25 },
new SmithChartData { Resistance= 6, Reactance= 4.5 },
new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
new SmithChartData { Resistance= 2, Reactance= 1.2 },
new SmithChartData { Resistance= 1, Reactance= 0.8 },
new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
}

```



Title and Subtitle in Blazor Smith Chart Component

Enable Title

The information about the data plotted in the Smith Chart is depicted using the titles and the subtitles. Using the [Text](#) property in the [SmithChartTitle](#) and the [SmithChartSubtitle](#), the Smith Chart's title and subtitle can be changed. By default, the title and the subtitles are visible. As shown in the following example, use the simple text for the title and the subtitles.

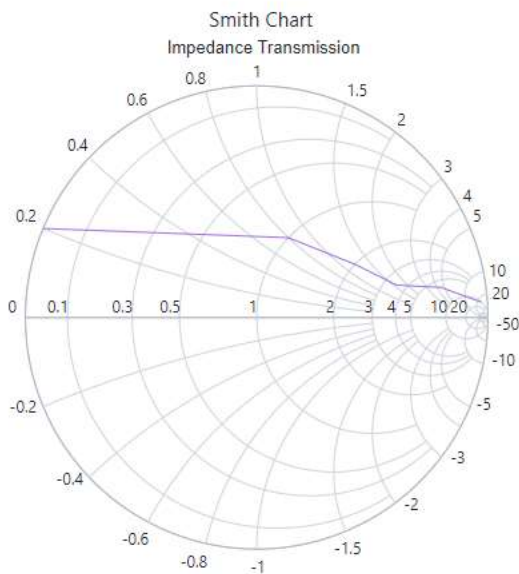
ASPX-CS

```
@using Syncfusion.Blazor.Charts
```

```

<SfSmithChart>
  <SmithChartTitle Text="Smith Chart">
  <SmithChartSubtitle Text="Impedance Transmission"></SmithChartSubtitle>
</SmithChartTitle>
  <SmithChartSeriesCollection>
  <SmithChartSeries DataSource='TransmissionData' Reactance="Reactance"
Resistance="Resistance">
</SmithChartSeries>
</SmithChartSeriesCollection>
</SfSmithChart>
@code {
public class SmithChartData
{
public double? Resistance { get; set; }
public double? Reactance { get; set; }
};
public List<SmithChartData> TransmissionData = new List<SmithChartData> {
new SmithChartData { Resistance= 10, Reactance= 25 },
new SmithChartData { Resistance= 6, Reactance= 4.5 },
new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
new SmithChartData { Resistance= 2, Reactance= 1.2 },
new SmithChartData { Resistance= 1, Reactance= 0.8 },
new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
}

```



Trim Title

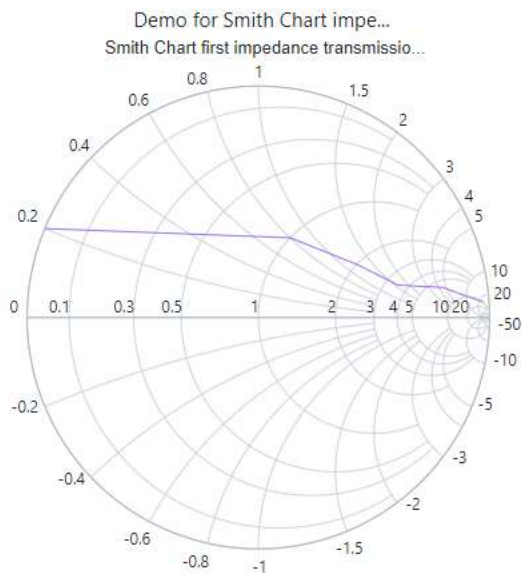
Both the title and the subtitle of the Smith Chart can be trimmed if it exceeds certain length. This length can be changed using the [MaximumWidth](#) property. Trimming is enabled by setting the [EnableTrim](#) property to **true**.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfSmithChart>
<SmithChartTitle Text="Demo for Smith Chart impedance transmission"
EnableTrim="true"
MaximumWidth="200">
<SmithChartSubtitle Text="Smith Chart first impedance transmission. For more
info."
EnableTrim="true"
MaximumWidth="250">
</SmithChartSubtitle>
</SmithChartTitle>
<SmithChartSeriesCollection>
<SmithChartSeries DataSource='TransmissionData' Reactance="Reactance"
Resistance="Resistance">
</SmithChartSeries>
</SmithChartSeriesCollection>
</SfSmithChart>
@code {
public class SmithChartData
{
public double? Resistance { get; set; }
public double? Reactance { get; set; }
};
public List<SmithChartData> TransmissionData = new List<SmithChartData> {
new SmithChartData { Resistance= 10, Reactance= 25 },
new SmithChartData { Resistance= 6, Reactance= 4.5 },
new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
new SmithChartData { Resistance= 2, Reactance= 1.2 },
new SmithChartData { Resistance= 1, Reactance= 0.8 },
new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
}

```



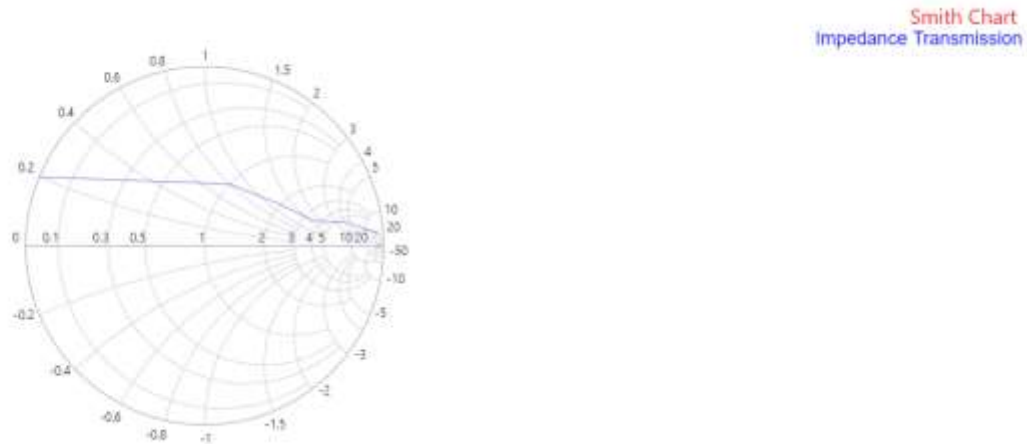
Title Customization

Title and subtitle can be customized using the following properties.

- [TextAlignment](#) - It can align the Smith Chart's title text in near, centre, and far positions. By default, it is aligned in the **Center** position.
- [SmithChartTitleTextStyle](#) - Used to customize the properties such as [FontFamily](#), [FontWeight](#), [FontStyle](#), [Opacity](#), [Color](#), and [Size](#) for title text.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSmithChart>
  <SmithChartTitle Text="Smith Chart"
  TextAlignment="@SmithChartAlignment.Far">
  <SmithChartTitleTextStyle Color="red"
  Size="20px"></SmithChartTitleTextStyle>
  <SmithChartSubtitle Text="Impedance Transmission"
  TextAlignment="@SmithChartAlignment.Far">
  <SmithChartSubtitleTextStyle Color="blue"
  Size="18px"></SmithChartSubtitleTextStyle>
</SmithChartSubtitle>
</SmithChartTitle>
<SmithChartSeriesCollection>
  <SmithChartSeries DataSource='TransmissionData' Reactance="Reactance"
  Resistance="Resistance">
</SmithChartSeries>
</SmithChartSeriesCollection>
</SfSmithChart>
@code {
public class SmithChartData
{
public double? Resistance { get; set; }
public double? Reactance { get; set; }
};
public List<SmithChartData> TransmissionData = new List<SmithChartData> {
new SmithChartData { Resistance= 10, Reactance= 25 },
new SmithChartData { Resistance= 6, Reactance= 4.5 },
new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
new SmithChartData { Resistance= 2, Reactance= 1.2 },
new SmithChartData { Resistance= 1, Reactance= 0.8 },
new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
}
```



Axis in Blazor Smith Chart Component

Smith Chart supports two different types of axes. They are:

- **Horizontal Axis** - The axis is drawn as a straight line in the horizontal direction of the Smith Chart.
- **Radial Axis** - The axis is drawn as a circular path.

Labels Customization

Axis labels are used to indicate what type of data is bound for the Smith Chart. The use of axis labels makes it easy to determine at which interval chart is being rendered. The axis labels for the horizontal and radial axes can be customized using the properties listed as following.

- [Visible](#) - Used to specify the visibility of the axis.
- [LabelPosition](#) - Used to place labels either inside or outside the axis line.
- [LabelIntersectAction](#) - Used to hide labels when intersecting.
- [SmithChartRadialAxisLabelStyle](#) and [SmithChartHorizontalAxisLabelStyle](#) - Used to customize properties such as [FontFamily](#), [FontWeight](#), [FontStyle](#), [Opacity](#), [Color](#), and [Size](#).

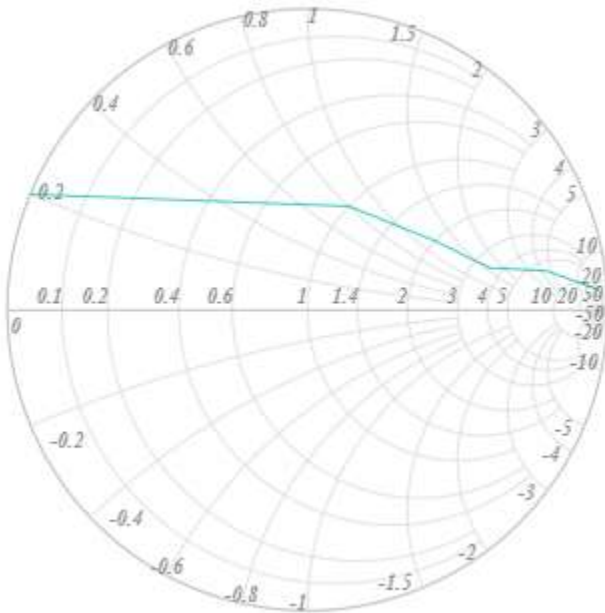
ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSmithChart>
  <SmithChartHorizontalAxis>
    <SmithChartHorizontalAxisLabelStyle FontFamily="Times New Roman"
    FontWeight="bold"
    FontStyle="Italic"
    Opacity='0.75'
    Size="14px">
  </SmithChartHorizontalAxisLabelStyle>
</SmithChartHorizontalAxis>
  <SmithChartRadialAxis LabelPosition='AxisLabelPosition.Inside'
  LabelIntersectAction='SmithChartLabelIntersectAction.None'>
    <SmithChartRadialAxisLabelStyle FontFamily="Times New Roman"
    FontWeight="bold"
    FontStyle="Italic"
    Opacity='0.75'
    Size="14px">
```

```

</SmithChartRadialAxisLabelStyle>
</SmithChartRadialAxis>
<SmithChartSeriesCollection>
<SmithChartSeries DataSource='TransmissionData' Reactance="Reactance"
Resistance="Resistance">
</SmithChartSeries>
</SmithChartSeriesCollection>
</SfSmithChart>
@code {
public class SmithChartData
{
public double? Resistance { get; set; }
public double? Reactance { get; set; }
};
public List<SmithChartData> TransmissionData = new List<SmithChartData> {
new SmithChartData { Resistance= 10, Reactance= 25 },
new SmithChartData { Resistance= 6, Reactance= 4.5 },
new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
new SmithChartData { Resistance= 2, Reactance= 1.2 },
new SmithChartData { Resistance= 1, Reactance= 0.8 },
new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
}

```



Gridlines

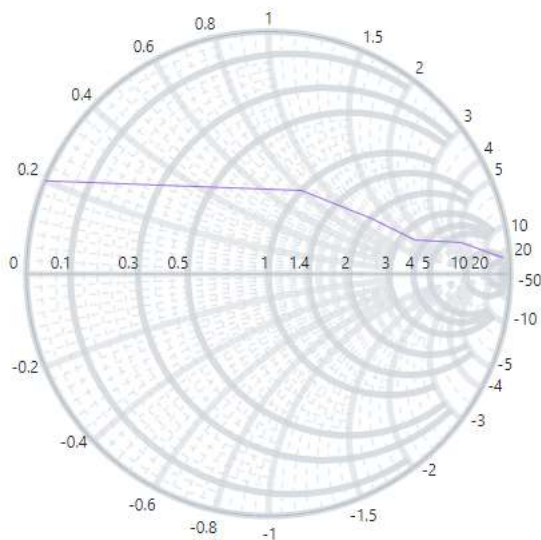
Gridlines on the horizontal and the radial axes can be used to make data in a chart easier to see. Gridlines extend from any horizontal or radial axis around the plot area of the Smith Chart. Both the horizontal and radial axes support major and minor gridlines. Major gridlines are drawn from the position in which the labels are rendered. Minor gridlines are drawn between two major gridlines using the [Count](#) property in the minor gridlines.

The following properties can be customized in both the major and minor gridlines.

- [Width](#) - Used to customize the width of the gridlines.
- [DashArray](#) - Used to customize whether the gridline has to render as normal line or dashed line.
- [Visible](#) - Used to enable or disable the visibility of the gridlines.
- [Opacity](#) - Used to customize the opacity of the major gridlines.
- [Color](#) - Used to customize the color of the major gridlines.
- [Count](#) - Used to customize the count of the minor gridlines.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSmithChart>
  <SmithChartHorizontalAxis>
    <SmithChartHorizontalMajorGridLines Visible='true' Opacity='0.8' Width='5'>
    </SmithChartHorizontalMajorGridLines>
    <SmithChartHorizontalMinorGridLines Visible='true' DashArray="5" Count="10">
    </SmithChartHorizontalMinorGridLines>
  </SmithChartHorizontalAxis>
  <SmithChartRadialAxis>
    <SmithChartRadialMajorGridLines Visible='true' Opacity='0.5' Width='5'>
    </SmithChartRadialMajorGridLines>
    <SmithChartRadialMinorGridLines Visible='true' DashArray="5" Count="10">
    </SmithChartRadialMinorGridLines>
  </SmithChartRadialAxis>
  <SmithChartSeriesCollection>
    <SmithChartSeries DataSource='TransmissionData' Reactance="Reactance"
    Resistance="Resistance">
    </SmithChartSeries>
  </SmithChartSeriesCollection>
</SfSmithChart>
@code {
public class SmithChartData
{
public double? Resistance { get; set; }
public double? Reactance { get; set; }
};
public List<SmithChartData> TransmissionData = new List<SmithChartData> {
new SmithChartData { Resistance= 10, Reactance= 25 },
new SmithChartData { Resistance= 6, Reactance= 4.5 },
new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
new SmithChartData { Resistance= 2, Reactance= 1.2 },
new SmithChartData { Resistance= 1, Reactance= 0.8 },
new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
}
```

Axis Line

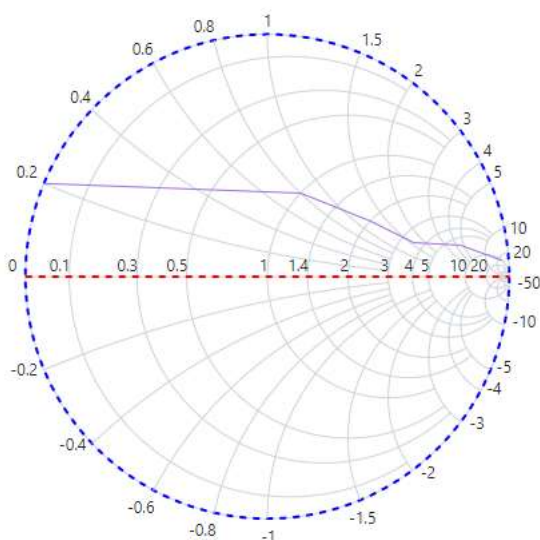
By default, the visibility of the axis line is **true**. Its visibility can be changed using the [Visible](#) property. Other than the visibility, the following properties can be used to customize the axis line.

- [Width](#) - Used to customize the width of the axis line.
- [DashArray](#) - Used to render the axis line as dashed line.
- [Visible](#) - Used to enable or disable the visibility of the axis line.
- [Color](#) - Used to customize the axis line color.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSmithChart>
  <SmithChartHorizontalAxis>
    <SmithChartHorizontalAxisLine Width="2" Visible="true" DashArray="5"
    Color="blue">
  </SmithChartHorizontalAxisLine>
  </SmithChartHorizontalAxis>
  <SmithChartRadialAxis>
    <SmithChartRadialAxisLine Width="2" Visible="true" DashArray="5"
    Color="red">
  </SmithChartRadialAxisLine>
  </SmithChartRadialAxis>
  <SmithChartSeriesCollection>
    <SmithChartSeries DataSource='TransmissionData' Reactance="Reactance"
    Resistance="Resistance">
  </SmithChartSeries>
  </SmithChartSeriesCollection>
</SfSmithChart>
@code {
public class SmithChartData
{
public double? Resistance { get; set; }
public double? Reactance { get; set; }
```

```
};
public List<SmithChartData> TransmissionData = new List<SmithChartData> {
    new SmithChartData { Resistance= 10, Reactance= 25 },
    new SmithChartData { Resistance= 6, Reactance= 4.5 },
    new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
    new SmithChartData { Resistance= 2, Reactance= 1.2 },
    new SmithChartData { Resistance= 1, Reactance= 0.8 },
    new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
}
```



Legend in Blazor Smith Chart Component

In the Smith Chart, a legend is a key containing symbols and descriptions. It can be interpreted in various colors, shapes, or other identifiers based on the data, and it provides valuable information for interpreting what the Smith Chart is displaying. In simple words, the legend is used to denote the series rendered in the Smith Chart.

Position

By default, the visibility of the legend is **false**. To enable the legend, kindly set the [Visible](#) property to **true** in the [SmithChartLegendSettings](#). The default position for the legend is **Bottom**. By using the [Position](#) property, the position of the legend can be changed. The legend can be placed on the Smith Chart's bottom, top, right, or left side.

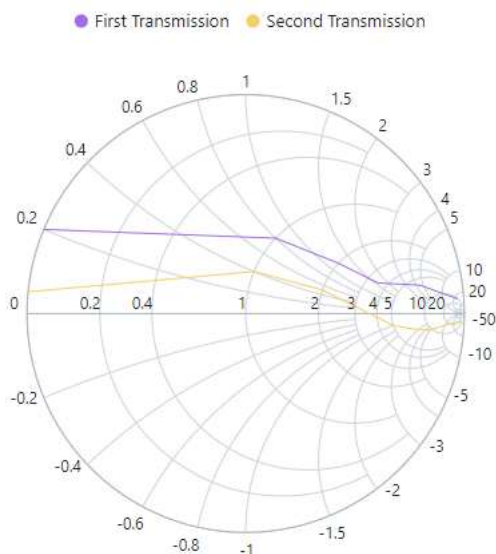
ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSmithChart>
    <SmithChartLegendSettings Visible='true' Position='@LegendPosition.Top'>
    </SmithChartLegendSettings>
    <SmithChartSeriesCollection>
    <SmithChartSeries Name="First Transmission"
        DataSource='FirstTransmissionData' Reactance="Reactance"
        Resistance="Resistance"></SmithChartSeries>
```

```

<SmithChartSeries Name="Second Transmission"
DataSource='SecondTransmissionData' Reactance="Reactance"
Resistance="Resistance"></SmithChartSeries>
</SmithChartSeriesCollection>
</SfSmithChart>
@code {
public class SmithChartData
{
public double? Resistance { get; set; }
public double? Reactance { get; set; }
};
public List<SmithChartData> FirstTransmissionData = new List<SmithChartData>
{
new SmithChartData { Resistance= 10, Reactance= 25 },
new SmithChartData { Resistance= 6, Reactance= 4.5 },
new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
new SmithChartData { Resistance= 2, Reactance= 1.2 },
new SmithChartData { Resistance= 1, Reactance= 0.8 },
new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
public List<SmithChartData> SecondTransmissionData = new
List<SmithChartData> {
new SmithChartData { Resistance= 20, Reactance= -50 },
new SmithChartData { Resistance= 9, Reactance= -4.5 },
new SmithChartData { Resistance= 7, Reactance= -2.5 },
new SmithChartData { Resistance= 5, Reactance= -1 },
new SmithChartData { Resistance= 2, Reactance= 0.5 },
new SmithChartData { Resistance= 1, Reactance= 0.4 },
new SmithChartData { Resistance= 0, Reactance= 0.05 }
};
}

```



Other than these positions, the legend can be placed anywhere in the Smith Chart. To achieve this, set the [Position](#) as **Custom** in the [SmithChartLegendSettings](#) and specify the X and Y coordinates using the [X](#) and [Y](#) properties in the [SmithChartLegendLocation](#).

ASPX-CS

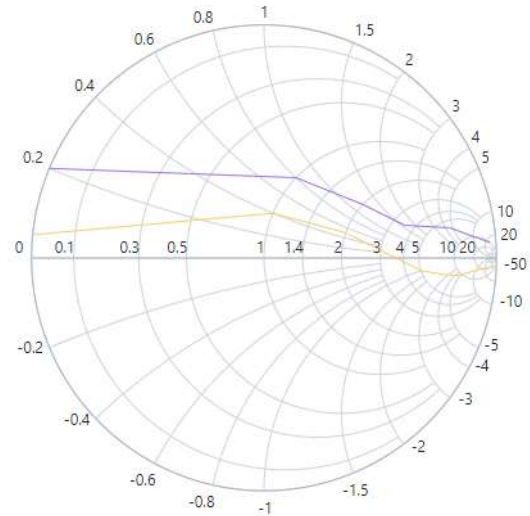
```
@using Syncfusion.Blazor.Charts
<SfSmithChart>
  <SmithChartLegendSettings Visible='true' Position='@LegendPosition.Custom'>
  <SmithChartLegendLocation X='80' Y='100'></SmithChartLegendLocation>
  </SmithChartLegendSettings>
  <SmithChartSeriesCollection>
  <SmithChartSeries Name="First Transmission"
  DataSource='FirstTransmissionData' Reactance="Reactance"
  Resistance="Resistance"></SmithChartSeries>
  <SmithChartSeries Name="Second Transmission"
  DataSource='SecondTransmissionData' Reactance="Reactance"
  Resistance="Resistance"></SmithChartSeries>
  </SmithChartSeriesCollection>
</SfSmithChart>

@code {
public class SmithChartData
{
public double? Resistance { get; set; }
public double? Reactance { get; set; }
};

public List<SmithChartData> FirstTransmissionData = new List<SmithChartData>
{
new SmithChartData { Resistance= 10, Reactance= 25 },
new SmithChartData { Resistance= 6, Reactance= 4.5 },
new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
new SmithChartData { Resistance= 2, Reactance= 1.2 },
new SmithChartData { Resistance= 1, Reactance= 0.8 },
new SmithChartData { Resistance= 0, Reactance= 0.2 }
};

public List<SmithChartData> SecondTransmissionData = new
List<SmithChartData> {
new SmithChartData { Resistance= 20, Reactance= -50 },
new SmithChartData { Resistance= 9, Reactance= -4.5 },
new SmithChartData { Resistance= 7, Reactance= -2.5 },
new SmithChartData { Resistance= 5, Reactance= -1 },
new SmithChartData { Resistance= 2, Reactance= 0.5 },
new SmithChartData { Resistance= 1, Reactance= 0.4 },
new SmithChartData { Resistance= 0, Reactance= 0.05 }
};
}
```

● First Transmission ● Second Transmission



Legend Alignment

Other than positioning the legend in the Smith Chart, its alignment also can be customized. By default, the legend is aligned in the **Center** position. Using the [Alignment](#) property, the legend can be aligned in the Smith Chart's near, centre, or far locations.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSmithChart>
  <SmithChartLegendSettings Visible='true'
    Position='@LegendPosition.Top'
    Alignment='@SmithChartAlignment.Near'>
  </SmithChartLegendSettings>
  <SmithChartSeriesCollection>
    <SmithChartSeries Name="First Transmission"
      DataSource='FirstTransmissionData' Reactance="Reactance"
      Resistance="Resistance"></SmithChartSeries>
    <SmithChartSeries Name="Second Transmission"
      DataSource='SecondTransmissionData' Reactance="Reactance"
      Resistance="Resistance"></SmithChartSeries>
  </SmithChartSeriesCollection>
</SfSmithChart>

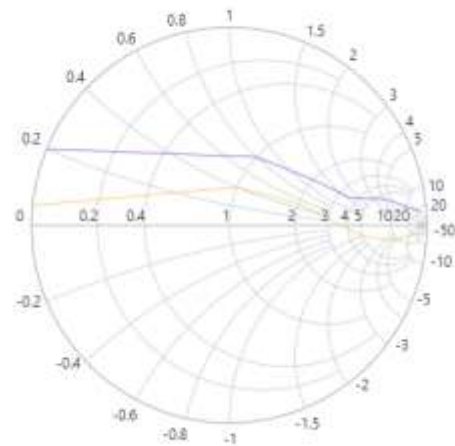
@code {
  public class SmithChartData
  {
    public double? Resistance { get; set; }
    public double? Reactance { get; set; }
  };
  public List<SmithChartData> FirstTransmissionData = new List<SmithChartData>
  {
    new SmithChartData { Resistance= 10, Reactance= 25 },
    new SmithChartData { Resistance= 6, Reactance= 4.5 },
    new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
    new SmithChartData { Resistance= 2, Reactance= 1.2 },
    new SmithChartData { Resistance= 1, Reactance= 0.8 },
  }
```

```

new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
public List<SmithChartData> SecondTransmissionData = new
List<SmithChartData> {
new SmithChartData { Resistance= 20, Reactance= -50 },
new SmithChartData { Resistance= 9, Reactance= -4.5 },
new SmithChartData { Resistance= 7, Reactance= -2.5 },
new SmithChartData { Resistance= 5, Reactance= -1 },
new SmithChartData { Resistance= 2, Reactance= 0.5 },
new SmithChartData { Resistance= 1, Reactance= 0.4 },
new SmithChartData { Resistance= 0, Reactance= 0.05 }
};
}

```

● First Transmission ● Second Transmission



Customization

Legend Shape

By default, the legend is rendered in **Circle** shape and the color of the shape is as same as the series color in the Smith Chart. Using the property [Shape](#) in the legend settings, the shape of the legend can be changed to rectangle, triangle, and so on.

ASPX-CS

```

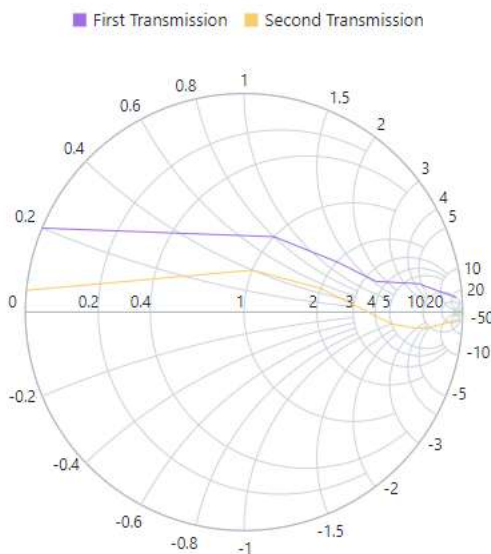
@using Syncfusion.Blazor.Charts
<SfSmithChart>
<SmithChartLegendSettings Visible='true'
Position='@LegendPosition.Top'
Shape='@Shape.Rectangle'>
</SmithChartLegendSettings>
<SmithChartSeriesCollection>
<SmithChartSeries Name="First Transmission"
DataSource='FirstTransmissionData' Reactance="Reactance"
Resistance="Resistance"></SmithChartSeries>
<SmithChartSeries Name="Second Transmission"
DataSource='SecondTransmissionData' Reactance="Reactance"
Resistance="Resistance"></SmithChartSeries>
</SmithChartSeriesCollection>

```

```

</SfSmithChart>
@code {
public class SmithChartData
{
public double? Resistance { get; set; }
public double? Reactance { get; set; }
};
public List<SmithChartData> FirstTransmissionData = new List<SmithChartData>
{
new SmithChartData { Resistance= 10, Reactance= 25 },
new SmithChartData { Resistance= 6, Reactance= 4.5 },
new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
new SmithChartData { Resistance= 2, Reactance= 1.2 },
new SmithChartData { Resistance= 1, Reactance= 0.8 },
new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
public List<SmithChartData> SecondTransmissionData = new
List<SmithChartData> {
new SmithChartData { Resistance= 20, Reactance= -50 },
new SmithChartData { Resistance= 9, Reactance= -4.5 },
new SmithChartData { Resistance= 7, Reactance= -2.5 },
new SmithChartData { Resistance= 5, Reactance= -1 },
new SmithChartData { Resistance= 2, Reactance= 0.5 },
new SmithChartData { Resistance= 1, Reactance= 0.4 },
new SmithChartData { Resistance= 0, Reactance= 0.05 }
};
}

```



Legend Size

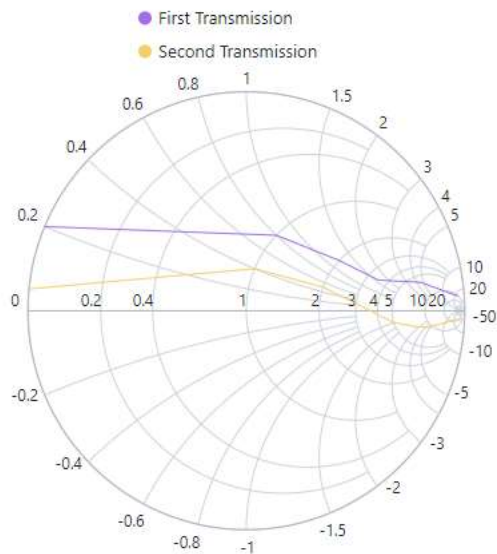
By default, the legend takes 20% - 25% of the Smith Chart's height horizontally when it is placed on the top or the bottom position, and 20% - 25% of the width vertically when it is placed on the left or the right position of the Chart. It can be changed by using the [Width](#) and the [Height](#) property of the legend settings. It can be in pixel or in percentage.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfSmithChart>
<SmithChartLegendSettings Visible='true' Position='@LegendPosition.Top'
Height='100px' Width='200px'>
</SmithChartLegendSettings>
<SmithChartSeriesCollection>
<SmithChartSeries Name="First Transmission"
DataSource='FirstTransmissionData' Reactance="Reactance"
Resistance="Resistance"></SmithChartSeries>
<SmithChartSeries Name="Second Transmission"
DataSource='SecondTransmissionData' Reactance="Reactance"
Resistance="Resistance"></SmithChartSeries>
</SmithChartSeriesCollection>
</SfSmithChart>
@code {
public class SmithChartData
{
public double? Resistance { get; set; }
public double? Reactance { get; set; }
};
public List<SmithChartData> FirstTransmissionData = new List<SmithChartData>
{
new SmithChartData { Resistance= 10, Reactance= 25 },
new SmithChartData { Resistance= 6, Reactance= 4.5 },
new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
new SmithChartData { Resistance= 2, Reactance= 1.2 },
new SmithChartData { Resistance= 1, Reactance= 0.8 },
new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
public List<SmithChartData> SecondTransmissionData = new
List<SmithChartData> {
new SmithChartData { Resistance= 20, Reactance= -50 },
new SmithChartData { Resistance= 9, Reactance= -4.5 },
new SmithChartData { Resistance= 7, Reactance= -2.5 },
new SmithChartData { Resistance= 5, Reactance= -1 },
new SmithChartData { Resistance= 2, Reactance= 0.5 },
new SmithChartData { Resistance= 1, Reactance= 0.4 },
new SmithChartData { Resistance= 0, Reactance= 0.05 }
};
}

```

Padding

The space between two legend items can be customized using the [ItemPadding](#) property and, the space between legend shape and text can be customized using the [ShapePadding](#) property.

ASPX-CS

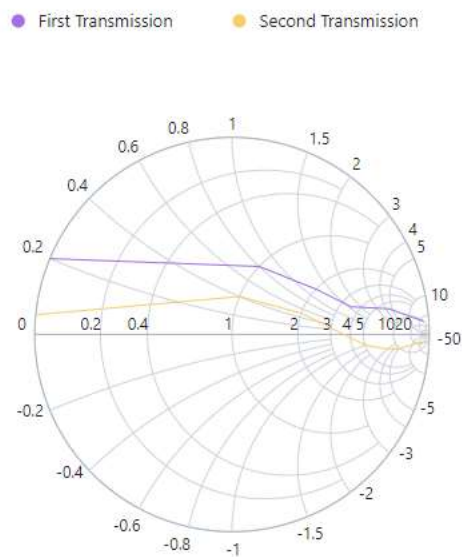
```
@using Syncfusion.Blazor.Charts
<SfSmithChart>
  <SmithChartLegendSettings Visible='true'
    Position='@LegendPosition.Top'
    ItemPadding='40'
    ShapePadding='10'>
  </SmithChartLegendSettings>
  <SmithChartSeriesCollection>
    <SmithChartSeries Name="First Transmission"
      DataSource='FirstTransmissionData' Reactance="Reactance"
      Resistance="Resistance"></SmithChartSeries>
    <SmithChartSeries Name="Second Transmission"
      DataSource='SecondTransmissionData' Reactance="Reactance"
      Resistance="Resistance"></SmithChartSeries>
  </SmithChartSeriesCollection>
</SfSmithChart>

@code {
  public class SmithChartData
  {
    public double? Resistance { get; set; }
    public double? Reactance { get; set; }
  };
  public List<SmithChartData> FirstTransmissionData = new List<SmithChartData>
  {
    new SmithChartData { Resistance= 10, Reactance= 25 },
    new SmithChartData { Resistance= 6, Reactance= 4.5 },
    new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
    new SmithChartData { Resistance= 2, Reactance= 1.2 },
    new SmithChartData { Resistance= 1, Reactance= 0.8 },
  }
}
```

```

new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
public List<SmithChartData> SecondTransmissionData = new
List<SmithChartData> {
new SmithChartData { Resistance= 20, Reactance= -50 },
new SmithChartData { Resistance= 9, Reactance= -4.5 },
new SmithChartData { Resistance= 7, Reactance= -2.5 },
new SmithChartData { Resistance= 5, Reactance= -1 },
new SmithChartData { Resistance= 2, Reactance= 0.5 },
new SmithChartData { Resistance= 1, Reactance= 0.4 },
new SmithChartData { Resistance= 0, Reactance= 0.05 }
};
}

```



Other customization

Each legend item's style, border, and text can be customized in the Smith Chart by using the following properties.

- [SmithChartLegendItemStyle](#) - Used to customize the height and the width of each legend item using the [Height](#) and the [Width](#) properties.
- [SmithChartLegendBorder](#) - Used to customize the border color and the width for legend collection using the [Color](#) and the [Width](#) properties.
- [SmithChartLegendTextStyle](#) - Used to customize the properties such as [FontFamily](#), [FontWeight](#), [FontStyle](#), [Opacity](#), [Color](#) and [Size](#) for each legend text.

ASPX-CS

```

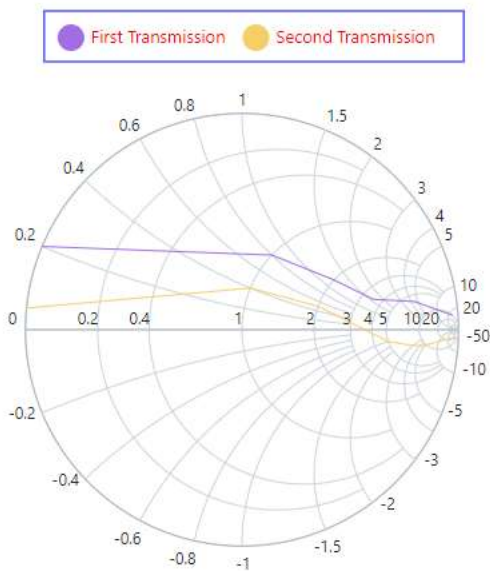
@using Syncfusion.Blazor.Charts
<SfSmithChart>
<SmithChartLegendSettings Visible='true' Position='@LegendPosition.Top'>
<SmithChartLegendTextStyle Color="red"></SmithChartLegendTextStyle>

```

```

<SmithChartLegendItemStyle Height="20"
Width="20"></SmithChartLegendItemStyle>
<SmithChartLegendBorder Color="blue" Width="1"></SmithChartLegendBorder>
</SmithChartLegendSettings>
<SmithChartSeriesCollection>
<SmithChartSeries Name="First Transmission"
DataSource='FirstTransmissionData' Reactance="Reactance"
Resistance="Resistance"></SmithChartSeries>
<SmithChartSeries Name="Second Transmission"
DataSource='SecondTransmissionData' Reactance="Reactance"
Resistance="Resistance"></SmithChartSeries>
</SmithChartSeriesCollection>
</SfSmithChart>
@code {
public class SmithChartData
{
public double? Resistance { get; set; }
public double? Reactance { get; set; }
};
public List<SmithChartData> FirstTransmissionData = new List<SmithChartData>
{
new SmithChartData { Resistance= 10, Reactance= 25 },
new SmithChartData { Resistance= 6, Reactance= 4.5 },
new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
new SmithChartData { Resistance= 2, Reactance= 1.2 },
new SmithChartData { Resistance= 1, Reactance= 0.8 },
new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
public List<SmithChartData> SecondTransmissionData = new
List<SmithChartData> {
new SmithChartData { Resistance= 20, Reactance= -50 },
new SmithChartData { Resistance= 9, Reactance= -4.5 },
new SmithChartData { Resistance= 7, Reactance= -2.5 },
new SmithChartData { Resistance= 5, Reactance= -1 },
new SmithChartData { Resistance= 2, Reactance= 0.5 },
new SmithChartData { Resistance= 1, Reactance= 0.4 },
new SmithChartData { Resistance= 0, Reactance= 0.05 }
};
}

```



Toggle Visibility

By default, the series name is displayed in the legend. The visibility of the series can be collapsed by clicking the legend of that particular series. The series visibility can be toggled by using the [ToggleVisibility](#) property. By default, it is **true**.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSmithChart>
  <SmithChartLegendSettings Visible='true'
    Position='@LegendPosition.Top'
    ToggleVisibility="true">
  </SmithChartLegendSettings>
  <SmithChartSeriesCollection>
    <SmithChartSeries Name="First Transmission"
      DataSource='FirstTransmissionData' Reactance="Reactance"
      Resistance="Resistance"></SmithChartSeries>
    <SmithChartSeries Name="Second Transmission"
      DataSource='SecondTransmissionData' Reactance="Reactance"
      Resistance="Resistance"></SmithChartSeries>
  </SmithChartSeriesCollection>
</SfSmithChart>

@code {
public class SmithChartData
{
public double? Resistance { get; set; }
public double? Reactance { get; set; }
};
public List<SmithChartData> FirstTransmissionData = new List<SmithChartData>
{
new SmithChartData { Resistance= 10, Reactance= 25 },
new SmithChartData { Resistance= 6, Reactance= 4.5 },
new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
new SmithChartData { Resistance= 2, Reactance= 1.2 },
}
```

```

new SmithChartData { Resistance= 1, Reactance= 0.8 },
new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
public List<SmithChartData> SecondTransmissionData = new
List<SmithChartData> {
new SmithChartData { Resistance= 20, Reactance= -50 },
new SmithChartData { Resistance= 9, Reactance= -4.5 },
new SmithChartData { Resistance= 7, Reactance= -2.5 },
new SmithChartData { Resistance= 5, Reactance= -1 },
new SmithChartData { Resistance= 2, Reactance= 0.5 },
new SmithChartData { Resistance= 1, Reactance= 0.4 },
new SmithChartData { Resistance= 0, Reactance= 0.05 }
};
}

```

Row and column

The legend can also be placed in rows and columns using the [RowCount](#) and the [ColumnCount](#) property. By default, their value is 0.

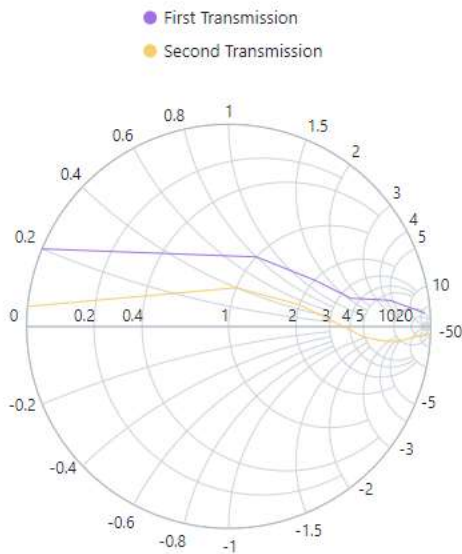
ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfSmithChart>
<SmithChartLegendSettings Visible='true' ColumnCount="1"
Position='@LegendPosition.Top'>
</SmithChartLegendSettings>
<SmithChartSeriesCollection>
<SmithChartSeries Name="First Transmission"
DataSource='FirstTransmissionData' Reactance="Reactance"
Resistance="Resistance"></SmithChartSeries>
<SmithChartSeries Name="Second Transmission"
DataSource='SecondTransmissionData' Reactance="Reactance"
Resistance="Resistance"></SmithChartSeries>
</SmithChartSeriesCollection>
</SfSmithChart>
@code {
public class SmithChartData
{
public double? Resistance { get; set; }
public double? Reactance { get; set; }
};
public List<SmithChartData> FirstTransmissionData = new List<SmithChartData>
{
new SmithChartData { Resistance= 10, Reactance= 25 },
new SmithChartData { Resistance= 6, Reactance= 4.5 },
new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
new SmithChartData { Resistance= 2, Reactance= 1.2 },
new SmithChartData { Resistance= 1, Reactance= 0.8 },
new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
public List<SmithChartData> SecondTransmissionData = new
List<SmithChartData> {
new SmithChartData { Resistance= 20, Reactance= -50 },
new SmithChartData { Resistance= 9, Reactance= -4.5 },
new SmithChartData { Resistance= 7, Reactance= -2.5 },
new SmithChartData { Resistance= 5, Reactance= -1 },

```

```
new SmithChartData { Resistance= 2, Reactance= 0.5 },
new SmithChartData { Resistance= 1, Reactance= 0.4 },
new SmithChartData { Resistance= 0, Reactance= 0.05 }
};
}
```



Title

The title depicts the information about the legend collection in the Smith Chart. It can be customized using the following properties in the [SmithChartLegendTitle](#).

- [Text](#) - Used to customize the legend title text.
- [Visible](#) - Used to specify the visibility of the legend title. By default, it is **true**.
- [TextAlignment](#) - Used to specify the legend title alignment.
- [SmithChartLegendTitleTextStyle](#) - Used to customize the properties such as [FontFamily](#), [FontWeight](#), [FontStyle](#), [Opacity](#), [Color](#) and [Size](#) for the title text.

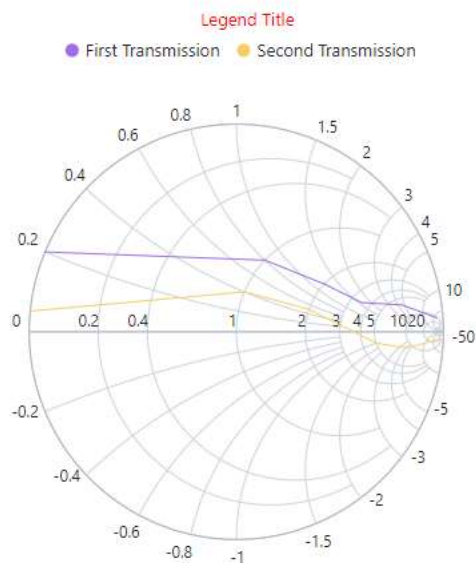
ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSmithChart>
  <SmithChartLegendSettings Visible='true' Position='@LegendPosition.Top'>
    <SmithChartLegendTitle Text="Legend Title"
      TextAlignment="@SmithChartAlignment.Center">
      <SmithChartLegendTitleTextStyle
        Color="red"></SmithChartLegendTitleTextStyle>
    </SmithChartLegendTitle>
  </SmithChartLegendSettings>
  <SmithChartSeriesCollection>
    <SmithChartSeries Name="First Transmission"
      DataSource='FirstTransmissionData' Reactance="Reactance"
      Resistance="Resistance"></SmithChartSeries>
  </SmithChartSeriesCollection>
</SfSmithChart>
```

```

<SmithChartSeries Name="Second Transmission"
DataSource='SecondTransmissionData' Reactance="Reactance"
Resistance="Resistance"></SmithChartSeries>
</SmithChartSeriesCollection>
</SfSmithChart>
@code {
public class SmithChartData
{
public double? Resistance { get; set; }
public double? Reactance { get; set; }
};
public List<SmithChartData> FirstTransmissionData = new List<SmithChartData>
{
new SmithChartData { Resistance= 10, Reactance= 25 },
new SmithChartData { Resistance= 6, Reactance= 4.5 },
new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
new SmithChartData { Resistance= 2, Reactance= 1.2 },
new SmithChartData { Resistance= 1, Reactance= 0.8 },
new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
public List<SmithChartData> SecondTransmissionData = new
List<SmithChartData> {
new SmithChartData { Resistance= 20, Reactance= -50 },
new SmithChartData { Resistance= 9, Reactance= -4.5 },
new SmithChartData { Resistance= 7, Reactance= -2.5 },
new SmithChartData { Resistance= 5, Reactance= -1 },
new SmithChartData { Resistance= 2, Reactance= 0.5 },
new SmithChartData { Resistance= 1, Reactance= 0.4 },
new SmithChartData { Resistance= 0, Reactance= 0.05 }
};
}

```

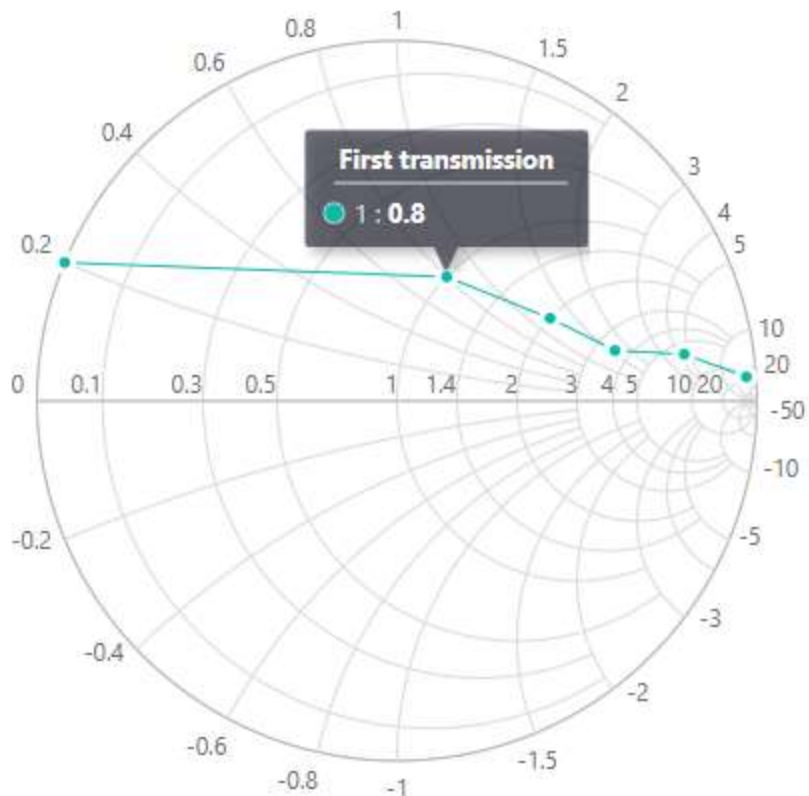


Tooltip in Blazor Smith Chart Component

When the mouse is moved over a point in the Smith Chart, a tooltip will appear displaying information about the point. By default, the tooltip is disabled. To enable the tooltip, set the [Visible](#) property to **true** in the [SmithChartSeriesTooltip](#).

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSmithChart>
<SmithChartSeriesCollection>
<SmithChartSeries Name="First transmission" DataSource='TransmissionData'
Reactance="Reactance" Resistance="Resistance">
<SmithChartSeriesMarker Visible='true'></SmithChartSeriesMarker>
<SmithChartSeriesTooltip Visible='true'></SmithChartSeriesTooltip>
</SmithChartSeries>
</SmithChartSeriesCollection>
</SfSmithChart>
@code {
public class SmithChartData
{
public double? Resistance { get; set; }
public double? Reactance { get; set; }
};
public List<SmithChartData> TransmissionData = new List<SmithChartData> {
new SmithChartData { Resistance= 10, Reactance= 25 },
new SmithChartData { Resistance= 6, Reactance= 4.5 },
new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
new SmithChartData { Resistance= 2, Reactance= 1.2 },
new SmithChartData { Resistance= 1, Reactance= 0.8 },
new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
}
```

Tooltip Customization

The tooltip can be customized for each series using the following properties.

- [Fill](#) - Used to change the fill color of the tooltip.
- [Opacity](#) - Used to control the opacity of the tooltip.
- [SmithChartSeriesTooltipBorder](#) - Used to customize the width and color of the border using the [Width](#) and the [Color](#) properties.

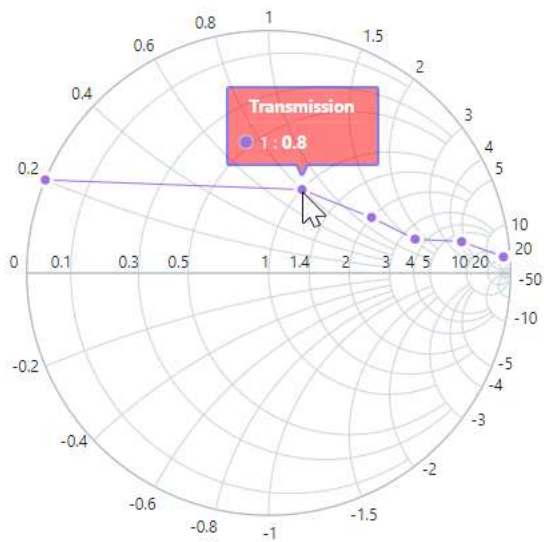
ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSmithChart>
<SmithChartSeriesCollection>
<SmithChartSeries Name="Transmission" DataSource='TransmissionData'
Reactance="Reactance" Resistance="Resistance">
<SmithChartSeriesMarker Visible='true'></SmithChartSeriesMarker>
<SmithChartSeriesTooltip Visible='true' Fill="red" Opacity="0.5">
<SmithChartSeriesTooltipBorder Color="blue"
Width="2"></SmithChartSeriesTooltipBorder>
</SmithChartSeriesTooltip>
</SmithChartSeries>
</SmithChartSeriesCollection>
</SfSmithChart>
@code {
public class SmithChartData
{
public double? Resistance { get; set; }
}
```

```

public double? Reactance { get; set; }
};
public List<SmithChartData> TransmissionData = new List<SmithChartData> {
new SmithChartData { Resistance= 10, Reactance= 25 },
new SmithChartData { Resistance= 6, Reactance= 4.5 },
new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
new SmithChartData { Resistance= 2, Reactance= 1.2 },
new SmithChartData { Resistance= 1, Reactance= 0.8 },
new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
}

```



Tooltip Template

To access the aggregate values inside the template, the implicit named parameter context can be used. The context can be typecast as the [SmithChartPoint](#) to get aggregate values inside the template. The tooltip template using the context is as follows.

ASPX-CS

```

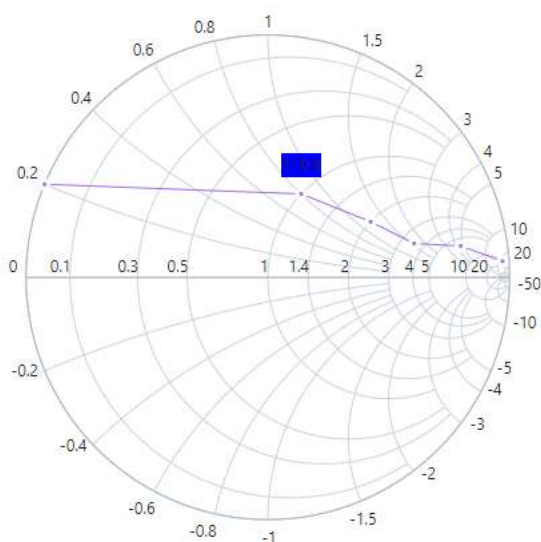
@using Syncfusion.Blazor.Charts
<SfSmithChart>
  <SmithChartSeriesCollection>
    <SmithChartSeries Name="Transmission" DataSource='TransmissionData'
      Reactance="Reactance" Resistance="Resistance">
      <SmithChartSeriesMarker Visible='true'></SmithChartSeriesMarker>
      <SmithChartSeriesTooltip Visible='true'>
        <Template>
          @{
            var data = context as SmithChartPoint;
          }
          <div style="background-color: blue">@data.Resistance: @data.Reactance</div>
        </Template>
      </SmithChartSeriesTooltip>
    </SmithChartSeries>
  </SmithChartSeriesCollection>

```

```

</SfSmithChart>
@code {
public class SmithChartData
{
public double? Resistance { get; set; }
public double? Reactance { get; set; }
};
public List<SmithChartData> TransmissionData = new List<SmithChartData> {
new SmithChartData { Resistance= 10, Reactance= 25 },
new SmithChartData { Resistance= 6, Reactance= 4.5 },
new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
new SmithChartData { Resistance= 2, Reactance= 1.2 },
new SmithChartData { Resistance= 1, Reactance= 0.8 },
new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
}

```



Marker and Data labels in Blazor Smith Chart Component

Markers and data labels are used to provide information about the data points in the series. Both the marker and the datalabel are disabled by default in the Smith Chart. Both can be enabled by making the [Visible](#) property in the marker and the datalabel settings to **true**.

Marker

By default, the visibility of the marker is **false**. It can be enabled by setting the [Visible](#) property to **true** in the [SmithChartSeriesMarker](#). This will add a marker for each data point in the series. Using marker setting, it can be customized differently for each series in the Smith Chart.

ASPX-CS

```

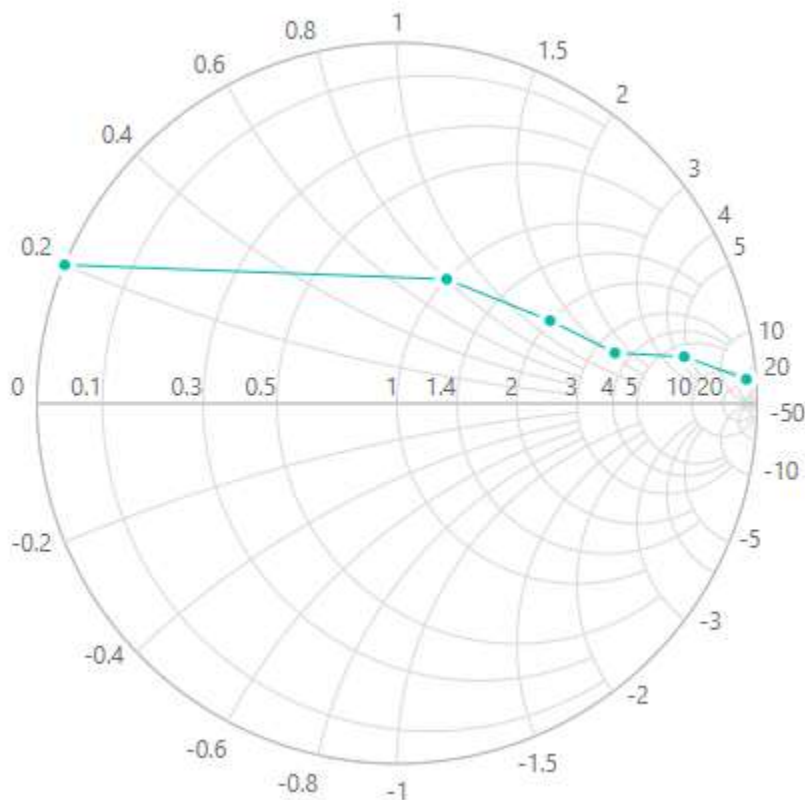
@using Syncfusion.Blazor.Charts
<SfSmithChart>
<SmithChartSeriesCollection>
<SmithChartSeries Name="Transmission" DataSource='TransmissionData'
Reactance="Reactance" Resistance="Resistance">

```

```

<SmithChartSeriesMarker Visible='true'></SmithChartSeriesMarker>
</SmithChartSeries>
</SmithChartSeriesCollection>
</SfSmithChart>
@code {
public class SmithChartData
{
public double? Resistance { get; set; }
public double? Reactance { get; set; }
};
public List<SmithChartData> TransmissionData = new List<SmithChartData> {
new SmithChartData { Resistance= 10, Reactance= 25 },
new SmithChartData { Resistance= 6, Reactance= 4.5 },
new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
new SmithChartData { Resistance= 2, Reactance= 1.2 },
new SmithChartData { Resistance= 1, Reactance= 0.8 },
new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
}

```



Marker Customization

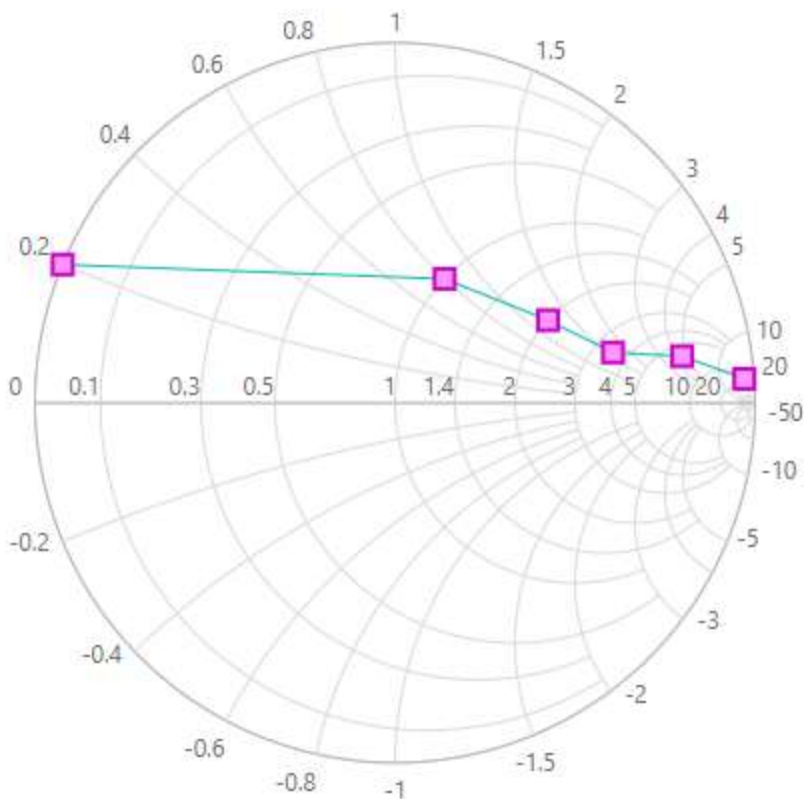
Using the [SmithChartSeriesMarker](#), the following marker properties can be customized differently for each series in the Smith Chart.

- [Width](#) - Used to customize the width of the marker.
- [Height](#) - Used to customize the height of the marker.

- [Fill](#) - Used to customize the fill color of the marker.
- [Opacity](#) - Used to customize the opacity of the marker.
- [SmithChartSeriesMarkerBorder](#) - Used to control the width and the color of the marker's border using the [Color](#) and the [Width](#) properties.
- [Shape](#) - Used to change the shape of the marker.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSmithChart>
<SmithChartSeriesCollection>
<SmithChartSeries Name="Transmission" DataSource='TransmissionData'
Reactance="Reactance" Resistance="Resistance">
<SmithChartSeriesMarker Visible='true'
Height='10'
Width='10'
Fill="#ff99ff"
Opacity='1'
Shape='@Shape.Rectangle'>
<SmithChartSeriesMarkerBorder Width='2' Color="#cc00cc">
</SmithChartSeriesMarkerBorder>
</SmithChartSeriesMarker>
</SmithChartSeries>
</SmithChartSeriesCollection>
</SfSmithChart>
@code {
public class SmithChartData
{
public double? Resistance { get; set; }
public double? Reactance { get; set; }
};
public List<SmithChartData> TransmissionData = new List<SmithChartData> {
new SmithChartData { Resistance= 10, Reactance= 25 },
new SmithChartData { Resistance= 6, Reactance= 4.5 },
new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
new SmithChartData { Resistance= 2, Reactance= 1.2 },
new SmithChartData { Resistance= 1, Reactance= 0.8 },
new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
}
```



Data labels

By default, the data labels are disabled. It can be enabled by setting the [Visible](#) property to **true** in the [SmithChartSeriesDatalabel](#). For each point in the series, a datalabel is created.

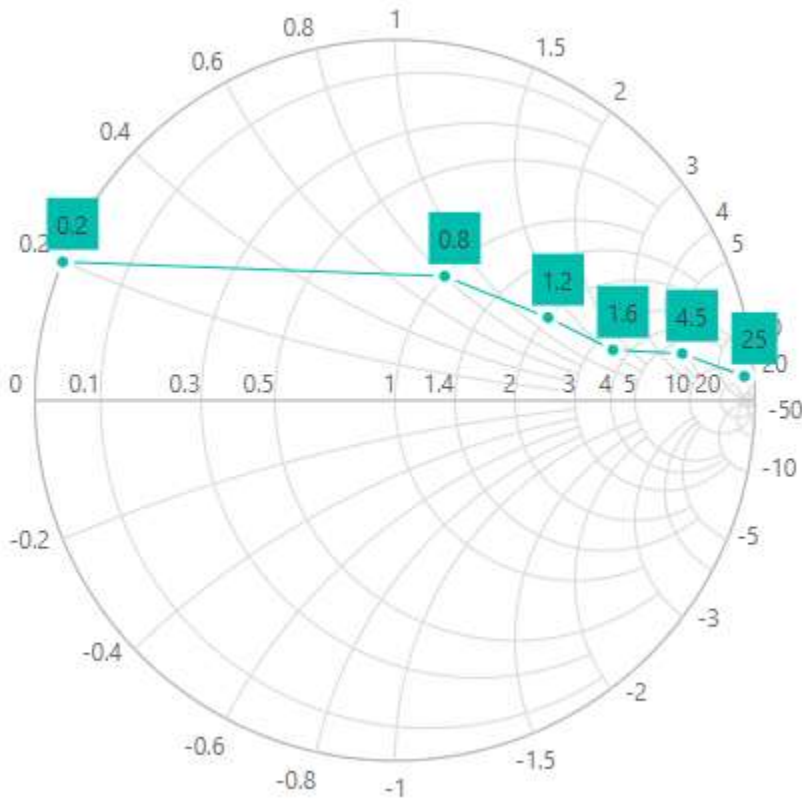
ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSmithChart>
  <SmithChartSeriesCollection>
    <SmithChartSeries Name="Transmission" DataSource='TransmissionData'
      Reactance="Reactance" Resistance="Resistance">
      <SmithChartSeriesMarker>
        <SmithChartSeriesDatalabel Visible='true'></SmithChartSeriesDatalabel>
      </SmithChartSeriesMarker>
    </SmithChartSeries>
  </SmithChartSeriesCollection>
</SfSmithChart>

@code {
  public class SmithChartData
  {
    public double? Resistance { get; set; }
    public double? Reactance { get; set; }
  };

  public List<SmithChartData> TransmissionData = new List<SmithChartData> {
    new SmithChartData { Resistance= 10, Reactance= 25 },
    new SmithChartData { Resistance= 6, Reactance= 4.5 },
    new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
    new SmithChartData { Resistance= 2, Reactance= 1.2 },
  }
```

```
new SmithChartData { Resistance= 1, Reactance= 0.8 },
new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
}
```



Data labels customization

The data labels can be customized using the following properties.

- [Fill](#) - Used to change the fill color of the data labels.
- [Opacity](#) - Used to control the opacity of the data labels.
- [SmithChartSeriesDataLabelBorder](#) - Used to customize width and color of the border using the [Width](#) and the [Color](#) properties.
- [SmithChartDataLabelTextStyle](#) - Used to customize properties such as [FontFamily](#), [FontWeight](#), [FontStyle](#), [Opacity](#), [Color](#), and [Size](#) for datalabel font.

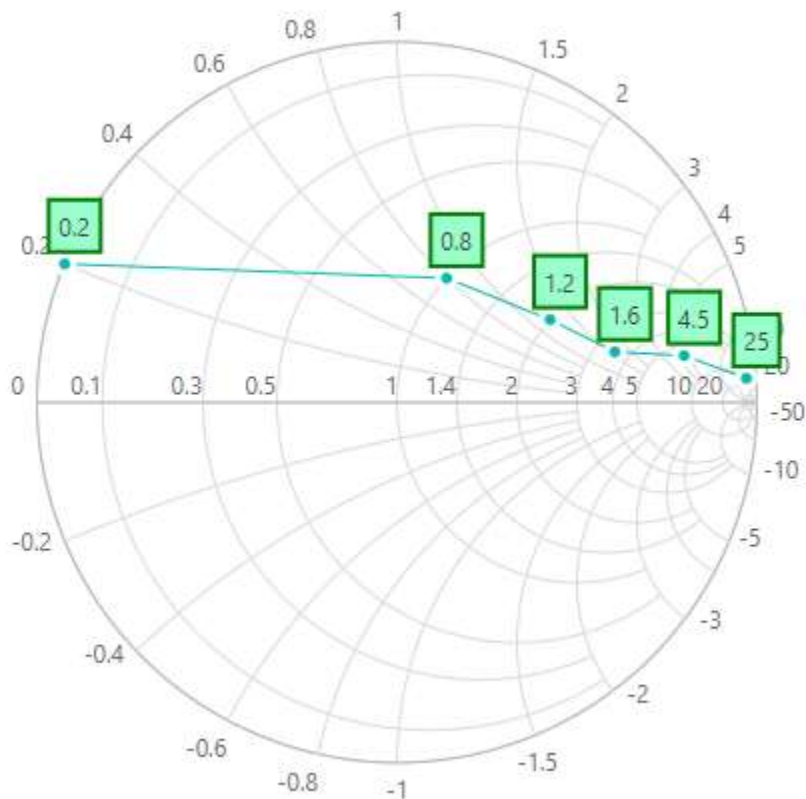
ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSmithChart>
<SmithChartSeriesCollection>
<SmithChartSeries Name="Transmission" DataSource='TransmissionData'
Reactance="Reactance" Resistance="Resistance">
<SmithChartSeriesMarker Visible='true'>
<SmithChartSeriesDataLabel Visible='true' Fill="#99ffcc" Opacity='1'>
<SmithChartDataLabelTextStyle Color="red"
Size="15px"></SmithChartDataLabelTextStyle>
```

```

<SmithChartSeriesDataLabelBorder Color="green" Width='2'>
</SmithChartSeriesDataLabelBorder>
</SmithChartSeriesDataLabel>
</SmithChartSeriesMarker>
</SmithChartSeries>
</SmithChartSeriesCollection>
</SfSmithChart>
@code {
public class SmithChartData
{
public double? Resistance { get; set; }
public double? Reactance { get; set; }
};
private List<SmithChartData> TransmissionData = new List<SmithChartData> {
new SmithChartData { Resistance= 10, Reactance= 25 },
new SmithChartData { Resistance= 6, Reactance= 4.5 },
new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
new SmithChartData { Resistance= 2, Reactance= 1.2 },
new SmithChartData { Resistance= 1, Reactance= 0.8 },
new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
}

```



Smart Labels

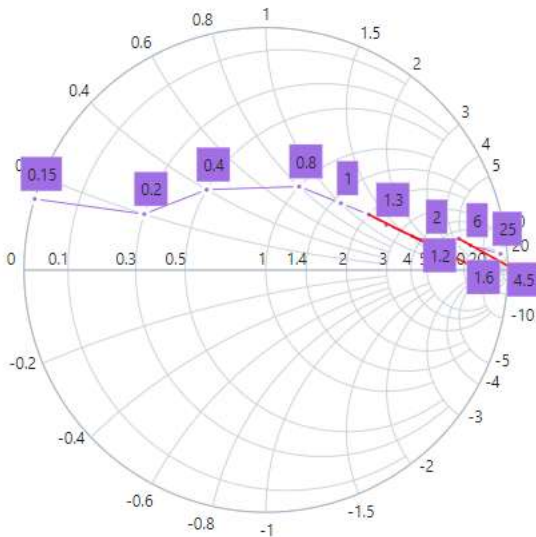
Data labels can be placed smartly by setting the [EnableSmartLabels](#) to **true** in the Smith Chart series. A line will be connected for smartly aligned labels. It's color and width can be customized using the [Color](#) and the [Width](#) properties in the [SmithChartDataLabelConnectorLine](#).

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSmithChart>
  <SmithChartSeriesCollection>
    <SmithChartSeries EnableSmartLabels="true" Name="Transmission"
      DataSource='TransmissionData' Reactance="Reactance" Resistance="Resistance">
      <SmithChartSeriesMarker Visible='true'>
        <SmithChartSeriesDatalabel Visible='true'>
          <SmithChartDataLabelConnectorLine Color="red"
            Width="1.5"></SmithChartDataLabelConnectorLine>
        </SmithChartSeriesDatalabel>
      </SmithChartSeriesMarker>
    </SmithChartSeries>
  </SmithChartSeriesCollection>
</SfSmithChart>

@code {
public class SmithChartData
{
public double? Resistance { get; set; }
public double? Reactance { get; set; }
};

public List<SmithChartData> TransmissionData = new List<SmithChartData> {
new SmithChartData { Resistance= 10, Reactance= 25 }, new SmithChartData {
Resistance= 8, Reactance= 6 },
new SmithChartData { Resistance= 6, Reactance= 4.5 }, new SmithChartData {
Resistance= 4.5, Reactance= 2 },
new SmithChartData { Resistance= 3.5, Reactance= 1.6 }, new SmithChartData {
Resistance= 2.5, Reactance= 1.3 },
new SmithChartData { Resistance= 2, Reactance= 1.2 }, new SmithChartData {
Resistance= 1.5, Reactance= 1 },
new SmithChartData { Resistance= 1, Reactance= 0.8 }, new SmithChartData {
Resistance= 0.5, Reactance= 0.4 },
new SmithChartData { Resistance= 0.3, Reactance= 0.2 }, new SmithChartData {
Resistance= 0, Reactance= 0.15 },
};
}
```



Datalabel Template

To access the aggregate values inside the template, the implicit named parameter context can be used. The context can be typecast as [SmithChartPoint](#) to get aggregate values inside the template. The datalabel template using the context is shown as following.

ASPX-CS

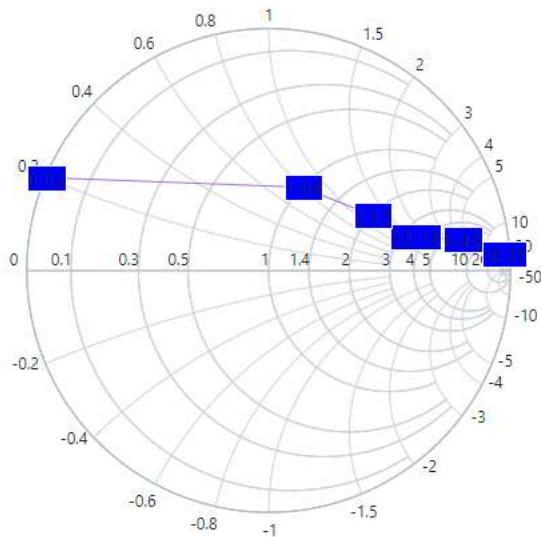
```
@using Syncfusion.Blazor.Charts
<SfSmithChart>
  <SmithChartSeriesCollection>
    <SmithChartSeries Name="Transmission" DataSource='TransmissionData'
      Reactance="Reactance" Resistance="Resistance">
      <SmithChartSeriesMarker Visible='true'>
      <SmithChartSeriesDatalabel Visible='true'>
      <Template>
        @{
          var data = context as SmithChartPoint;
        }
        <div style="background-color: blue">@data.Resistance: @data.Reactance</div>
      </Template>
    </SmithChartSeriesDatalabel>
    </SmithChartSeriesMarker>
    </SmithChartSeries>
  </SmithChartSeriesCollection>
</SfSmithChart>

@code {
  public class SmithChartData
  {
    public double? Resistance { get; set; }
    public double? Reactance { get; set; }
  };
  public List<SmithChartData> TransmissionData = new List<SmithChartData> {
    new SmithChartData { Resistance= 10, Reactance= 25 },
    new SmithChartData { Resistance= 6, Reactance= 4.5 },
    new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
```

```

new SmithChartData { Resistance= 2, Reactance= 1.2 },
new SmithChartData { Resistance= 1, Reactance= 0.8 },
new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
}

```



Series in Blazor Smith Chart Component

The [SmithChartSeries](#) is the visual representation of the data. Using the following options in the [SmithChartSeries](#), each series can be customized in the Smith Chart.

- [Fill](#) - Used to customize the fill color for the series.
- [Visible](#) - Used to handle the visibility of the series.
- [Opacity](#) - Used to control the opacity of the series line.
- [Width](#) - Used to customize the width of the series line.

ASPX-CS

```

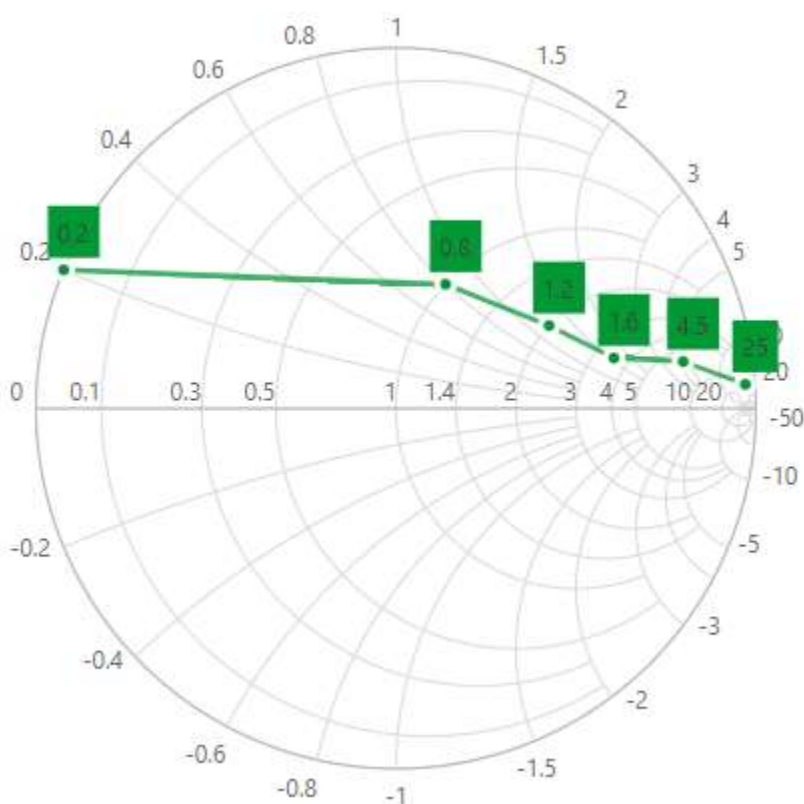
@using Syncfusion.Blazor.Charts
<SfSmithChart>
  <SmithChartSeriesCollection>
    <SmithChartSeries Name="Transmission" DataSource='TransmissionData'
      Reactance="Reactance" Resistance="Resistance" Fill="#009933"
      Visible="true"
      Opacity="0.75"
      Width="2.5">
      <SmithChartSeriesMarker Visible='true'>
        <SmithChartSeriesDatalabel Visible='true'></SmithChartSeriesDatalabel>
      </SmithChartSeriesMarker>
    </SmithChartSeries>
  </SmithChartSeriesCollection>
</SfSmithChart>
@code {

```

```

public class SmithChartData
{
    public double? Resistance { get; set; }
    public double? Reactance { get; set; }
};
public List<SmithChartData> TransmissionData = new List<SmithChartData> {
    new SmithChartData { Resistance= 10, Reactance= 25 },
    new SmithChartData { Resistance= 6, Reactance= 4.5 },
    new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
    new SmithChartData { Resistance= 2, Reactance= 1.2 },
    new SmithChartData { Resistance= 1, Reactance= 0.8 },
    new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
}

```



Animation

Animation for the Smith Chart series can be enabled by using the [EnableAnimation](#) property in the [SmithChartSeries](#). By default, the value is **false**. The speed of the animation can be controlled using the [AnimationDuration](#) property. By default, the value of the [AnimationDuration](#) property is **2000** milliseconds.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfSmithChart>
    <SmithChartSeriesCollection>
        <SmithChartSeries Name="Transmission" DataSource='TransmissionData'

```

```

Reactance="Reactance" Resistance="Resistance" EnableAnimation="true"
AnimationDuration="1500">
<SmithChartSeriesMarker Visible='true'>
<SmithChartSeriesDatalabel Visible='true'></SmithChartSeriesDatalabel>
</SmithChartSeriesMarker>
</SmithChartSeries>
</SmithChartSeriesCollection>
</SfSmithChart>
@code {
public class SmithChartData
{
public double? Resistance { get; set; }
public double? Reactance { get; set; }
};
public List<SmithChartData> TransmissionData = new List<SmithChartData> {
new SmithChartData { Resistance= 10, Reactance= 25 },
new SmithChartData { Resistance= 6, Reactance= 4.5 },
new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
new SmithChartData { Resistance= 2, Reactance= 1.2 },
new SmithChartData { Resistance= 1, Reactance= 0.8 },
new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
}

```

Print and Export in Blazor Smith Chart Component

Print

The rendered Smith Chart can be printed directly from the browser by calling the public method `PrintAsync`.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
<button id="print" @onclick="Print">Print</button>
<SfSmithChart @ref="smithChart">
<SmithChartSeriesCollection>
<SmithChartSeries Name="Transmission" DataSource='TransmissionData'
Reactance="Reactance" Resistance="Resistance">
<SmithChartSeriesMarker Visible='true'>
<SmithChartSeriesDatalabel Visible='true'></SmithChartSeriesDatalabel>
</SmithChartSeriesMarker>
</SmithChartSeries>
</SmithChartSeriesCollection>
</SfSmithChart>
@code {
private SfSmithChart smithChart;
public class SmithChartData
{
public double? Resistance { get; set; }
public double? Reactance { get; set; }
};
public List<SmithChartData> TransmissionData = new List<SmithChartData> {
new SmithChartData { Resistance= 10, Reactance= 25 },
new SmithChartData { Resistance= 6, Reactance= 4.5 },
new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
new SmithChartData { Resistance= 2, Reactance= 1.2 },

```

```

new SmithChartData { Resistance= 1, Reactance= 0.8 },
new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
private async Task Print()
{
    await smithChart.PrintAsync();
}
}

```

Export

The rendered Smith Chart can be exported to **JPEG**, **PNG**, **SVG**, or **PDF** format by using the **ExportAsync** method in the Smith Chart. This method contains the following parameters:

- **Type** - To specify the export type. The component can be exported to **JPEG**, **PNG**, **SVG**, or **PDF** format.
- **File name** - To specify the file name to export.
- **Orientation** - To specify the orientation type. This is applicable only for PDF export type. It is an optional parameter.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
<button id="export" @onclick="Export">Export</button>
<SfSmithChart @ref="smithChart">
    <SmithChartSeriesCollection>
        <SmithChartSeries Name="Transmission" DataSource='TransmissionData'
            Reactance="Reactance" Resistance="Resistance">
            <SmithChartSeriesMarker Visible='true'>
            <SmithChartSeriesDatalabel Visible='true'></SmithChartSeriesDatalabel>
            </SmithChartSeriesMarker>
        </SmithChartSeries>
    </SmithChartSeriesCollection>
</SfSmithChart>
@code {
    private SfSmithChart smithChart;
    public class SmithChartData
    {
        public double? Resistance { get; set; }
        public double? Reactance { get; set; }
    };
    public List<SmithChartData> TransmissionData = new List<SmithChartData> {
        new SmithChartData { Resistance= 10, Reactance= 25 },
        new SmithChartData { Resistance= 6, Reactance= 4.5 },
        new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
        new SmithChartData { Resistance= 2, Reactance= 1.2 },
        new SmithChartData { Resistance= 1, Reactance= 0.8 },
        new SmithChartData { Resistance= 0, Reactance= 0.2 }
    };
    private async Task Export()
    {
        await smithChart.ExportAsync(ExportType.PDF, "SmithChart",
            Syncfusion.PdfExport.PdfPageOrientation.Landscape);
    }
}

```

Events in Blazor Smith Chart Component

This section describes about the Smith Chart component's events that will be triggered when appropriate actions are performed. The events should be provided to the Smith Chart through the **SmithChartEvents** component.

The Smith Chart component supports the following events.

- [Loaded](#)
- [OnPrintComplete](#)
- [OnExportComplete](#)
- [AxisLabelRendering](#)
- [LegendRendering](#)
- [SeriesRender](#)
- [TitleRendering](#)
- [SubtitleRendering](#)
- [TextRendering](#)
- [SizeChanged](#)

Loaded

The [Loaded](#) event triggers after the Smith Chart is rendered.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSmithChart>
<SmithChartEvents Loaded="SmithChartLoaded"></SmithChartEvents>
<SmithChartSeriesCollection>
<SmithChartSeries DataSource='FirstTransmissionData'
Reactance="Reactance" Resistance="Resistance">
</SmithChartSeries>
</SmithChartSeriesCollection>
</SfSmithChart>
@code {
public class SmithChartData
{
public double? Resistance { get; set; }
public double? Reactance { get; set; }
};
public List<SmithChartData> FirstTransmissionData = new List<SmithChartData>
{
new SmithChartData { Resistance= 10, Reactance= 25 },
new SmithChartData { Resistance= 6, Reactance= 4.5 },
new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
new SmithChartData { Resistance= 2, Reactance= 1.2 },
new SmithChartData { Resistance= 1, Reactance= 0.8 },
new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
public void SmithChartLoaded(SmithChartLoadedEventArgs args)
{
// Here you can customize your code.
}
}
```

OnPrintComplete

The `OnPrintComplete` event triggers after the Smith Chart is printed.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
<button id="print" @onclick="Print">Print</button>
<SfSmithChart @ref="SmithChart">
  <SmithChartEvents OnPrintComplete="PrintCompleted"></SmithChartEvents>
  <SmithChartSeriesCollection>
    <SmithChartSeries DataSource='FirstTransmissionData'
      Reactance="Reactance" Resistance="Resistance">
    </SmithChartSeries>
  </SmithChartSeriesCollection>
</SfSmithChart>
@code {
public SfSmithChart SmithChart;
public class SmithChartData
{
public double? Resistance { get; set; }
public double? Reactance { get; set; }
};
public List<SmithChartData> FirstTransmissionData = new List<SmithChartData>
{
new SmithChartData { Resistance= 10, Reactance= 25 },
new SmithChartData { Resistance= 6, Reactance= 4.5 },
new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
new SmithChartData { Resistance= 2, Reactance= 1.2 },
new SmithChartData { Resistance= 1, Reactance= 0.8 },
new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
private async Task Print()
{
await SmithChart.PrintAsync();
}
public void PrintCompleted()
{
// Here you can customize your code.
}
}

```

OnExportComplete

The `OnExportComplete` event triggers after the Smith Chart is exported.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
<button id="export" @onclick="Export">Export</button>
<SfSmithChart @ref="SmithChart">
  <SmithChartEvents OnExportComplete="ExportCompleted"></SmithChartEvents>
  <SmithChartSeriesCollection>
    <SmithChartSeries Name="Transmission" DataSource='TransmissionData'
      Reactance="Reactance" Resistance="Resistance">
    <SmithChartSeriesMarker Visible='true'>
    <SmithChartSeriesDatalabel Visible='true'></SmithChartSeriesDatalabel>
  </SmithChartSeriesCollection>
</SfSmithChart>

```



```

</SmithChartSeriesMarker>
</SmithChartSeries>
</SmithChartSeriesCollection>
</SfSmithChart>
@code {
public SfSmithChart SmithChart;
public class SmithChartData
{
public double? Resistance { get; set; }
public double? Reactance { get; set; }
};
public List<SmithChartData> TransmissionData = new List<SmithChartData> {
new SmithChartData { Resistance= 10, Reactance= 25 },
new SmithChartData { Resistance= 6, Reactance= 4.5 },
new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
new SmithChartData { Resistance= 2, Reactance= 1.2 },
new SmithChartData { Resistance= 1, Reactance= 0.8 },
new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
private async Task Export()
{
await SmithChart.ExportAsync(ExportType.PNG, "SmithChart");
}
public void ExportCompleted(SmithChartExportEventArgs args)
{
// Here you can customize your code.
}
}

```

AxisLabelRendering

Before rendering each axis label, the [AxisLabelRendering](#) event is triggered. The following arguments are present in this event:

- **Text** - Specifies the current axis label text.
- **X** - Specifies the current axis label X position.
- **Y** - Specifies the current axis label Y position.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfSmithChart>
<SmithChartEvents
AxisLabelRendering="AxisLabelCustomization"></SmithChartEvents>
<SmithChartSeriesCollection>
<SmithChartSeries DataSource='FirstTransmissionData'
Reactance="Reactance" Resistance="Resistance">
</SmithChartSeries>
</SmithChartSeriesCollection>
</SfSmithChart>
@code {
public class SmithChartData
{
public double? Resistance { get; set; }
public double? Reactance { get; set; }
}

```

```

};
public List<SmithChartData> FirstTransmissionData = new List<SmithChartData>
{
    new SmithChartData { Resistance= 10, Reactance= 25 },
    new SmithChartData { Resistance= 6, Reactance= 4.5 },
    new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
    new SmithChartData { Resistance= 2, Reactance= 1.2 },
    new SmithChartData { Resistance= 1, Reactance= 0.8 },
    new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
public void AxisLabelCustomization(SmithChartAxisLabelRenderEventArgs args)
{
    // Here you can customize your code
}
}

```

LegendRendering

Before rendering each legend, the [LegendRendering](#) event is triggered. The following arguments are present in this event:

- **Text** - Specifies the current legend text.
- **Shape** - Customize the shape of the legend.
- **Fill** - Specifies the legend shape color.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfSmithChart>
<SmithChartEvents LegendRendering="LegendCustomization"></SmithChartEvents>
<SmithChartLegendSettings Visible="true"></SmithChartLegendSettings>
<SmithChartSeriesCollection>
<SmithChartSeries Name="Transmission" DataSource='FirstTransmissionData'
Reactance="Reactance" Resistance="Resistance">
</SmithChartSeries>
</SmithChartSeriesCollection>
</SfSmithChart>
@code {
public class SmithChartData
{
    public double? Resistance { get; set; }
    public double? Reactance { get; set; }
};
public List<SmithChartData> FirstTransmissionData = new List<SmithChartData>
{
    new SmithChartData { Resistance= 10, Reactance= 25 },
    new SmithChartData { Resistance= 6, Reactance= 4.5 },
    new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
    new SmithChartData { Resistance= 2, Reactance= 1.2 },
    new SmithChartData { Resistance= 1, Reactance= 0.8 },
    new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
public void LegendCustomization(SmithChartLegendRenderEventArgs args)
{
    // Here you can customize your code.
}
}

```

```
}
}
```

SeriesRender

Before rendering each series, the [SeriesRender](#) event is triggered. The following arguments are present in this event:

- **Text** - Specifies the current series text.
- **Index** - Specifies the current series index.
- **Fill** - Specifies the current series color.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSmithChart>
  <SmithChartEvents SeriesRender="SeriesCustomization"></SmithChartEvents>
  <SmithChartSeriesCollection>
    <SmithChartSeries DataSource='FirstTransmissionData'
      Reactance="Reactance" Resistance="Resistance">
    </SmithChartSeries>
  </SmithChartSeriesCollection>
</SfSmithChart>

@code {
  public class SmithChartData
  {
    public double? Resistance { get; set; }
    public double? Reactance { get; set; }
  };
  public List<SmithChartData> FirstTransmissionData = new List<SmithChartData>
  {
    new SmithChartData { Resistance= 10, Reactance= 25 },
    new SmithChartData { Resistance= 6, Reactance= 4.5 },
    new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
    new SmithChartData { Resistance= 2, Reactance= 1.2 },
    new SmithChartData { Resistance= 1, Reactance= 0.8 },
    new SmithChartData { Resistance= 0, Reactance= 0.2 }
  };
  public void SeriesCustomization(SmithChartSeriesRenderEventArgs args)
  {
    // Here you can customize your code.
  }
}
```

TitleRendering

The [TitleRendering](#) event triggers before the title is rendered. The following arguments are present in this event:

- **Text** - Specifies the current title text.
- **X** - Specifies the current title X position.
- **Y** - Specifies the current title Y position.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfSmithChart>
<SmithChartTitle Text="Smith Chart Title">
</SmithChartTitle>
<SmithChartEvents TitleRendering="TitleCustomization"></SmithChartEvents>
<SmithChartSeriesCollection>
<SmithChartSeries DataSource='FirstTransmissionData'
Reactance="Reactance" Resistance="Resistance">
</SmithChartSeries>
</SmithChartSeriesCollection>
</SfSmithChart>
@code {
public class SmithChartData
{
public double? Resistance { get; set; }
public double? Reactance { get; set; }
};
public List<SmithChartData> FirstTransmissionData = new List<SmithChartData>
{
new SmithChartData { Resistance= 10, Reactance= 25 },
new SmithChartData { Resistance= 6, Reactance= 4.5 },
new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
new SmithChartData { Resistance= 2, Reactance= 1.2 },
new SmithChartData { Resistance= 1, Reactance= 0.8 },
new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
public void TitleCustomization(TitleRenderEventArgs args)
{
// Here you can customize your code.
}
}

```

SubtitleRendering

The [SubtitleRendering](#) event triggers before the subtitle is rendered. The following arguments are present in this event:

- **Text** - Specifies the current subtitle text.
- **X** - Specifies the current subtitle X position.
- **Y** - Specifies the current subtitle Y position.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfSmithChart>
<SmithChartTitle Text="Smith Chart Title">
<SmithChartSubtitle Text="Smith Chart Subtitle"></SmithChartSubtitle>
</SmithChartTitle>
<SmithChartEvents
SubtitleRendering="SubtitleCustomization"></SmithChartEvents>
<SmithChartSeriesCollection>
<SmithChartSeries DataSource='FirstTransmissionData'
Reactance="Reactance" Resistance="Resistance">

```

```

</SmithChartSeries>
</SmithChartSeriesCollection>
</SfSmithChart>
@code {
public class SmithChartData
{
public double? Resistance { get; set; }
public double? Reactance { get; set; }
};
public List<SmithChartData> FirstTransmissionData = new List<SmithChartData>
{
new SmithChartData { Resistance= 10, Reactance= 25 },
new SmithChartData { Resistance= 6, Reactance= 4.5 },
new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
new SmithChartData { Resistance= 2, Reactance= 1.2 },
new SmithChartData { Resistance= 1, Reactance= 0.8 },
new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
public void SubtitleCustomization(SubTitleRenderEventArgs args)
{
// Here you can customize your code.
}
}

```

TextRendering

The [TextRendering](#) event triggers before the datalabel text is rendered. The following arguments are present in this event:

- **Text** - Specifies the current text of the label.
- **X** - Specifies the current datalabel X position.
- **Y** - Specifies the current datalabel Y position.
- **PointIndex** - Specifies the current point index of the datalabel.
- **SeriesIndex** - Specifies the current series index of the datalabel.
- **Border** - Specifies the current datalabel border.
- **Color** - Specifies the current datalabel color.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfSmithChart>
<SmithChartEvents TextRendering="DataLabelCustomization"></SmithChartEvents>
<SmithChartSeriesCollection>
<SmithChartSeries DataSource='FirstTransmissionData'
Reactance="Reactance" Resistance="Resistance">
<SmithChartSeriesMarker>
<SmithChartSeriesDatalabel Visible="true"></SmithChartSeriesDatalabel>
</SmithChartSeriesMarker>
</SmithChartSeries>
</SmithChartSeriesCollection>
</SfSmithChart>
@code {
public class SmithChartData
{

```

```

public double? Resistance { get; set; }
public double? Reactance { get; set; }
};
public List<SmithChartData> FirstTransmissionData = new List<SmithChartData>
{
    new SmithChartData { Resistance= 10, Reactance= 25 },
    new SmithChartData { Resistance= 6, Reactance= 4.5 },
    new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
    new SmithChartData { Resistance= 2, Reactance= 1.2 },
    new SmithChartData { Resistance= 1, Reactance= 0.8 },
    new SmithChartData { Resistance= 0, Reactance= 0.2 }
};
public void DataLabelCustomization(SmithChartTextRenderEventArgs args)
{
    // Here you can customize your code.
}
}

```

SizeChanged

The [SizeChanged](#) event triggers when the browser window is resized. The following arguments are present in this event:

- **CurrentSize** - Specifies the current size of the Chart.
- **PreviousSize** - Specifies the previous size of the Chart.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfSmithChart>
    <SmithChartEvents SizeChanged="Resized"></SmithChartEvents>
    <SmithChartSeriesCollection>
        <SmithChartSeries DataSource='FirstTransmissionData'
            Reactance="Reactance" Resistance="Resistance">
        </SmithChartSeries>
    </SmithChartSeriesCollection>
</SfSmithChart>
@code {
    public class SmithChartData
    {
        public double? Resistance { get; set; }
        public double? Reactance { get; set; }
    };
    public List<SmithChartData> FirstTransmissionData = new List<SmithChartData>
    {
        new SmithChartData { Resistance= 10, Reactance= 25 },
        new SmithChartData { Resistance= 6, Reactance= 4.5 },
        new SmithChartData { Resistance= 3.5, Reactance= 1.6 },
        new SmithChartData { Resistance= 2, Reactance= 1.2 },
        new SmithChartData { Resistance= 1, Reactance= 0.8 },
        new SmithChartData { Resistance= 0, Reactance= 0.2 }
    };
    public void Resized(SmithChartResizeEventArgs args)
    {
        // Here you can customize your code.
    }
}

```

```
}
}
```

Sparkline

Getting Started with Blazor Sparkline Component

This section briefly explains how to include a Sparkline component in the Blazor server-side application. Refer to [Getting Started with Syncfusion Blazor for server-side in Visual Studio](#) page for introduction and configuring common specifications.

Importing Syncfusion Blazor Sparkline component in the application

1. Install **Syncfusion.Blazor** NuGet package to the application by using the **NuGet Package Manager**.
2. Add the client-side resources through [CDN](#) or from [NuGet](#) package in the **HEAD** element of the `~/Pages/_Host.cshtml` page.

HTML

```
<head>
<link href="_content/Syncfusion.Blazor.Themes/bootstrap4.css"
rel="stylesheet" />
<!--CDN-->
@*<link href="https://cdn.syncfusion.com/blazor/{{ site.blazorversion
}}/styles/bootstrap4.css" rel="stylesheet" />*@
</head>
```

For the Internet Explorer 11, kindly refer to the polyfills. For more information, refer to the [documentation](#)

HTML

```
<head>
<link href="https://cdn.syncfusion.com/blazor/{{ site.blazorversion
}}/styles/bootstrap4.css" rel="stylesheet" />
<script
src="https://github.com/Daddoon/Blazor.Polyfill/releases/download/3.0.1/blaz
or.polyfill.min.js"></script>
</head>
```

Adding component package to the application

Open the `~/_Imports.razor` file and include the **Syncfusion.Blazor.Charts** namespace.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
```

Adding SyncfusionBlazor service in Startup.cs

Open the **Startup.cs** file and add services required by the Syncfusion components using **services.AddSyncfusionBlazor()** method. Add this method in the **ConfigureServices** function as follows.

CSHARP

```
using Syncfusion.Blazor;
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void ConfigureServices(IServiceCollection services)
        {
            ....
            ....
            services.AddSyncfusionBlazor();
        }
    }
}
```

To enable the custom client-side source loading from CRG or CDN, please refer to the section about the [custom resources in Blazor application](#).

Adding Sparkline Component

To initialize the Sparkline component, add the below code to the **Index.razor** view page under **~/Pages** folder. In a new application, if **Index.razor** page has any default content template, then those content can be completely removed and the following code can be added.

ASPX-CS

```
@page "/"
<SfSparkline>
</SfSparkline>
```

Populate Sparkline with Data

To bind data for the Sparkline component, assign a **IEnumerable** object to the [DataSource](#) property. It can also be provided as an instance of the [DataManager](#).

ASPX-CS

```
@code {
    public class WeatherReport
    {
        public string Month { get; set; }
        public double Celsius { get; set; }
    };
    public List<WeatherReport> ClimateData = new List<WeatherReport> {
        new WeatherReport { Month= "Jan", Celsius= 34 },
        new WeatherReport { Month= "Feb", Celsius= 36 },
        new WeatherReport { Month= "Mar", Celsius= 32 },
        new WeatherReport { Month= "Apr", Celsius= 35 },
        new WeatherReport { Month= "May", Celsius= 40 },
        new WeatherReport { Month= "Jun", Celsius= 38 },
        new WeatherReport { Month= "Jul", Celsius= 33 },
        new WeatherReport { Month= "Aug", Celsius= 37 },
        new WeatherReport { Month= "Sep", Celsius= 34 },
        new WeatherReport { Month= "Oct", Celsius= 31 },
    };
}
```



```
new WeatherReport { Month= "Nov", Celsius= 30 },
new WeatherReport { Month= "Dec", Celsius= 29}
};
}
```

Now map the **Month** and the **Celsius** fields from the datasource to [XName](#) and [YName](#) properties for x-axis and y-axis in the Sparkline and then set the **ClimateData** to [DataSource](#) property. Because the **Month** field is a value-based category, the [ValueType](#) property is used to specify it.

ASPX-CS

```
<SfSparkline XName="Month"
YName="Celsius"
ValueType="SparklineValueType.Category"
TValue="WeatherReport"
DataSource="ClimateData"
Height="80px"
Width="150px">
</SfSparkline>
```

On successful compilation of the application, the Syncfusion Blazor Sparkline component will render in the web browser as following.



Change the type of Sparkline

Change the Sparkline type using the [Type](#) property set to **Line**, **Column**, **WinLoss**, **Pie** or **Area**. Here, the Sparkline type is set to **Area**.

ASPX-CS

```
<SfSparkline XName="Month"
YName="Celsius"
ValueType="SparklineValueType.Category"
Type="SparklineType.Area"
TValue="WeatherReport"
DataSource="ClimateData"
Height="80px"
Width="150px">
</SfSparkline>
```



Refer to [code block](#) to know about the property value of **ClimateData**.

Adding Data Label

Add the Data Labels to improve the readability of the Sparkline component. This can be achieved by setting the [Visible](#) property to **true** in the [SparklineDataLabelSettings](#).

Available types are:

- Start
- End
- All
- High
- Low
- Negative

ASPX-CS

```
<SfSparkline DataSource="ClimateData"
TValue="WeatherReport"
XName="Month"
YName="Celsius"
ValueType="SparklineValueType.Category"
Height="80px"
Width="150px">
<SparklineDataLabelSettings Visible="new List<VisibleType> {
VisibleType.Start, VisibleType.End }"></SparklineDataLabelSettings>
<SparklinePadding Left="10" Right="10"></SparklinePadding>
</SfSparkline>
```

Refer to the [code block](#) to know about the property value of **ClimateData**.



Enable tooltip

When space constraints prevent from displaying information using Data Labels, the tooltip comes in handy. The tooltip can be enabled by setting the [Visible](#) property to **true** in the [SparklineTooltipSettings](#).

ASPX-CS

```
<SfSparkline DataSource="ClimateData"
TValue="WeatherReport"
XName="Month"
YName="Celsius"
ValueType="SparklineValueType.Category"
Height="80px"
Width="150px">
<SparklineDataLabelSettings Visible="new List<VisibleType> {
VisibleType.Start, VisibleType.End }"></SparklineDataLabelSettings>
<SparklinePadding Left="10" Right="10"></SparklinePadding>
</SfSparkline>
```

```
<SparklineTooltipSettings TValue="WeatherReport"
Visible="true"></SparklineTooltipSettings>
</SfSparkline>
```

Refer to the [code block](#) to know about the property value of the **ClimateData**.



See also

- [Getting Started with Syncfusion Blazor for Client-Side in .NET Core CLI](#)
- [Getting Started with Syncfusion Blazor for Server-Side in Visual Studio](#)
- [Getting Started with Syncfusion Blazor for Server-Side in .NET Core CLI](#)

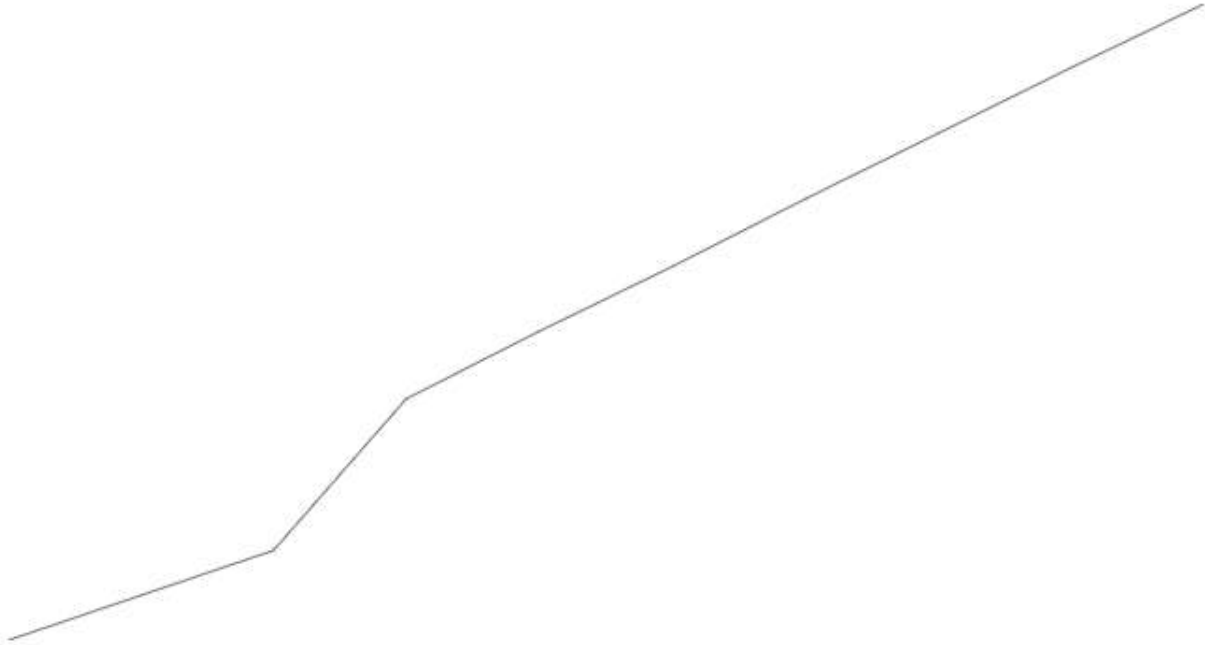
Dimensions in Blazor Sparkline Component

Size for the container

The size of the Sparkline Chart is determined by the container size, and it can be changed inline or via CSS as following.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<div style="width:650px; height:350px;">
<SfSparkline XName="Year" YName="Population" TValue="PopulationReport"
DataSource="PopulationData"></SfSparkline>
</div>
@code {
public class PopulationReport
{
public int Year;
public int Population;
};
private List<PopulationReport> populationData = new List<PopulationReport> {
new PopulationReport { Year= 2005, Population= 20090440 },
new PopulationReport { Year= 2006, Population= 20264080 },
new PopulationReport { Year= 2007, Population= 20434180 },
new PopulationReport { Year= 2008, Population= 21007310 },
new PopulationReport { Year= 2009, Population= 21262640 },
new PopulationReport { Year= 2010, Population= 21515750 },
new PopulationReport { Year= 2011, Population= 21766710 },
new PopulationReport { Year= 2012, Population= 22015580 },
new PopulationReport { Year= 2013, Population= 22262500 },
new PopulationReport { Year= 2014, Population= 22507620 }
};
}
```



Size for Sparkline

The [Width](#) and the [Height](#) properties can be used to set the size of the Sparkline Chart.

In Pixel

The Sparkline Chart can be sized in pixels.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline XName="Year"
YName="Population"
TValue="PopulationReport"
DataSource="PopulationData"
Width="350px"
Height="150px">
</SfSparkline>
```

Refer to the [code block](#) to know about the property value of the `populationData`.



In Percentage

By setting values in percentage, the Sparkline Chart gets their dimension with respect to their containers. For example, when the height is set to "50%", the Sparkline Chart is rendered to half of its container width.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<div style="width:650px; height:350px;">
<SfSparkline XName="Year"
YName="Population"
TValue="PopulationReport"
DataSource="PopulationData"
Width="50%"
Height="80%">
</SfSparkline>
</div>
```

Refer to the [code block](#) to know about the property value of the `populationData`.

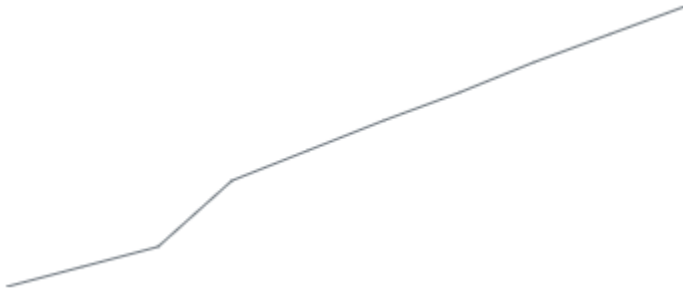


Chart Types in Blazor Sparkline Component

The different type of shapes can be used to represent the Sparkline Chart, and the type of Sparkline Chart can be changed by specifying the [Type](#) property.

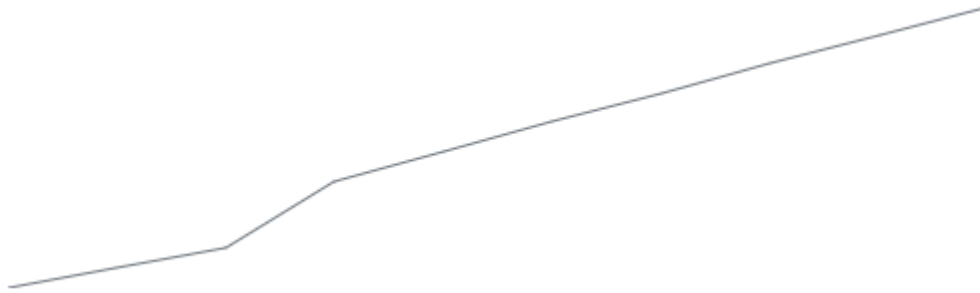
Line

The [Line](#) chart type is used to render the Sparkline series as line.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline DataSource="PopulationData" TValue="PopulationReport"
XName="Year" YName="Population" Width="200px" Height="150px"
Type="SparklineType.Line">
</SfSparkline>
@code {
public class PopulationReport
{
public int Year { get; set; }
public int Population { get; set; }
};
private List<PopulationReport> PopulationData = new List<PopulationReport> {
new PopulationReport { Year= 2005, Population= 20090440 },
new PopulationReport { Year= 2006, Population= 20264080 },
new PopulationReport { Year= 2007, Population= 20434180 },
new PopulationReport { Year= 2008, Population= 21007310 },
```

```
new PopulationReport { Year= 2009, Population= 21262640 },
new PopulationReport { Year= 2010, Population= 21515750 },
new PopulationReport { Year= 2011, Population= 21766710 },
new PopulationReport { Year= 2012, Population= 22015580 },
new PopulationReport { Year= 2013, Population= 22262500 },
new PopulationReport { Year= 2014, Population= 22507620 }
};
}
```



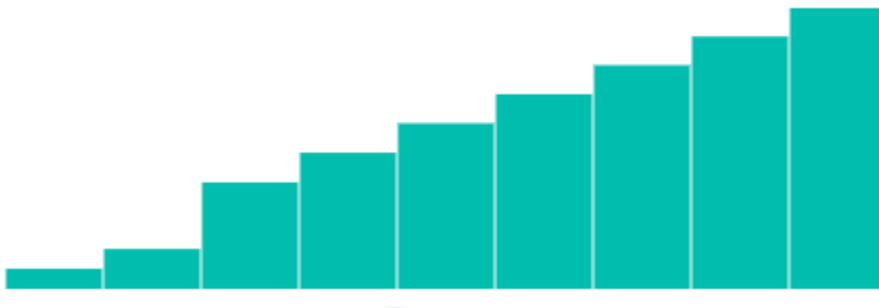
Column

The [Column](#) chart type is used to render the Sparkline series as column.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline DataSource="PopulationData" TValue="PopulationReport"
XName="Year" YName="Population" Width="500px" Height="150px"
Type="SparklineType.Column">
</SfSparkline>
```

Refer to the [code block](#) to know about the property value of the **PopulationData**.



Pie

The [Pie](#) chart type is used to render the Sparkline series as pie.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline DataSource="PopulationData" TValue="PopulationReport"
XName="Year" YName="Population" Width="500px" Height="250px"
Type="SparklineType.Pie">
</SfSparkline>
```

Refer to the [code block](#) to know about the property value of the **PopulationData**.



WinLoss

The [WinLoss](#) chart type is used to render the Sparkline series as WinLoss.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline DataSource="new int[]{12, 15, -10, 13, 15, 6, -12, 17, 13, 0, 8, -10}" Width="500px" Height="200px" Type="SparklineType.WinLoss">
</SfSparkline>
```



Area

The [Area](#) chart type is used to render the Sparkline series as area.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline DataSource="PopulationData" TValue="PopulationReport"
XName="Year" YName="Population" Width="500px" Height="100px"
Type="SparklineType.Area">
</SfSparkline>
```

Refer to the [code block](#) to know about the property value of the **PopulationData**.



Axis Customization in Blazor Sparkline Component

Change the value type of the Sparkline Chart

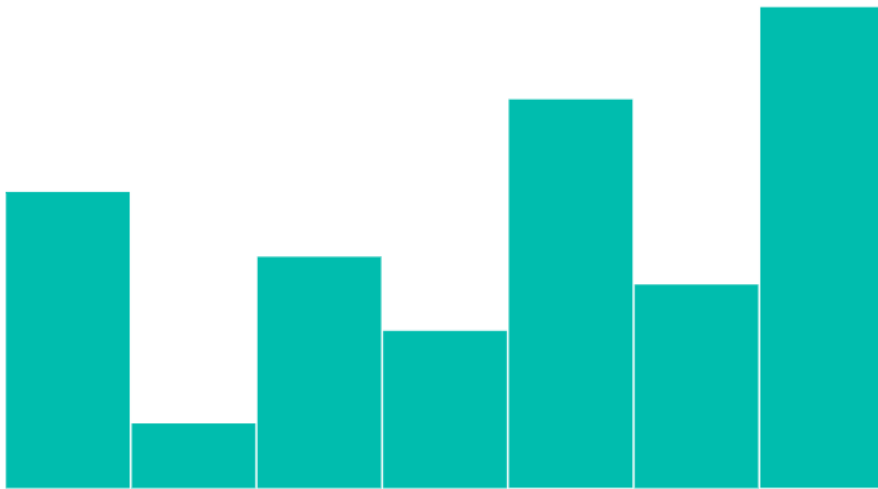
The [ValueType](#) property is used to specify the Sparkline value type, which can be [Numeric](#), [Category](#), or [DateTime](#).

Numeric

The numeric axis value can be provided by specifying the [ValueType](#) property to the [Numeric](#).

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline DataSource="@ExpenditureReport" TValue="ExpenditureDetails"
XName="Year" YName="Expense" Type="SparklineType.Column"
ValueType="SparklineValueType.Numeric" Height="250px" Width="450px">
</SfSparkline>
@code {
public class ExpenditureDetails
{
public int Year { get; set; }
public int Expense { get; set; }
};
public List<ExpenditureDetails> ExpenditureReport = new
List<ExpenditureDetails> {
new ExpenditureDetails{ Year= 2010, Expense= 190 },
new ExpenditureDetails{ Year= 2011, Expense= 165 },
new ExpenditureDetails{ Year= 2012, Expense= 158 },
new ExpenditureDetails{ Year= 2013, Expense= 175 },
new ExpenditureDetails{ Year= 2014, Expense= 200 },
new ExpenditureDetails{ Year= 2015, Expense= 180 },
new ExpenditureDetails{ Year= 2016, Expense= 210 }
};
}
```

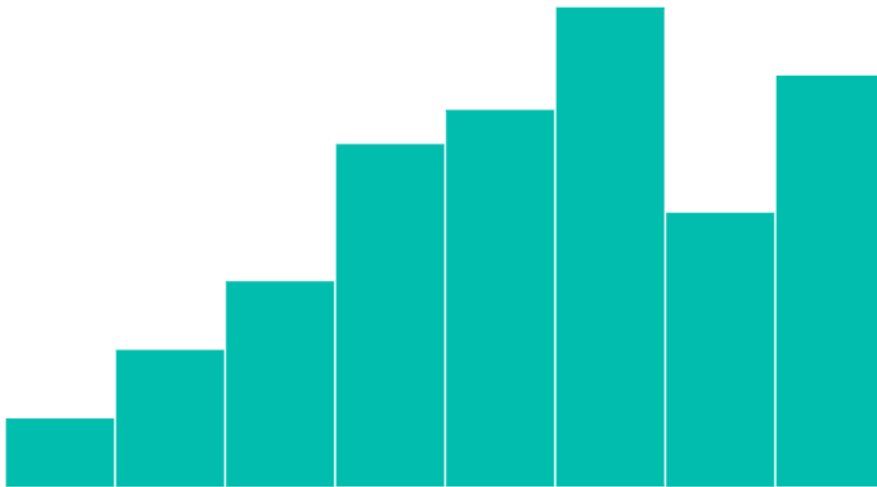



Category

The category axis value can be provided by specifying the [ValueType](#) property to the [Category](#).

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline
XName="EmployeeName"YName="WorkHours" TValue="WorkDetails"DataSource="@EmployeeWorkReport"
Type="SparklineType.Column"ValueType="SparklineValueType.Category"
Height="250px"Width="450px">
</SfSparkline>
@code {
public class WorkDetails
{
public string EmployeeName { get; set; }
public double WorkHours { get; set; }
};
public List<WorkDetails> EmployeeWorkReport = new List<WorkDetails> {
new WorkDetails { EmployeeName = "Robert", WorkHours= 60 },
new WorkDetails { EmployeeName = "Andrew", WorkHours= 65 },
new WorkDetails { EmployeeName = "Suyama", WorkHours= 70 },
new WorkDetails { EmployeeName = "Michael", WorkHours= 80 },
new WorkDetails { EmployeeName = "Janet", WorkHours= 55 },
new WorkDetails { EmployeeName = "Davolio", WorkHours= 90 },
new WorkDetails { EmployeeName = "Fuller", WorkHours= 75 },
new WorkDetails { EmployeeName = "Nancy", WorkHours= 85 }
};
}
```

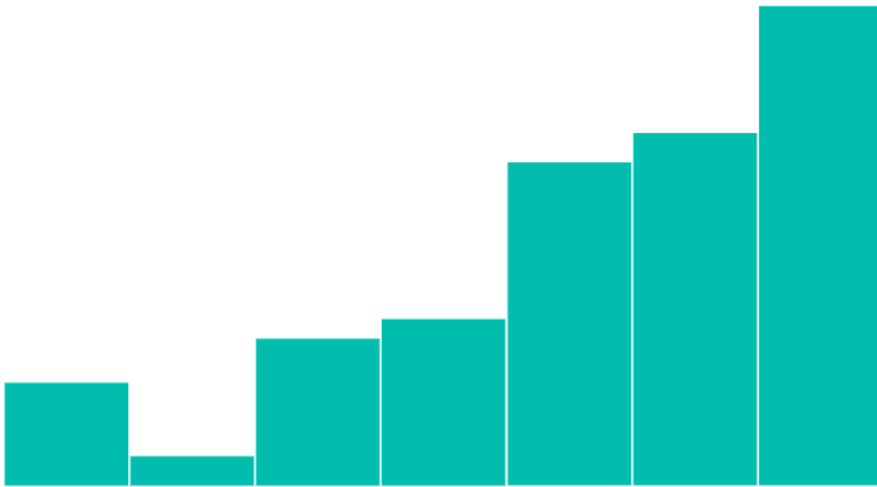


DateTime

The DateTime axis value can be provided by specifying the [ValueType](#) property to the [DateTime](#).

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline XName="Date" YName="Expense" TValue="ExpenditureDetail"
DataSource="@ExpenditureReports" Type="SparklineType.Column"
ValueType="SparklineValueType.DateTime" Height="250px" Width="450px">
</SfSparkline>
@code {
    public class ExpenditureDetail
    {
        public DateTime Date { get; set; }
        public double Expense { get; set; }
    }
    public List<ExpenditureDetail> ExpenditureReports = new
    List<ExpenditureDetail>
    {
        new ExpenditureDetail { Date = new DateTime(2005, 01, 01), Expense = 21 },
        new ExpenditureDetail { Date = new DateTime(2006, 01, 01), Expense = 24 },
        new ExpenditureDetail { Date = new DateTime(2007, 01, 01), Expense = 36 },
        new ExpenditureDetail { Date = new DateTime(2008, 01, 01), Expense = 38 },
        new ExpenditureDetail { Date = new DateTime(2009, 01, 01), Expense = 54 },
        new ExpenditureDetail { Date = new DateTime(2010, 01, 01), Expense = 57 },
        new ExpenditureDetail { Date = new DateTime(2011, 01, 01), Expense = 70 }
    };
}
```



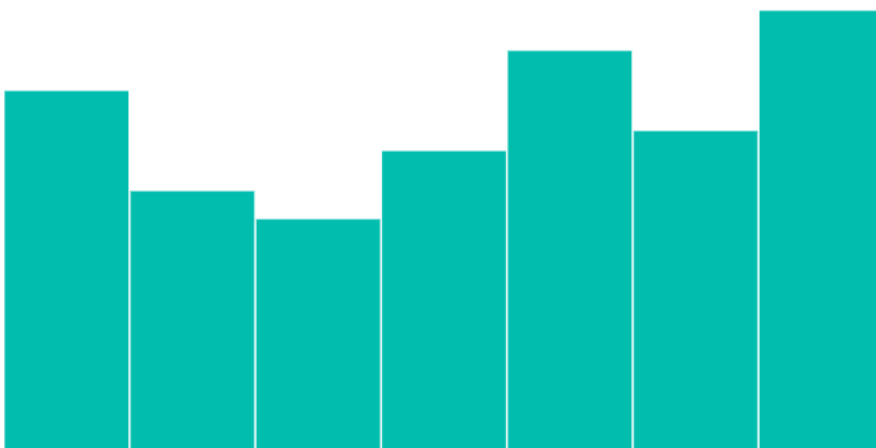
Change the min and the max values of axis

The min and the max values of the X-axis can be customized using the [MinX](#) and the [MaxX](#) properties of the [SparklineAxisSettings](#), and the min and the max values of the Y-axis using the [MinY](#) and the [MaxY](#) properties.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline DataSource="@ExpenditureReport" TValue="ExpenditureDetails"
XName="Year" YName="Expense" Type="SparklineType.Column"
ValueType="SparklineValueType.Numeric" Height="250px" Width="450px">
<SparklineAxisSettings MinY="100" MaxY="220"></SparklineAxisSettings>
</SfSparkline>
```

Refer to the [code block](#) to know about the property value of the **ExpenditureReport**.



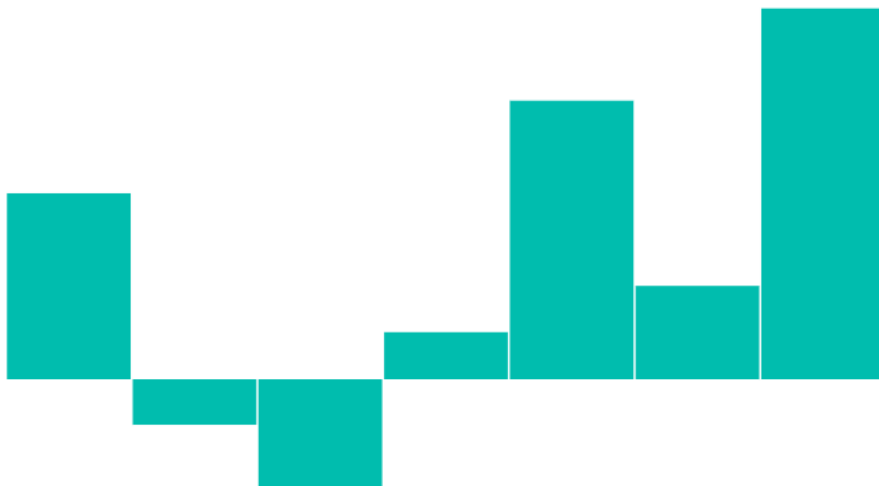
Change value of axis

The horizontal axis line value can be customized by setting the [Value](#) in the [SparklineAxisSettings](#).

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline DataSource="@ExpenditureReport" TValue="ExpenditureDetails"
XName="Year" YName="Expense" Type="SparklineType.Column"
ValueType="SparklineValueType.Numeric" Height="250px" Width="450px">
<SparklineAxisSettings Value="170"></SparklineAxisSettings>
</SfSparkline>
```

Refer to the [code block](#) to know about the property value of the **ExpenditureReport**.

**Axis Line Customization**

The axis line can be enabled by specifying the [Visible](#) property to **true** in the [SparklineAxisLineSettings](#).

The axis line is not applicable for the [WinLoss](#) chart type.

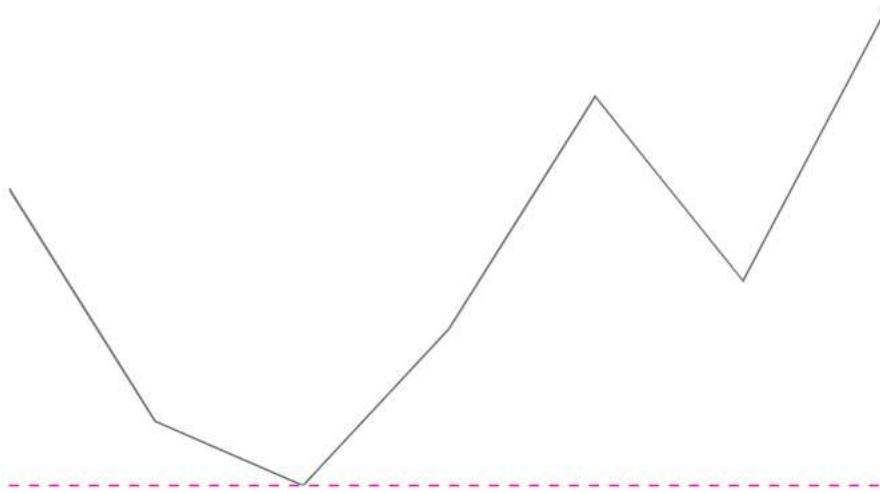
The axis line can be customized using the following properties.

- [Color](#) - Specifies the color of the axis line.
- [Opacity](#) - Specifies the opacity for the [Color](#) of the axis line.
- [Width](#) - Specifies the width of the axis line.
- [DashArray](#) - Specifies the dash array of the axis line.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline DataSource="ExpenditureReport" TValue="ExpenditureDetails"
XName="Year" YName="Expense" Type="SparklineType.Line" Height="250px"
Width="450px">
<SparklineAxisSettings>
<SparklineAxisLineSettings Visible="true" Color="#ff14ae" DashArray="5"
Opacity="1"></SparklineAxisLineSettings>
</SparklineAxisSettings>
</SfSparkline>
```

Refer to the [code block](#) to know about the property value of **ExpenditureReport**.



Special Points Customization in Blazor Sparkline Component

Add custom color for special points

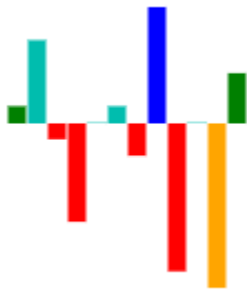
The color of special points can be changed by specifying the corresponding property, and it is applicable for [Line](#), [Column](#) and [Area](#) chart types in the Sparkline. The following properties are used to customize the special points.

- [StartPointColor](#)
- [EndPointColor](#)
- [NegativePointColor](#)
- [LowPointColor](#)
- [HighPointColor](#)

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline XName="CarName" YName="Rating" Width="130px" Height="150px"
TValue="CarRating" DataSource="CarRatings" Type="SparklineType.Column"
ValueType="SparklineValueType.Category"
HighPointColor="blue" LowPointColor="orange" StartPointColor="green"
EndPointColor="green" NegativePointColor="red">
</SfSparkline>
@code {
public class CarRating
{
public string CarName { get; set; }
public double Rating { get; set; }
};
public List<CarRating> CarRatings = new List<CarRating> {
new CarRating { CarName= "AUDI", Rating= 1 },
new CarRating { CarName= "BMW", Rating= 5 },
new CarRating { CarName= "BUICK", Rating= -1 },
new CarRating { CarName= "CETROEN", Rating= -6 },
new CarRating { CarName= "CHEVROLET", Rating= 0.01 },
new CarRating { CarName= "FIAT", Rating= 1 },
new CarRating { CarName= "FORD", Rating= -2 },
```

```
new CarRating { CarName= "HONDA", Rating= 7 },
new CarRating { CarName= "HYUNDAI", Rating= -9 },
new CarRating { CarName= "JEEP", Rating= 0.01 },
new CarRating { CarName= "KIA", Rating= -10 },
new CarRating { CarName= "MAZDA", Rating= 3 }
};
}
```

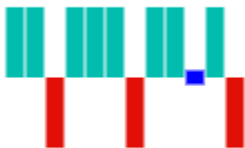


Tie point color

To highlight the tie area of the Y-axis value, use the [TiePointColor](#) property that is only applicable to the [WinLoss](#) type.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline Width="130px" Height="150px" Type="SparklineType.WinLoss"
TiePointColor="blue" DataSource="new int[]{12, 15, -10, 13, 15, 6, -12, 17,
13, 0, 8, -10}">
</SfSparkline>
```



Range Band in Blazor Sparkline Component

Range band represents the quality or improves the readability of a specific range for the Sparkline y-axis by specifying the [StartRange](#) and [EndRange](#) properties in the [SparklineRangeBand](#) in the [SparklineRangeBandSettings](#).

The following properties are used for the customization of the range band:

- [Color](#) - Specifies the color of the range band.
- [Opacity](#) - Specifies the opacity of [Color](#) in the range band.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
```

```
<SfSparkline DataSource="new int[]{ 0, 6, 4, 1, 3, 2, 5 }" Height="150px"
Width="150px" LineWidth="2" Fill="#0d3c9b">
  <SparklineAxisSettings MinX="-1" MaxX="7" MaxY="7" MinY="-1">
  </SparklineAxisSettings>
  <SparklineRangeBandSettings>
  <SparklineRangeBand StartRange="1" EndRange="2" Color="#bfd4fc"
Opacity="0.4">
  </SparklineRangeBand>
  <SparklineRangeBand StartRange="4" EndRange="5" Color="red" Opacity="0.4">
  </SparklineRangeBand>
  </SparklineRangeBandSettings>
</SfSparkline>
```



Markers in Blazor Sparkline Component

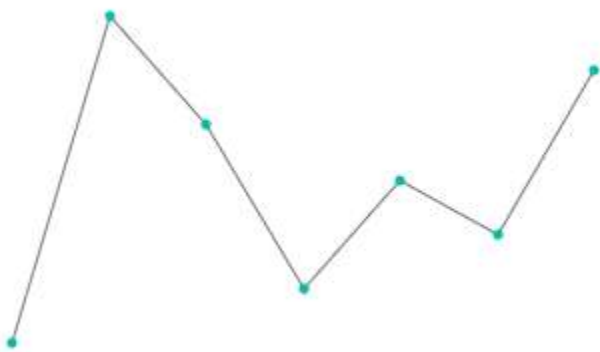
Data markers are used to provide information about the data points in the Sparkline series.

Adding markers

The [Visible](#) property in the [SparklineMarkerSettings](#) can be used to enable a marker by specifying a collection of special points. The following code example shows how to enable markers for all points.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline DataSource="new int[]{ 0, 6, 4, 1, 3, 2, 5 }"
Type="SparklineType.Line" Height="200px" Width="350px">
  <SparklineMarkerSettings Visible="new List<VisibleType> { VisibleType.All
}"></SparklineMarkerSettings>
  <SparklineAxisSettings MinX="-1" MaxX="7" MaxY="7" MinY="-
1"></SparklineAxisSettings>
</SfSparkline>
```



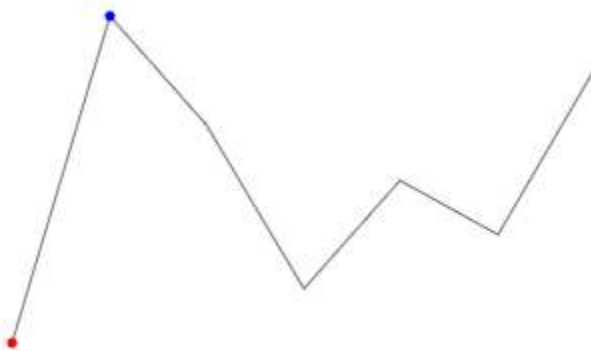
Adding special point markers

The markers can be enabled for specific points as a collection. The following special points are applicable for markers.

- [All](#) - Markers for all points are enabled.
- [Start](#) - Markers for start points are enabled.
- [End](#) - Markers for end points are enabled.
- [High](#) - Markers for high points are enabled.
- [Low](#) - Markers for low points are enabled.
- [Negative](#) - Markers for negative points are enabled.
- [None](#) - Markers for all points are disabled.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline DataSource="new int[]{ 0, 6, 4, 1, 3, 2, 5 }"
Type="SparklineType.Line" Height="200px" Width="350px" HighPointColor="Blue"
LowPointColor="Red">
<SparklineMarkerSettings Visible="new List<VisibleType> { VisibleType.High,
VisibleType.Low }"></SparklineMarkerSettings>
<SparklineAxisSettings MinX="-1" MaxX="7" MaxY="7" MinY="-1"></SparklineAxisSettings>
</SfSparkline>
```



Markers customization

The following properties can be used to customize markers:

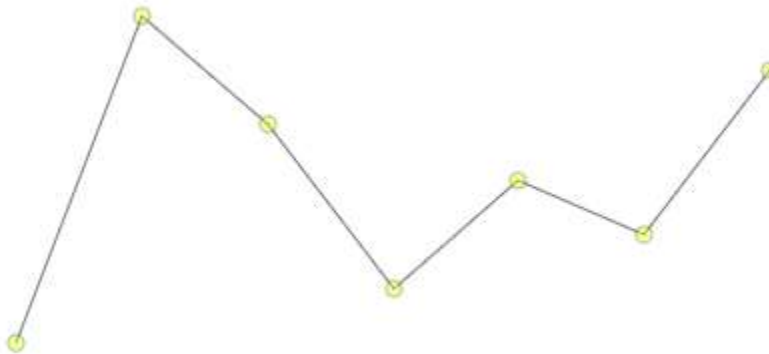
- [Fill](#) - Specifies fill color for marker.
- [Opacity](#) - Specifies opacity of [Fill](#) color for marker.
- [Size](#) - Specifies marker size.
- [SparklineMarkerBorder](#) - Specifies color and width for marker border.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline DataSource="new int[]{ 0, 6, 4, 1, 3, 2, 5 }"
Type="SparklineType.Line" Height="200px" Width="450px">
```



```
<SparklineMarkerSettings Visible="new List<VisibleType> { VisibleType.All }"
Fill="yellow" Opacity="0.4" Size="8">
<SparklineMarkerBorder Color="green" Width="1">
</SparklineMarkerBorder>
</SparklineMarkerSettings>
<SparklineAxisSettings MinX="-1" MaxX="7" MaxY="7" MinY="-
1"></SparklineAxisSettings>
</SfSparkline>
```



Data Labels in Blazor Sparkline Component

To improve readability, the Data Labels are used to display the value of data points.

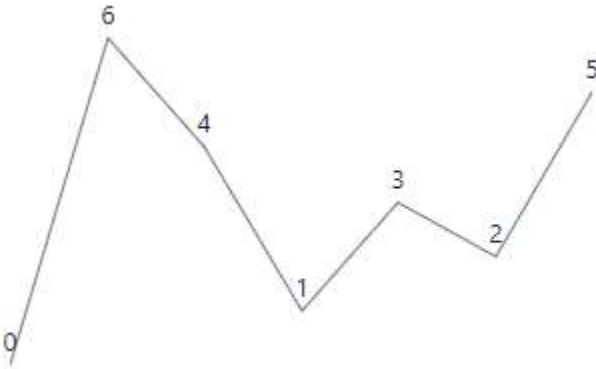
Enable Data Label

The [Visible](#) property in the [SparklineDataLabelSettings](#) can be used to enable the Data Label by specifying a collection of special points. The following special points are applicable for the Sparkline Data Label.

- [All](#) - Data label for all points are enabled.
- [Start](#) - Data label for start points are enabled.
- [End](#) - Data label for end points are enabled.
- [High](#) - Data label for high points are enabled.
- [Low](#) - Data label for low points are enabled.
- [Negative](#) - Data label for negative points are enabled.
- [None](#) - Data label for all points are disabled.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline DataSource="new int[]{ 0, 6, 4, 1, 3, 2, 5 }"
Type="SparklineType.Line" Height="200px" Width="350px">
<SparklineAxisSettings MinX="-1" MaxX="7" MaxY="7" MinY="-
1"></SparklineAxisSettings>
<SparklineDataLabelSettings Visible="new List<VisibleType>() {
VisibleType.All }"></SparklineDataLabelSettings>
</SfSparkline>
```



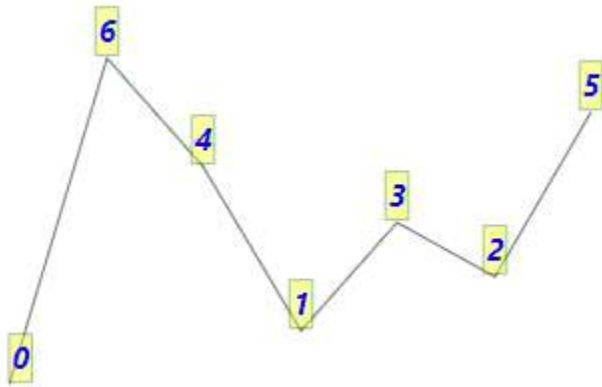
Data Label customization

The following properties can be used to customize the Sparkline Data Label:

- [Fill](#) - Specifies color for the Data Label.
- [Opacity](#) - Specifies opacity of [Fill](#) color for the Data Label.
- [EdgeLabelMode](#) - Specifies controlling option when the label comes in the edge. Available options are the [Shift](#), the [None](#) and the [Hide](#).
- [SparklineFont](#) - To customize the Data Label font family, font style, font weight, color, opacity and size.
- [SparklineDataLabelBorder](#) - Specifies the color and the width for the Data Label border.
- [SparklineDataLabelOffset](#) - Specifies the label offset position from its default position.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline DataSource="new int[]{ 0, 6, 4, 1, 3, 2, 5 }"
Type="SparklineType.Line" Height="200px" Width="350px">
<SparklineAxisSettings MinX="-1" MaxX="7" MaxY="7" MinY="-1"></SparklineAxisSettings>
<SparklineDataLabelSettings Visible="new List<VisibleType>() {
VisibleType.All }" Fill="yellow" Opacity="0.4"
EdgeLabelMode="EdgeLabelMode.Shift">
<SparklineFont Color="blue" FontStyle="italic" FontWeight="bold" Size="15"
Opacity="0.8">
</SparklineFont>
<SparklineDataLabelBorder Color="green" Width="1">
</SparklineDataLabelBorder>
</SparklineDataLabelSettings>
</SfSparkline>
```

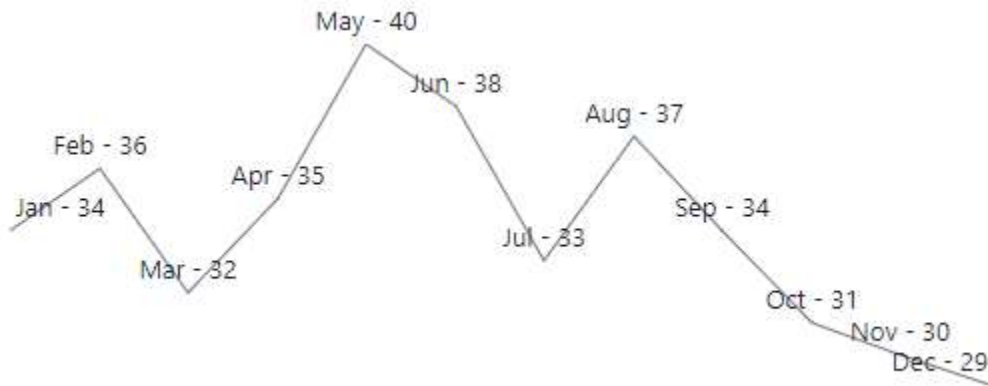


Format

The Data Label text can be formatted by specifying the property name from the datasource to the [Format](#) property in the [SparklineDataLabelSettings](#). By default, Data Label text will be based on [YName](#) property.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline DataSource="ClimateData" TValue="WeatherReport" XName="Month"
YName="Celsius" ValueType="SparklineValueType.Category" Height="200px"
Width="500px">
  <SparklineDataLabelSettings Visible="new List<VisibleType> {
  VisibleType.All}" Format="{Month} - {Celsius}"
  EdgeLabelMode="EdgeLabelMode.Shift">
  </SparklineDataLabelSettings>
  <SparklinePadding Top="25"></SparklinePadding>
</SfSparkline>
@code {
public class WeatherReport
{
public string Month { get; set; }
public double Celsius { get; set; }
};
public List<WeatherReport> ClimateData = new List<WeatherReport> {
new WeatherReport { Month= "Jan", Celsius= 34 },
new WeatherReport { Month= "Feb", Celsius= 36 },
new WeatherReport { Month= "Mar", Celsius= 32 },
new WeatherReport { Month= "Apr", Celsius= 35 },
new WeatherReport { Month= "May", Celsius= 40 },
new WeatherReport { Month= "Jun", Celsius= 38 },
new WeatherReport { Month= "Jul", Celsius= 33 },
new WeatherReport { Month= "Aug", Celsius= 37 },
new WeatherReport { Month= "Sep", Celsius= 34 },
new WeatherReport { Month= "Oct", Celsius= 31 },
new WeatherReport { Month= "Nov", Celsius= 30 },
new WeatherReport { Month= "Dec", Celsius= 29}
};
}
```



User Interaction in Blazor Sparkline Component

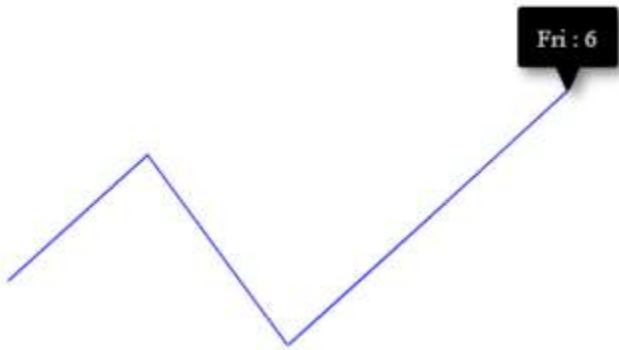
The Sparkline's user interaction features include the tooltip and the tracker line.

Tooltip

When the mouse is hovered over a data point, the Sparkline provides the option to display details about the value of the data point via a tooltip. The following code example shows how to enable Sparkline's tooltip with a custom format.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline Width="500" Height="200" TValue="WorkLog" DataSource="WorkLogs"
XName="Day" YName="Hour" Fill="blue"
ValueType="SparklineValueType.Category">
<SparklineAxisSettings MinX="-1" MaxX="7" MaxY="8" MinY="-1">
</SparklineAxisSettings>
<SparklineTooltipSettings TValue="WorkLog" Visible="true" Format="${Day} :
${Hour}">
</SparklineTooltipSettings>
</SfSparkline>
@code {
public class WorkLog
{
public string Day { get; set; }
public double Hour { get; set; }
};
public List<WorkLog> WorkLogs = new List<WorkLog> {
new WorkLog {Day= "Mon", Hour= 3 },
new WorkLog {Day= "Tue", Hour= 5 },
new WorkLog {Day= "Wed", Hour= 2 },
new WorkLog {Day= "Thu", Hour= 4 },
new WorkLog {Day= "Fri", Hour= 6 }
};
}
```



Tooltip Customization

The following properties can be used to customize the Sparkline tooltip:

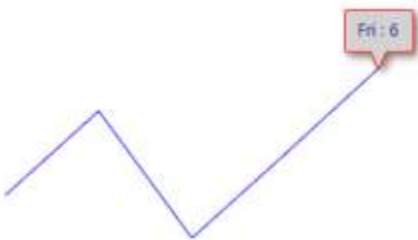
- [Fill](#) - Specifies fill color for the tooltip.
- [Format](#) - Specifies custom content of tooltip by assigning the properties from the datasource.
- [SparklineTooltipTextStyle](#) - Specifies font family, font style, font weight, color, opacity and size of the tooltip content.
- [SparklineTooltipBorder](#) - To customize border width and color of the tooltip.

The following code example shows customizing tooltip format, text color and fill color.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline Width="500" Height="200" TValue="WorkLog" DataSource="WorkLogs"
XName="Day" YName="Hour" Fill="blue"
ValueType="SparklineValueType.Category">
<SparklineAxisSettings MinX="-1" MaxX="7" MaxY="8" MinY="-1">
</SparklineAxisSettings>
<SparklineTooltipSettings TValue="WorkLog" Visible="true" Format="{Day} :
{Hour}" Fill="lightgray">
<SparklineTooltipTextStyle Color="darkblue"></SparklineTooltipTextStyle>
<SparklineTooltipBorder Color="red" Width="1"></SparklineTooltipBorder>
</SparklineTooltipSettings>
</SfSparkline>
```

Refer to the [code block](#) to know about the property value of **WorkLogs**.



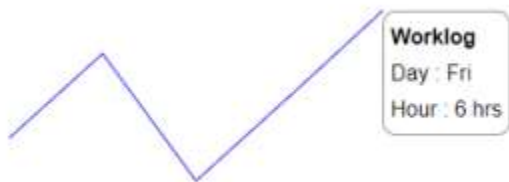
Tooltip Template

The tooltip can be rendered as a custom component by specifying the [Template](#) property in the [SparklineTooltipSettings](#) that accepts one or more UI elements as an input and renders them as a part of the tooltip rendering.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline Width="500" Height="200" TValue="WorkLog" DataSource="WorkLogs"
XName="Day" YName="Hour" Fill="blue"
ValueType="SparklineValueType.Category">
<SparklineAxisSettings MinX="-1" MaxX="7" MaxY="8" MinY="-1">
</SparklineAxisSettings>
<SparklineTooltipSettings TValue="WorkLog" Visible="true" Fill="lightgray">
<Template>
@{
<table style="width:100%; background-color: #ffffff; border-spacing: 0px;
border-collapse: separate; border: 1px solid grey; border-radius: 10px;
padding-top: 5px; padding-bottom: 5px">
<tr>
<td style="font-weight: bold; color: black; padding-left: 5px; padding-top: 2px; padding-bottom: 2px;">Worklog</td>
</tr>
<tr>
<td style="padding-left: 5px; color: black; padding-right: 5px; padding-bottom: 2px;">Day : @context.Day </td>
</tr>
<tr>
<td style="padding-left: 5px; color: black; padding-right: 5px;">Hour :
@context.Hour hrs </td>
</tr>
</table>
}
</Template>
</SparklineTooltipSettings>
</SfSparkline>
```

Refer to the [code block](#) to know about the property value of the **WorkLogs**.

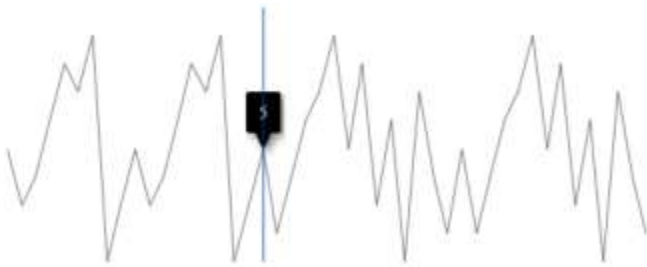


Track Line

The track line tracks data points that are closest to the mouse position or touch interaction, and it can be enabled by setting the [Visible](#) property to **true** in the [SparklineTrackLineSettings](#). The track line color and width can be customized using the [Color](#) and the [Width](#) properties in the [SparklineTrackLineSettings](#).

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline Width="500px" Height="200px"
DataSource="new int[]{ 5, 3, 4, 6, 8, 7, 9, 1, 3, 5, 3, 4, 6, 8, 7, 9, 1, 3,
5, 2, 4, 6, 7, 9, 5, 8, 3, 6, 1, 7, 4, 2, 5, 2, 4, 6, 7, 9, 5, 8, 3, 6, 1,
7, 4, 2 }">
<SparklineAxisSettings MinX="-1" MaxX="46" MaxY="10" MinY="-1">
</SparklineAxisSettings>
<SparklineTooltipSettings TValue="int" Visible="true">
<SparklineTrackLineSettings Visible="true" Color="#033e96" Width="1">
</SparklineTrackLineSettings>
</SparklineTooltipSettings>
</SfSparkline>
```



Appearance in Blazor Sparkline Component

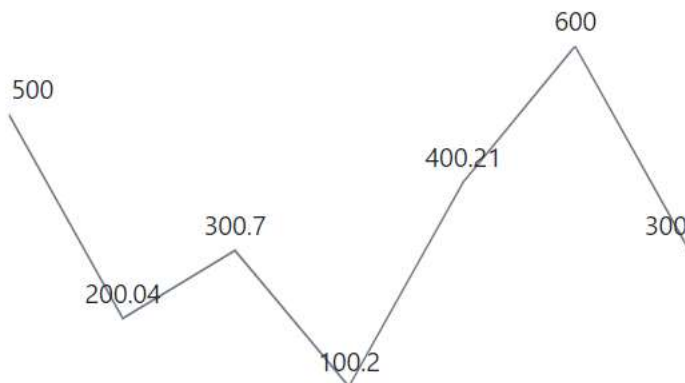
The rendering direction, padding, border, and the background appearance of the Sparkline can all be customized.

Right-to-left (RTL)

The Sparkline supports the right-to-left (RTL) rendering that can be enabled by setting the [EnableRtl](#) property to **true**.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline DataSource="new double[]{ 300.00, 600.00, 400.21, 100.20,
300.70, 200.04, 500.00 }" Height="200px" Width="350px" Format="c2"
EnableRtl="true">
<SparklineDataLabelSettings Visible="new List<VisibleType> { VisibleType.All
}" EdgeLabelMode="EdgeLabelMode.Shift"></SparklineDataLabelSettings>
<SparklinePadding Top="25"></SparklinePadding>
</SfSparkline>
```

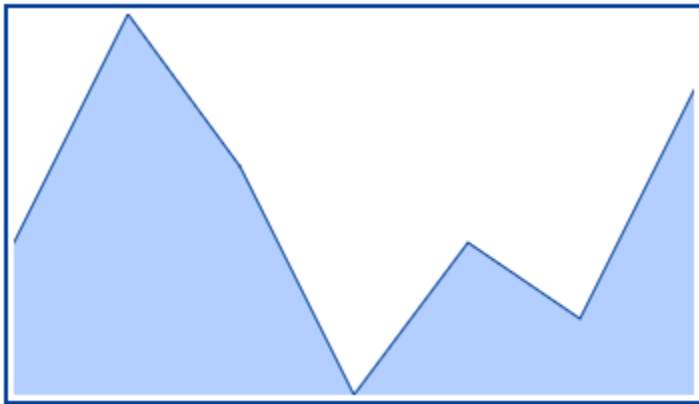


Border

The border can be enabled and customized by specifying the [Color](#) and the [Width](#) properties of the [SparklineContainerAreaBorder](#).

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline DataSource="new int[]{ 3, 6, 4, 1, 3, 2, 5 }"
Type="SparklineType.Area" Height="200px" Width="350px" Fill="#b2cfff"
LineWidth="1">
<SparklineContainerArea>
<SparklineContainerAreaBorder Color="#033e96"
Width="1"></SparklineContainerAreaBorder>
</SparklineContainerArea>
</SfSparkline>
```

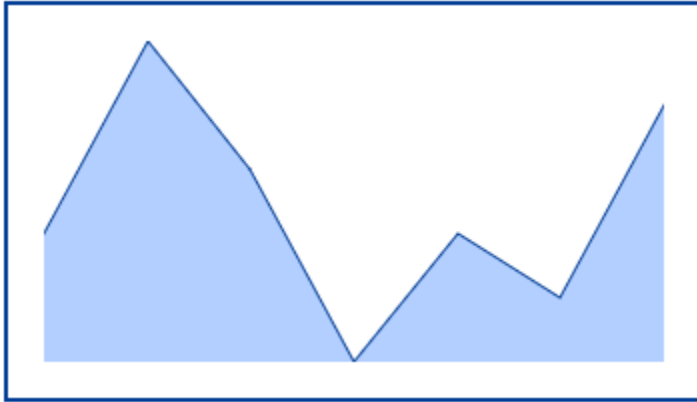


Padding

The Sparkline supports padding between the container and the component using the [SparklinePadding](#). The code example in the following shows the Sparkline Chart with overall padding set to 20.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline DataSource="new int[]{ 3, 6, 4, 1, 3, 2, 5 }"
Type="SparklineType.Area" Height="200px" Width="350px" Fill="#b2cfff"
LineWidth="1">
<SparklineContainerArea>
<SparklineContainerAreaBorder Color="#033e96"
Width="2"></SparklineContainerAreaBorder>
</SparklineContainerArea>
<SparklineBorder Color="#033e96" Width="1"></SparklineBorder>
<SparklinePadding Left="20" Right="20" Bottom="20"
Top="20"></SparklinePadding>
</SfSparkline>
```

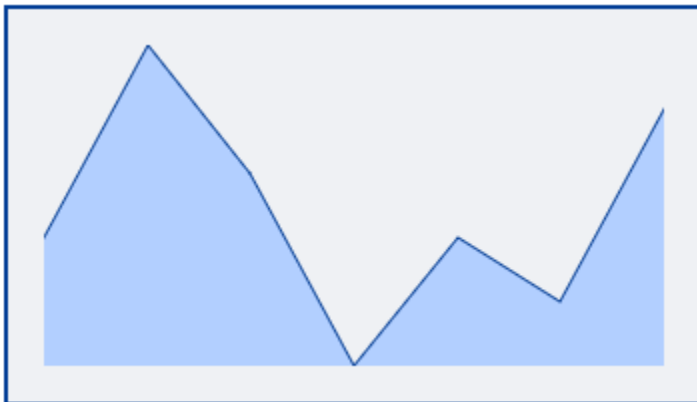



Background

The background color of the Sparkline area can be changed using the [Background](#) property of the [SparklineContainerArea](#). By default, the Sparkline background color is **Transparent**.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline DataSource="new int[]{ 3, 6, 4, 1, 3, 2, 5 }"
Type="SparklineType.Area" Height="200px" Width="350px" Fill="#b2cfff"
LineWidth="1">
  <SparklineContainerArea Background="#eff1f4">
    <SparklineContainerAreaBorder Color="#033e96" Width="2">
    </SparklineContainerAreaBorder>
  </SparklineContainerArea>
  <SparklineBorder Color="#033e96" Width="1"></SparklineBorder>
  <SparklinePadding Left="20" Right="20" Bottom="20"
Top="20"></SparklinePadding>
</SfSparkline>
```



Globalization in Blazor Sparkline Component

Globalization is the process of designing and developing a component that can work in different cultures or locations. In the Sparkline component, the [Format](#) property is used to globalize number, date, and time values. The tooltip in the following code example is globalized to currency format in the Deutsch culture.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline DataSource="new double[]{ 300.00, 600.00, 400.21, 100.20,
300.70, 200.04, 500.00 }" Height="200px" Width="350px" Format="C">
<SparklineTooltipSettings TValue="double"
Visible="true"></SparklineTooltipSettings>
</SfSparkline>
```

Refer [here](#) to configure localization for the Blazor server application, and [here](#) for the Blazor web assembly application.

On successful configuration, the Sparkline will be rendered as following.

**Events in Blazor Sparkline Component**

This section describes the Sparkline component's events that will be triggered when appropriate actions are performed. The events should be provided to the Sparkline through the [SparklineEvents](#).

Loaded

The **Loaded** event triggers after the Sparkline component has been loaded.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline DataSource="new int[]{ 0, 6, 4, 1, 3, 2, 5 }"
Type="SparklineType.Line" Height="200px" Width="450px">
<SparklineEvents OnLoaded="@LoadedHandler"></SparklineEvents>
</SfSparkline>
@code{
public void LoadedHandler(System.EventArgs args)
{
// Here you can customize the code.
}
}
```

OnPointRendering

The [OnPointRendering](#) event triggers before the point rendering.

Argument name	Description
-----	-----

PointIndex	Specifies the current point index.	
Fill	Specifies the point index color.	
Border	Specifies the color and the width of the point border.	
Cancel	Specifies the event cancel status.	

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline DataSource="new int[]{ 6, 4, 1, 3, 2, 5 }"
Type="SparklineType.Column" Height="200px" Width="450px">
<SparklineMarkerSettings Visible="new List<VisibleType>() { VisibleType.All
}"></SparklineMarkerSettings>
<SparklineEvents OnPointRendering="@PointRenderHandler"></SparklineEvents>
</SfSparkline>
@code{
public void PointRenderHandler(SparklinePointEventArgs args)
{
// Here you can customize the code.
}
}
```

OnPointRegionMouseClicked

The [OnPointRegionMouseClicked](#) event triggers when the mouse click on the point region.

Argument name	Description	
-----	-----	
PointerIndex	Specifies the Sparkline point index region.	
Cancel	Specifies the event cancel status.	

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline DataSource="new int[]{ 6, 4, 1, 3, 2, 5 }"
Type="SparklineType.Column" Height="200px" Width="450px">
<SparklineMarkerSettings Visible="new List<VisibleType>() { VisibleType.All
}"></SparklineMarkerSettings>
<SparklineEvents
OnPointRegionMouseClicked="@PointRegionMouseClickedHandler"></SparklineEvents>
</SfSparkline>
@code{
public void PointRegionMouseClickedHandler(PointRegionEventArgs args)
{
// Here you can customize the code.
}
}
```

OnResizing

The [OnResizing](#) event triggers while resizing the window.

Argument name	Description	
-----	-----	

CurrentSize	Specifies the size of Sparkline.	
PreviousSize	Specifies the previous size of Sparkline.	
Cancel	Specifies the event cancel status.	

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline DataSource="new int[]{ 0, 6, 4, 1, 3, 2, 5 }"
Type="SparklineType.Line" Height="200px" Width="450px">
<SparklineEvents OnResizing="@ResizeHandler"></SparklineEvents>
</SfSparkline>
@code{
public void ResizeHandler(SparklineResizeEventArgs args)
{
// Here you can customize the code.
}
}
```

OnSeriesRendering

The [OnSeriesRendering](#) event triggers before rendering on each Sparkline series.

Argument name	Description	
-----	-----	
Border	Specifies the color and width of the series border.	
Fill	Specifies the series fill color.	
LineWidth	Specifies the series line width.	
Cancel	Specifies the event cancel status.	

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline DataSource="new int[]{ 0, 6, 4, 1, 3, 2, 5 }"
Type="SparklineType.Line" Height="200px" Width="450px">
<SparklineEvents
OnSeriesRendering="@SeriesRenderingHandler"></SparklineEvents>
</SfSparkline>
@code{
public void SeriesRenderingHandler(SeriesRenderingEventArgs args)
{
// Here you can customize the code.
}
}
```

OnMarkerRendering

The [OnMarkerRendering](#) event triggers before rendering the Sparkline marker render.

Argument name	Description	
-----	-----	
Border	Specifies the color and the width of the marker border.	

Fill	Specifies the marker fill color.	
PointIndex	Specifies the marker point index.	
X	Specifies the x axis of the marker.	
Y	Specifies the y axis of the marker.	
Size	Specifies the size of the marker.	
Cancel	Specifies the event cancel status.	

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline DataSource="new int[]{ 6, 4, 1, 3, 2, 5 }"
Type="SparklineType.Line" Height="200px" Width="450px">
<SparklineMarkerSettings Visible="new List<VisibleType>() { VisibleType.All
}"></SparklineMarkerSettings>
<SparklineEvents
OnMarkerRendering="@MarkerRenderingHandler"></SparklineEvents>
</SfSparkline>
@code{
public void MarkerRenderingHandler(MarkerRenderingEventArgs args)
{
// Here you can customize the code.
}
}
```

OnDataLabelRendering

The [OnDataLabelRendering](#) event triggers before rendering the Sparkline data label render.

Argument name	Description	
-----	-----	
Border	Specifies the color and the width of the data label border.	
Fill	Specifies the series fill color of the data label.	
PointIndex	Specifies the data label point index.	
X	Specifies the x axis of the data label.	
Y	Specifies the y axis of the data label.	
Text	Specifies the content of the data label.	
Color	Specifies the content color.	
Cancel	Specifies the event cancel status.	

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfSparkline DataSource="new int[]{ 6, 4, 1, 3, 2, 5 }"
Type="SparklineType.Line" Height="200px" Width="450px">
<SparklineDataLabelSettings Visible="new List<VisibleType>() {
VisibleType.All }"></SparklineDataLabelSettings>
<SparklineEvents
OnDataLabelRendering="@DataLabelRenderingHandler"></SparklineEvents>
```

```
<SparklineAxisSettings MinX="-1" MaxX="7" MaxY="7" MinY="-1"></SparklineAxisSettings>
</SfSparkline>
@code{
public void DataLabelRenderingHandler(DataLabelRenderingEventArgs args)
{
// Here you can customize the code.
}
}
```

Methods in Blazor Sparkline Component

Using the `@ref` property, create an object for the Sparkline component and call the desired method.

Refresh

The `RefreshAsync` method helps to render the Sparkline component again.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<button @onclick="RefreshCall">Refresh</button>
<SfSparkline @ref="@Sparkline" DataSource="new int[]{ 3, 6, 4, 1, 3, 2, 5 }"
Type="SparklineType.Area" Height="200px" Width="350px" Fill="#b2cfff"
LineWidth="1">
</SfSparkline>
@code
{
public SfSparkline<int> Sparkline { get; set; }
public async Task RefreshCall()
{
await Sparkline.RefreshAsync();
}
}
```

Spinner

Getting Started with Blazor Spinner Component

This section briefly explains how to include a [Blazor Spinner](#) component in the Blazor Server-side application. Refer to [Getting Started with Syncfusion Blazor for Server-Side Spinner in Visual Studio 2019 page](#) for the introduction and configuring the common specifications.

Importing Syncfusion Blazor component in the application

- Install **Syncfusion.Blazor.Spinner** NuGet package to the application by using the **NuGet Package Manager**.
- The client-side resources can be added through [CDN](#) or from [NuGet](#) package in the **HEAD** element of the `~/Pages/_Host.cshtml` page.

ASPX-CS

```
<head>
<environment include="Development">
....
....
```

```
<link href="_content/Syncfusion.Blazor/styles/fabric.css" rel="stylesheet" />
<!--CDN-->
@*<link href="https://cdn.syncfusion.com/blazor/18.4.42/styles/fabric.css"
rel="stylesheet" />*@
</environment>
</head>
```

For Internet Explorer 11 kindly refer the polyfills. Refer the [documentation](#) for more information.

ASPX-CS

```
<head>
<environment include="Development">
<link href="_content/Syncfusion.Blazor/styles/fabric.css" rel="stylesheet" />
<script
src="https://github.com/Daddoon/Blazor.Polyfill/releases/download/3.0.1/blazor.polyfill.min.js"></script>
</environment>
</head>
```

Adding component package to the application

Open ~/_Imports.razor file and import the **Syncfusion.Blazor.Spinner** package.

ASPX-CS

```
@using Syncfusion.Blazor.Spinner
```

Add SyncfusionBlazor service in Startup.cs

Open the **Startup.cs** file and add services required by Syncfusion components using **services.AddSyncfusionBlazor()** method. Add this method in the **ConfigureServices** function as follows.

CSHARP

```
using Syncfusion.Blazor;
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void ConfigureServices(IServiceCollection services)
        {
            ....
            ....
            services.AddSyncfusionBlazor();
        }
    }
}
```

Add Spinner component

To initialize the Spinner component, add the below code to the **Index.razor** view page which is present under ~/Pages folder.

The Blazor Spinner component is used to display the loading indication with a specified target area while loading.

Initialization

Import the `Syncfusion.Blazor.Spinner` NuGet package and initialize the Spinner component by adding the spinner as a child of the target element where the spinner needs to be shown.

The following code explains how to initialize a simple Spinner in the Blazor Razor page.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.Spinner
<div>
<SfButton @onclick="@ClickHandler">Show/Hide Spinner</SfButton>
<div id="container">
<SfSpinner @bind-Visible="@VisibleProperty">
</SfSpinner>
</div>
</div>
@code{
private bool VisibleProperty { get; set; } = false;
private async Task ClickHandler()
{
this.VisibleProperty = true;
await Task.Delay(2000);
this.VisibleProperty = false;
}
}
```

Run the application

After successful compilation of the application, simply press **F5** to run the application.



You can also explore our [Blazor Spinner example](#) that shows how to configure the spinner in Blazor.

Customize the Spinner in Blazor Spinner Component

The Spinner component can be customized when initializing or after rendering it.

Customize when initializing the Spinner component

Provided support to change the default Spinner appearance when initializing Spinner component using the following properties.

- `CssClass`
- `Label`
- `Type`
- `Size`

CssClass

Add the customized **Class** name to a Spinner root element to customize the Blazor Spinner component UI styles. The following code explains how to initialize a Spinner with the custom class name in the Blazor Razor page.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.Spinner
<div>
<SfButton @onclick="@ClickHandler">Show/Hide Spinner</SfButton>
<div id="container">
<SfSpinner @bind-Visible="@VisibleProperty" CssClass="e-customClass">
</SfSpinner>
</div>
</div>
@code{
private bool VisibleProperty { get; set; } = false;
private async Task ClickHandler()
{
this.VisibleProperty = true;
await Task.Delay(2000);
this.VisibleProperty = false;
}
}
<style>
.e-spinner-pane.e-customClass .e-spinner-inner .e-spin-material {
stroke: #808080;
}
</style>
```



Modal Spinner

A modal spinner can be initialized by adding the class **e-spin-overlay** to the **CssClass** property of the spinner.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.Spinner
<SfButton @onclick="@ClickHandler">Show/Hide Spinner</SfButton>
<div id="container">
<SfSpinner @bind-Visible="@VisibleProperty" CssClass="e-spin-overlay" />
</div>
<style>
#container {
position: relative;
height: 550px;
}
</style>
@code{
private bool VisibleProperty { get; set; } = false;
```

```
private async Task ClickHandler()
{
    this.VisibleProperty = true;
    await Task.Delay(10000);
    this.VisibleProperty = false;
}
```



Label

Add the customize label text in Blazor Spinner component at the bottom.

The following code explains how to set the **Label** on Spinner in Blazor Razor page.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.Spinner
<div>
    <SfButton @onclick="@ClickHandler">Show/Hide Spinner</SfButton>
    <div id="container">
        <SfSpinner @bind-Visible="@VisibleProperty" Label="Loading....">
        </SfSpinner>
    </div>
</div>
@code{
    private bool VisibleProperty { get; set; } = false;
    private async Task ClickHandler()
    {
        this.VisibleProperty = true;
```

```
await Task.Delay(2000);  
this.VisibleProperty = false;  
}  
}
```



Type

By default, the **Type** is **None** where the Blazor Spinner is loaded based on the theme used in the application. The type can also be customized and shown on Spinner using the **Type** property. The available types are:

- None
- Material
- Fabric
- Bootstrap
- HighContrast
- Bootstrap4

The following code explains how to use the **Type** property when initializing Spinner in Blazor Razor page.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons  
@using Syncfusion.Blazor.Spinner  
<div>  
<SfButton @onclick="@ClickHandler">Show/Hide Spinner</SfButton>  
<div id="container">  
<SfSpinner @bind-Visible="@VisibleProperty" Type="@SpinnerType.Bootstrap">  
</SfSpinner>  
</div>  
</div>  
@code{  
private bool VisibleProperty { get; set; } = false;  
private async Task ClickHandler()  
{  
this.VisibleProperty = true;  
await Task.Delay(2000);  
this.VisibleProperty = false;  
}  
}
```



Size

By default, the Spinner size is 30px. The size of the Spinner can be changed based on the application using the `Size` property.

The following code explains how to use the `Size` property when initializing Spinner in Blazor Razor page.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.Spinner
<div>
<SfButton @onclick="@ClickHandler">Show/Hide Spinner</SfButton>
<div id="container">
<SfSpinner @bind-Visible="@VisibleProperty" Size="50">
</SfSpinner>
</div>
</div>
@code{
private bool VisibleProperty { get; set; } = false;
private async Task ClickHandler()
{
this.VisibleProperty = true;
await Task.Delay(2000);
this.VisibleProperty = false;
}
}
```



Customize after creating the Spinner component

The Spinner component can be customized dynamically after initialize the Spinner component by using the following properties:

- `Type`
- `CssClass`

Type

The type of the Spinner can dynamically be changed using the `Type` property.

The following code explains how to use the `Type` property after creating the Spinner in Blazor Razor page.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.Spinner
<div>
<SfButton @onclick="@ClickHandler">Show/Hide Spinner</SfButton>
<SfButton @onclick="@ChangeType">Change Type</SfButton>
<div id="container">
<SfSpinner @bind-Visible="@VisibleProperty" Type="@SpinnerType">
</SfSpinner>
</div>
</div>
```

```

</SfSpinner>
</div>
</div>
@code{
private SpinnerType SpinnerType = SpinnerType.Fabric;
private bool VisibleProperty { get; set; } = false;
private async Task ClickHandler()
{
this.VisibleProperty = true;
await Task.Delay(2000);
this.VisibleProperty = false;
}
private async Task ChangeType()
{
SpinnerType = SpinnerType.Material;
}
}

```



CssClass

Add the custom class name to Spinner after creating the Spinner component.

The following code explains how to dynamically add the `CssClass` property after creating the Spinner in Blazor Razor page.

ASPX-CS

```

@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.Spinner
<div>
<SfButton @onclick="@ClickHandler">Show/Hide Spinner</SfButton>
<SfButton @onclick="@ChangeClass">Change CSS Class</SfButton>
<div id="container">
<SfSpinner @bind-Visible="@VisibleProperty" CssClass="@CssClassName">
</SfSpinner>
</div>
</div>
@code{
private string CssClassName { get; set; } = "";
private bool VisibleProperty { get; set; } = false;
private async Task ClickHandler()
{
this.VisibleProperty = true;
await Task.Delay(2000);
this.VisibleProperty = false;
}
private async Task ChangeClass()
{
this.CssClassName = "e-customClass";
StateHasChanged();
}
}
<style>

```

```
.e-spinner-pane.e-customClass .e-spinner-inner .e-spin-material {
stroke: #808080;
}
</style>
```



Spinner Integration in Blazor Spinner Component

The Spinner component is rendered with other Blazor components. For example, rendered the Blazor Tab component with Spinner component. To render the spinner inside the Tab component, the Spinner is set as a child of the Tab component and show or hide the Spinner when Tab switching.

ASPX-CS

```
@using Syncfusion.Blazor.Spinner
@using Syncfusion.Blazor.Navigations
<div>
<div id="tab_container">
<SfTab ID="tab">
<TabItems>
<TabItem>
<ChildContent>
<TabHeader Text="Twitter"></TabHeader>
</ChildContent>
<ContentTemplate>
<p>Twitter is an online social networking service that enables users to send
and read short 140-character messages called tweets. Registered users can
read and post tweets, but those who are unregistered can only read them.
Users access Twitter through the website interface, SMS or mobile device app
Twitter Inc. is based in SanFrancisco and has more than 25 offices around
the world. Twitter was created in March 2006 by Jack Dorsey, Evan Williams,
Biz Stone, and Noah Glass and launched in July 2006. The service rapidly
gained worldwide popularity, with more than 100 million users posting 340
million tweets a day in 2012. The service also handled 1.6 billionsearch
queries per day.</p>
</ContentTemplate>
</TabItem>
<TabItem>
<ChildContent>
<TabHeader Text="Facebook"></TabHeader>
</ChildContent>
<ContentTemplate>
<p>
Facebook is an online social networking service headquartered in Menlo Park,
California. Its website was launched on February 4, 2004, by Mark Zuckerberg
with his Harvard College roommates and fellow students EduardoSaverin,
Andrew McCollum, Dustin Moskovitz and Chris Hughes. The founders had
initially limited the website membership to Harvard students, but later
expanded it to colleges in the Boston area, the Ivy League, and Stanford
University. It gradually added support for students at various other
universities and later to high-school students.
</p>
</ContentTemplate>
</TabItem>
```

```

<TabItem>
  <ChildContent>
    <TabHeader Text="Whatsapp"></TabHeader>
  </ChildContent>
  <ContentTemplate>
    <p>WhatsApp Messenger is a proprietary cross-platform instant messaging
    client for smartphones that operates under a subscription business model.
    It uses the Internet to send text messages, images, video, user location
    and audio media messages to other users using standard cellular mobile
    numbers. As of February 2016, WhatsApp had a userbase of up to one
    billion,[10] making it the most globally popular messaging application
    WhatsApp Inc., based in Mountain View, California, was acquired by Facebook
    Inc. on February 19, 2014, for approximately US $19.3 billion.</p>
  </ContentTemplate>
</TabItem>
</TabItems>
<TabEvents Selecting="@Selecting" Selected="@Selected"></TabEvents>
<SfSpinner @bind-Visible="@VisibleProperty">
</SfSpinner>
</SfTab>
</div>
</div>
@code{
private bool VisibleProperty { get; set; } = false;
private async Task Selected(SelectEventArgs args)
{
await Task.Delay(2000);
this.VisibleProperty = false;
}
private void Selecting(SelectingEventArgs args)
{
this.VisibleProperty = true;
}
}
<style>
.e-content .e-item {
font-size: 12px;
padding: 10px;
text-align: justify;
}
.e-tab .e-tab-icon {
font-family: 'Socialicons' !important;
}
.e-tab .e-icons.e-tab-icon {
position: relative;
top: 1px;
}
</style>

```

TWITTER

FACEBOOK

WHATSAPP

Facebook is an online social networking service headquartered in Menlo Park, California. Its website was launched on February 4, 2004, by Mark Zuckerberg with his Harvard College roommates and fellow students Eduardo Saverin, Andrew McCollum, Dustin Moskovitz and Chris Hughes. The founders had initially limited the website membership to Harvard students, but later expanded it to colleges in the Boston area, the Ivy League, and Stanford University. It gradually added support for students at various other universities and later to high-school students.

Style and appearance in Blazor Spinner 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 spinner

Use the following CSS to customize the spinner stroke color.

Material theme

CSS

```
.e-spinner-pane .e-spinner-inner .e-spin-material {
  stroke: green;
}
```

Fabric theme

CSS

```
.e-spinner-pane .e-spinner-inner .e-spin-fabric {
  stroke: green;
}
```

Bootstrap theme

CSS

```
.e-spinner-pane .e-spinner-inner .e-spin-bootstrap {
  fill: green;
  stroke: green;
}
```

Bootstrap4 theme

CSS

```
.e-spinner-pane .e-spinner-inner .e-spin-bootstrap4 {
  stroke: green;
}
```

High Contrast theme

CSS

```
.e-spinner-pane .e-spinner-inner .e-spin-high-contrast .e-path-arc {
  stroke: green;
}
```



```
}
```

Events in Blazor Spinner Component

This section explains the list of events of the Spinner component which will be triggered for appropriate Spinner actions.

Created

Created event triggers after the Spinner is created.

ASPX-CS

```
@using Syncfusion.Blazor.Spinner
<SfSpinner>
  <SpinnerEvents Created="@CreatedHandler" >/SpinnerEvents>
</SfSpinner>
@code{
public void CreatedHandler(Object args)
{
  // Here you can customize your code
}
}
```

OnBeforeOpen

OnBeforeOpen event triggers before the Spinner is opened.

ASPX-CS

```
@using Syncfusion.Blazor.Spinner
<SfSpinner>
  <SpinnerEvents OnBeforeOpen="@OnBeforeOpenHandler" >/SpinnerEvents>
</SfSpinner>
@code{
public void OnBeforeOpenHandler(SpinnerEventArgs args)
{
  // Here you can customize your code
}
}
```

OnBeforeClose

OnBeforeClose event triggers before the Spinner is closed.

ASPX-CS

```
@using Syncfusion.Blazor.Spinner
<SfSpinner>
  <SpinnerEvents OnBeforeClose="@OnBeforeCloseHandler" >/SpinnerEvents>
</SfSpinner>
@code{
public void OnBeforeCloseHandler(SpinnerEventArgs args)
{
  // Here you can customize your code
}
}
```

Destroyed

Destroyed event triggers after the Spinner is destroyed.

ASPX-CS

```
@using Syncfusion.Blazor.Spinner
<SfSpinner>
  <SpinnerEvents Destroyed="@DestroyedHandler" >/SpinnerEvents>
</SfSpinner>
@code{
public void DestroyedHandler(Object args)
{
  // Here you can customize your code
}
}
```

SplitButton

<!-- markdownlint-disable MD024 -->

Getting Started with Blazor SplitButton Component

This section briefly explains about how to include Split Button Component in the Blazor server-side application. Refer [Getting Started with Syncfusion Blazor for Server-side in Visual Studio 2019 page](#) page for the introduction and configuring the common specifications.

Importing Syncfusion Blazor component in the application

1. Install the **Syncfusion.Blazor** NuGet package to the application by using the **NuGet Package Manager**.
2. The client-side style resources can be added through [CDN](#) or from [NuGet](#) package in the element of the `~/Pages/_Host.cshtml` page.

Please ensure to check the **Include prerelease** option.

HTML

```
<head>
<link href="_content/Syncfusion.Blazor/styles/bootstrap4.css"
rel="stylesheet" />
@*<link href="https://cdn.syncfusion.com/blazor/{ site.blazorversion
}}/styles/bootstrap4.css" rel="stylesheet" />*@
</head>
```

For Internet Explorer 11 kindly refer the polyfills. Refer the [documentation](#) for more information.

HTML

```
<head>
<link href="_content/Syncfusion.Blazor/styles/bootstrap4.css"
rel="stylesheet" />
<script
src="https://github.com/Daddoon/Blazor.Polyfill/releases/download/3.0.1/blaz
or.polyfill.min.js"></script>
</head>
```

Adding component package to the application

Open `/_Imports.razor` file and import the **Syncfusion.Blazor.Buttons** package.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
```

Add SyncfusionBlazor service in Startup.cs

Open the **Startup.cs** file and add services required by Syncfusion components. Add **services.AddSyncfusionBlazor()** method in the **ConfigureServices** function as follows.

CSHARP

```
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void ConfigureServices(IServiceCollection services)
        {
            ....
            ....
            services.AddSyncfusionBlazor();
        }
    }
}
```

To enable custom client side resource loading from CRG or CDN. Disable the resource loading by **AddSyncfusionBlazor(true)** and load the scripts in the HEAD element of the `~/Pages/_Host.cshtml` page.

ASPX-CS

```
<head>
<environment include="Development">
<script src="https://cdn.syncfusion.com/blazor/{{ site.blazorversion
}}/syncfusion-blazor.min.js">
</script>
</environment>
</head>
```

Adding Split Button component to the application

Now, add the Syncfusion Blazor Split Button component in `razor` page in the `Pages` folder. For example, the Split Button component is added in the `~/Pages/Index.razor` page.

ASPX-CS

```
<SfSplitButton Content="Paste">
<DropDownMenuItems>
<DropDownMenuItem Text="Cut" ></DropDownMenuItem>
<DropDownMenuItem Text="Copy" ></DropDownMenuItem>
<DropDownMenuItem Text="Paste"></DropDownMenuItem>
</DropDownMenuItems>
```

```
</SfSplitButton>
```

Run the application

After successful compilation of the application, simply press F5 to run the application. The Blazor Split Button component will render in the web browser.



See Also

- [Getting Started with Syncfusion Blazor for Client-Side in .NET Core CLI](#)
- [Getting Started with Syncfusion Blazor for Server-Side in Visual Studio 2019 Preview](#)
- [Getting Started with Syncfusion Blazor for Server-Side in .NET Core CLI](#)

Events in Blazor SplitButton Component

The split button event can be defined using on [SplitButtonEvents](#) in component. The value of event is treated as an event handler. The event specific data will be available in event arguments.

List of events supported

The following event support is provided to the SplitButton component. The different event argument types for each event are,

- OnOpen - BeforeOpenCloseMenuEventArgs
- Opened - OpenCloseMenuEventArgs
- ItemSelected - MenuEventArgs
- OnClose – BeforeOpenCloseMenuEventArgs
- OnItemRender – MenuEventArgs
- Closed – OpenCloseMenuEventArgs

How to bind event to Split Button

Above defined events bind the Split Button component. Here, we have explained about the sample code snippets of SplitButton.

ASPX-CS

```
@using Syncfusion.Blazor.SplitButtons
<SfSplitButton Content="Profile">
  <SplitButtonEvents Created="Created" Clicked="Clicked" OnOpen="OnOpen"
    Opened="Opened" ItemSelected="ItemSelected" OnClose="OnClose"
    OnItemRender="ItemRender" Closed="Closed">
```

```

</SplitButtonEvents>
<DropDownMenuItems>
<DropDownMenuItem Text="Cut"></DropDownMenuItem>
<DropDownMenuItem Text="Copy"></DropDownMenuItem>
<DropDownMenuItem Text="Paste"></DropDownMenuItem>
</DropDownMenuItems>
</SfSplitButton>
@code {
private void Created()
{
}
private void OnOpen(BeforeOpenCloseMenuEventArgs args)
{
}
private void Opened(OpenCloseMenuEventArgs args)
{
}
private void ItemSelected(MenuEventArgs args)
{
}
private void OnClose(BeforeOpenCloseMenuEventArgs args)
{
}
private void ItemRender(MenuEventArgs args)
{
}
private void Closed(OpenCloseMenuEventArgs args)
{
}
private void Clicked(Syncfusion.Blazor.SplitButtons.ClickEventArgs args)
{
}
}

```

Icons And Separator in Blazor SplitButton Component

Split Button Icons

Split Button can have an icon to provide the visual representation of the action. To place the icon on a Split 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 Split Button. The icon's position can be customized by using the [IconPosition](#) property.

ASPX-CS

```

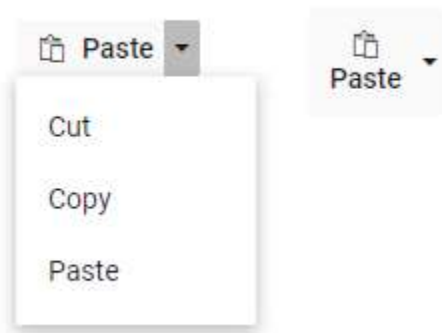
@using Syncfusion.Blazor.SplitButtons
<SfSplitButton Content="Paste" IconCss="e-icons e-paste">
<DropDownMenuItems>
<DropDownMenuItem Text="Cut"></DropDownMenuItem>
<DropDownMenuItem Text="Copy"></DropDownMenuItem>
<DropDownMenuItem Text="Paste"></DropDownMenuItem>
</DropDownMenuItems>
</SfSplitButton>
<SfSplitButton Content="Paste" IconCss="e-icons e-paste"
IconPosition="SplitButtonIconPosition.Top">
<DropDownMenuItems>
<DropDownMenuItem Text="Cut"></DropDownMenuItem>

```

```

<DropDownMenuItem Text="Copy"></DropDownMenuItem>
<DropDownMenuItem Text="Paste"></DropDownMenuItem>
</DropDownMenuItems>
</SfSplitButton>
<style>
.e-paste::before {
content: '\e739';
}
</style>

```



The third party icons on the Split Button can be used by the [IconCss](#) property.

Vertical Button

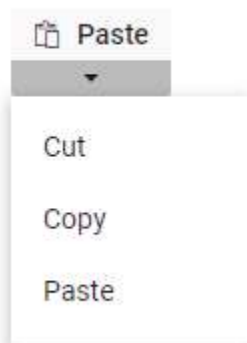
Vertical Button in Split Button can be achieved by adding `e-vertical` class using [CssClass](#) property.

ASPX-CS

```

@using Syncfusion.Blazor.SplitButtons
<SfSplitButton Content="Paste" IconCss="e-icons e-paste" CssClass="e-
vertical">
<DropDownMenuItems>
<DropDownMenuItem Text="Cut" ></DropDownMenuItem>
<DropDownMenuItem Text="Copy" ></DropDownMenuItem>
<DropDownMenuItem Text="Paste"></DropDownMenuItem>
</DropDownMenuItems>
</SfSplitButton>
<style>
.e-paste::before {
content: '\e739';
}
</style>

```



Separator

The Separators are the horizontal lines that are used to separate the popup items. You cannot select the separators. You can enable separators to group the popup items using the separator property.

The following example illustrates how to enable [Separator](#) support in Split Button component.

ASPX-CS

```
@using Syncfusion.Blazor.SplitButtons
<SfSplitButton Content="Paste" IconCss="e-icons e-paste" CssClass="e-
vertical">
  <DropDownMenuItems>
    <DropDownMenuItem Text="Cut"></DropDownMenuItem>
    <DropDownMenuItem Text="Copy"></DropDownMenuItem>
    <DropDownMenuItem Separator="true"></DropDownMenuItem>
    <DropDownMenuItem Text="Paste"></DropDownMenuItem>
  </DropDownMenuItems>
</SfSplitButton>
<style>
.e-paste::before {
content: '\e739';
}
</style>
```



See Also

- [Split Button popup with icons](#)

Popup Items in Blazor SplitButton Component

Icons

The Popup action item have 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 Cut, Copy, Paste menu items are added using the IconCss property.

ASPX-CS

```
@using Syncfusion.Blazor.SplitButtons
<SfSplitButton Content="Paste" IconCss="e-icons e-paste">
  <DropDownMenuItems>
    <DropDownMenuItem IconCss="e-icons e-cut" Text="Cut"></DropDownMenuItem>
    <DropDownMenuItem IconCss="e-icons e-copy" Text="Copy"></DropDownMenuItem>
    <DropDownMenuItem IconCss="e-icons e-paste" Text="Paste"></DropDownMenuItem>
  </DropDownMenuItems>
</SfSplitButton>
<style>
.e-paste::before {
content: '\e739';
}
.e-cut::before {
content: '\e73f';
}
.e-copy::before {
content: '\e77b';
}
</style>
```



Template

Item Templating

Popup items can be customized using the `CssClass` property. Customize the items using CSS style.

ASPX-CS

```
@using Syncfusion.Blazor.SplitButtons
<SfSplitButton Content="Edit" CssClass="custom">
  <DropDownMenuItems>
    <DropDownMenuItem Text="Cut"></DropDownMenuItem>
    <DropDownMenuItem Text="Copy"></DropDownMenuItem>
    <DropDownMenuItem Text="Paste"></DropDownMenuItem>
  </DropDownMenuItems>
</SfSplitButton>
<style>
.custom li {
float: left;
font-size: 10px;
padding-left: 50px;
font-style: oblique;
}
</style>
```



Accessibility in Blazor SplitButton Component

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.

Split Button provides built-in compliance with WAI-ARIA specifications. WAI-ARIA support is achieved through the attributes like aria-expanded, aria-owns and aria-haspopup applied for action item in

Split Button. It helps the people with disabilities by providing information about the widget for assistive technology in the screen readers. Split Button component contains the MenuItem role.

| Properties | Functionality |

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

| menuItem | This role will be specified for an action items. |

| aria-haspopup | Indicates the availability and type of interactive SplitButton popup element. |

| aria-expanded | Indicates whether the SplitButton 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. |

Keyboard interaction

<!-- markdownlint-disable MD033 -->

Keyboard shortcuts	Actions
Esc	Closes the opened 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.
Down	Navigates down or to the next action item.
Alt + Up Arrow	Opens the popup.
Alt + Down Arrow	Closes the popup.

ASPX-CS

```
@using Syncfusion.Blazor.SplitButtons
<SfSplitButton Content="Paste">
  <DropDownMenuItems>
    <DropDownMenuItem Text="Cut" ></DropDownMenuItem>
    <DropDownMenuItem Text="Copy" ></DropDownMenuItem>
    <DropDownMenuItem Text="Paste"></DropDownMenuItem>
  </DropDownMenuItems>
</SfSplitButton>
```



Styles and Appearances in Blazor SplitButton Component

To modify the SplitButton appearance, override the default CSS of SplitButton component. Please find the list of CSS classes and its corresponding section in SplitButton. Also, there is an option to create own custom theme for the controls using the [Theme Studio](#).

| CSS Class | Purpose of Class |

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

| .e-dropdown-btn | To customize the button part in splitbutton. |

| .e-split-btn | To customize the dropdown part in split button. |

| .e-dropdown-popup ul .e-item | To customize the progress button list items. |

| .e-dropdown-popup ul .e-item .e-menu-icon | To customize the progress button list items icon. |

How To

Create right-to-left Blazor SplitButton Component

Split 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 Split Button component.

ASPX-CS

```
@using Syncfusion.Blazor.SplitButtons
<SfSplitButton Content="Paste" IconCss="e-icons e-paste" EnableRtl="true">
  <DropDownMenuItems>
    <DropDownMenuItem IconCss="e-icons e-cut" Text="Cut"></DropDownMenuItem>
    <DropDownMenuItem IconCss="e-icons e-copy" Text="Copy"></DropDownMenuItem>
    <DropDownMenuItem IconCss="e-icons e-paste" Text="Paste"></DropDownMenuItem>
  </DropDownMenuItems>
</SfSplitButton>
<style>
.e-paste::before {
content: '\e739';
}
.e-cut::before {
content: '\e73f';
}
.e-copy::before {
content: '\e77b';
}
</style>
```



Group items in popup in Blazor SplitButton Component

Items in popup can be grouped in Split Button by templating entire popup with `ListView`. To achieve grouping in `ListView`, check [ListView Grouping](#) documentation. To template `ListView` in popup, render the `ListView` Component in Split button popup using `PopupContent` property.

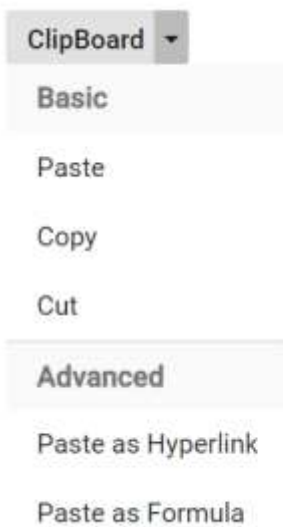
ASPX-CS

```
@using Syncfusion.Blazor.SplitButtons
@using Syncfusion.Blazor.Lists
<SfSplitButton Content="Clipboard">
```

```

<PopupContent>
<SfListView ID="listview" DataSource="@Data"
SortOrder="Syncfusion.Blazor.Lists.SortOrder.Descending" TValue="ListData">
<ListViewFieldSettings Text="Text" GroupBy="Category"
TValue="ListData"></ListViewFieldSettings>
</SfListView>
</PopupContent>
<ChildContent>
<SplitButtonEvents OnClose="popupClose"></SplitButtonEvents>
</ChildContent>
</SfSplitButton>
@code {
private void popupClose(BeforeOpenCloseMenuEventArgs args)
{
}
public List<ListData> Data = new List<ListData>{
new ListData{ Text = "Cut", Category = "Basic" },
new ListData{ Text = "Copy", Category = "Basic" },
new ListData{ Text = "Paste", Category = "Basic" },
new ListData{ Text = "Paste as Formula", Category = "Advanced" },
new ListData{ Text = "Paste as Hyperlink", Category = "Advanced" }
};
public class ListData
{
public string Text { get; set; }
public string Category { get; set; }
}
}

```



Open a dialog on popup item click in Blazor SplitButton Component

This section explains about how to open a dialog on Split Button popup item click. This can be achieved by handling dialog open in [ItemSelected](#) event of the Split Button.

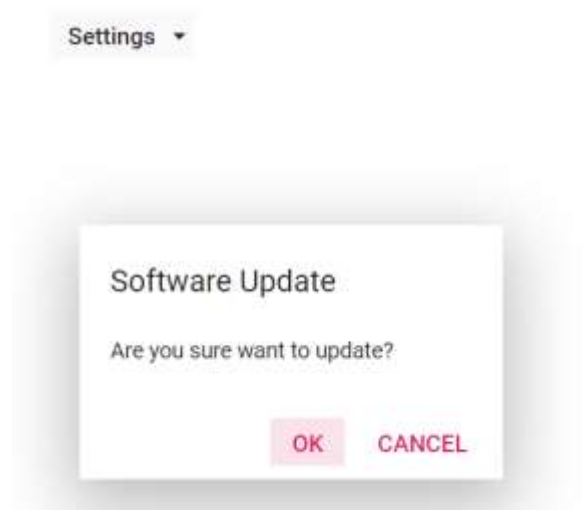
In the following example, Dialog will open while selecting **Update** item.

ASPX-CS

```

@using Syncfusion.Blazor.SplitButtons
@using Syncfusion.Blazor.Popups
<SfSplitButton Content="Settings">
<SplitButtonEvents ItemSelected="select"></SplitButtonEvents>
<DropDownMenuItems>
<DropDownMenuItem Text="Help"></DropDownMenuItem>
<DropDownMenuItem Text="About"></DropDownMenuItem>
<DropDownMenuItem Text="Update"></DropDownMenuItem>
</DropDownMenuItems>
</SfSplitButton>
<SfDialog Content="@Content" Header="@Header" Width="250px" Height="150px"
Visible="false" @ref="DialogObj">
<DialogPositionData X="300" Y="200"></DialogPositionData>
<DialogButtons>
<DialogButton OnClick="@click">
<DialogButton Content="OK" IsPrimary="true"></DialogButton>
</DialogButton>
<DialogButton OnClick="@click">
<DialogButton Content="Cancel"></DialogButton>
</DialogButton>
</DialogButtons>
</SfDialog>
@code {
SfDialog DialogObj;
public string Content = "Are you sure want to update?";
public string Header = "Software Update";
private void click(object args)
{
DialogObj.Hide();
}
private void select(MenuEventArgs args)
{
if (args.Item.Text == "Update")
{
DialogObj.Show();
}
}
}
<style>
.e-setting-icon::before {
content: '\e679';
}
</style>

```

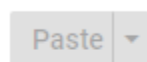


Set the disabled state in Blazor SplitButton Component

Split Button component can be enabled or disabled by disabled property. To disable Split Button component, set the [Disabled](#) property as true.

ASPX-CS

```
@using Syncfusion.Blazor.SplitButtons
<SfSplitButton Content="Paste" Disabled="true">
  <DropDownMenuItems>
    <DropDownMenuItem Text="Cut" ></DropDownMenuItem>
    <DropDownMenuItem Text="Copy" ></DropDownMenuItem>
    <DropDownMenuItem Text="Paste"></DropDownMenuItem>
  </DropDownMenuItems>
</SfSplitButton>
```



Add and Remove Items in Blazor SplitButton Component

Split Button component can dynamically add or remove items using `AddItems`, `RemoveItems` method.

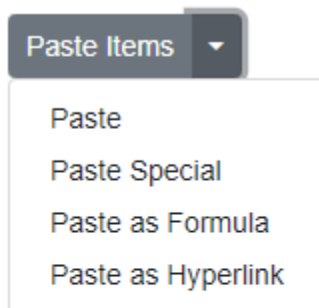
ASPX-CS

```
@using Syncfusion.Blazor.SplitButtons
@using Syncfusion.Blazor.Buttons
<SfSplitButton Content="Paste Items" @ref="SplitbuttonRef">
  <DropDownMenuItems>
    <DropDownMenuItem Text="Paste"></DropDownMenuItem>
  </DropDownMenuItems>
```

```

</SfSplitButton>
<div>
<SfButton Content="Additem" IsPrimary="true" @onclick="addItem"></SfButton>
<SfButton Content="Removeitem" IsPrimary="true"
@onclick="removeItem"></SfButton>
</div>
@code {
SfSplitButton SplitbuttonRef;
private void addItem()
{
SplitbuttonRef.AddItem(SplitbtnItems);
}
private void removeItem()
{
SplitbuttonRef.RemoveItems(removeItems);
}
public List<DropDownMenuItem> SplitbtnItems = new List<DropDownMenuItem>
{
new DropDownMenuItem{ Text="Paste Special" },
new DropDownMenuItem{ Text="Paste as Formula" },
new DropDownMenuItem{ Text="Paste as Hyperlink" }
};
public List<string> removeItems = new List<string>()
{
"Paste"
};
}

```



Splitter

<!-- markdownlint-disable MD024 -->

Getting Started with Blazor Splitter Component

This section briefly explains how to include a Splitter component in the Blazor Server-side application. Refer to Getting Started with [Syncfusion Blazor for Server-Side in Visual Studio 2019 page](#) for the introduction and configuring the common specifications.

Importing Syncfusion Blazor component in the application

- Install **Syncfusion.Blazor.Layouts** NuGet package to the application by using the **NuGet Package Manager**.
- The client-side resources can be added through [CDN](#) or from [NuGet](#) package in the **HEAD** element of the `~/Pages/_Host.cshtml` page.

ASPX-CS

```
<head>
<environment include="Development">
    ....
    ....
<link href="_content/Syncfusion.Blazor/styles/fabric.css" rel="stylesheet"
/>
<!--CDN-->
@*<link href="https://cdn.syncfusion.com/blazor/18.4.42/styles/fabric.css"
rel="stylesheet" />*@
</environment>
</head>
```

For Internet Explorer 11 kindly refer the polyfills. Refer the [documentation](#) for more information.

ASPX-CS

```
<head>
<environment include="Development">
<link href="_content/Syncfusion.Blazor/styles/fabric.css" rel="stylesheet"
/>
<script
src="https://github.com/Daddoon/Blazor.Polyfill/releases/download/3.0.1/blaz
or.polyfill.min.js"></script>
</environment>
</head>
```

Adding component package to the application

Open `~/_Imports.razor` file and import the **Syncfusion.Blazor.Layouts** package.

ASPX-CS

```
@using Syncfusion.Blazor.Layouts
```

Add SyncfusionBlazor service in Startup.cs

Open the **Startup.cs** file and add services required by Syncfusion components using **services.AddSyncfusionBlazor()** method. Add this method in the **ConfigureServices** function as follows.

CSHARP

```
using Syncfusion.Blazor;
namespace BlazorApplication
{
    public class Startup
    {
        ....
    }
}
```



```
.....  
public void ConfigureServices(IServiceCollection services)  
{  
    .....  
    .....  
    services.AddSyncfusionBlazor();  
}  
}  
}
```

Add Splitter component

To initialize the Splitter component, add the below code to **Index.razor** view page which is present under **~/Pages** folder.

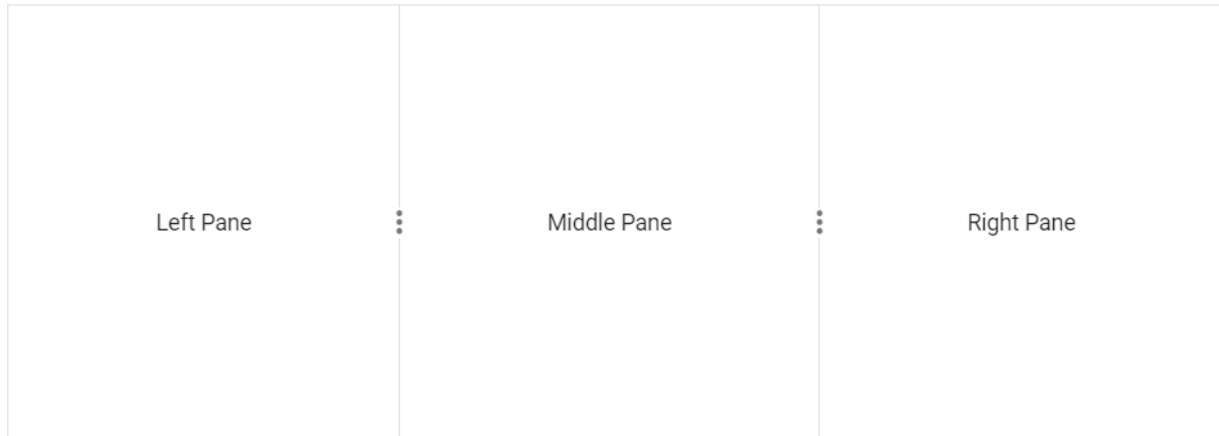
The following code explains how to initialize a simple horizontal Splitter in the Razor page.

ASPX-CS

```
@using Syncfusion.Blazor.Layouts  
<div>Horizontal Splitter</div>  
<SfSplitter Height="240px" Width="100%">  
    <SplitterPanels>  
        <SplitterPanel>  
            <ContentTemplate>  
                <div> Left Pane </div>  
            </ContentTemplate>  
        </SplitterPanel>  
        <SplitterPanel>  
            <ContentTemplate>  
                <div> Middle Pane </div>  
            </ContentTemplate>  
        </SplitterPanel>  
        <SplitterPanel>  
            <ContentTemplate>  
                <div> Right Pane </div>  
            </ContentTemplate>  
        </SplitterPanel>  
    </SplitterPanels>  
</SfSplitter>
```

Run the application

After successful compilation of the application, run the application.



See Also

- [Getting Started with Syncfusion Blazor for client-side in .NET Core CLI](#)
- [Getting Started with Syncfusion Blazor for server-side in Visual Studio 2019](#)
- [Getting Started with Syncfusion Blazor for server-side in .NET Core CLI](#)

Pane Sizing in Blazor Splitter Component

Splitter allows to provide pane sizes either in **Pixel** or **Percentage** formats.

Auto size panes

The splitter's panes are adjusted automatically during resizing if the size is not specified externally to panes, because the panes are designed based on flex layout by default. When you add/remove or show/hide the panes, the panes are auto aligned within its container.

ASPX-CS

```
using Syncfusion.Blazor.Layouts
<SfSplitter Height="200px" Width="600px">
  <SplitterPanels>
    <SplitterPanel>
      <ContentTemplate>
        <div>
          <div class="content">
            <h3 class="h3">Grid </h3>
```

The ASP.NET DataGrid control, or DataTable is a feature-rich control used to display data in a tabular format.

```
</div>
        </div>
      </ContentTemplate>
    </SplitterPanel>
    <SplitterPanel>
      <ContentTemplate>
        <div>
          <div class="content">
            <h3 class="h3">Schedule </h3>
            The ASP.NET Scheduler, a.k.a. event calendar, facilitates almost all
            calendar features, thus allowing users to manage their time efficiently
          </div>
        </div>
```

```

</ContentTemplate>
</SplitterPane>
<SplitterPane>
<ContentTemplate>
<div>
<div class="content">
<h3 class="h3">Chart </h3>
ASP.NET charts, a well-crafted easy-to-use charting package, is used to add
beautiful charts in web and mobile applications
</div>
</div>
</ContentTemplate>
</SplitterPane>
</SplitterPanels>
</SfSplitter>
<style>
.content {
padding: 10px;
}
</style>

```

Grid	Schedule	Chart
The ASP.NET DataGrid control, or DataTable is a feature-rich control used to display data in a tabular format.	The ASP.NET Scheduler, a.k.a. event calendar, facilitates almost all calendar features, thus allowing users to manage their time efficiently	ASP.NET charts, a well-crafted easy-to-use charting package, is used to add beautiful charts in web and mobile applications

Fixed pane

The split panes can be rendered with fixed size in both **Horizontal** and **Vertical** mode. Even if a fixed size is provided to all the panes, the last pane is considered as a flexible pane.

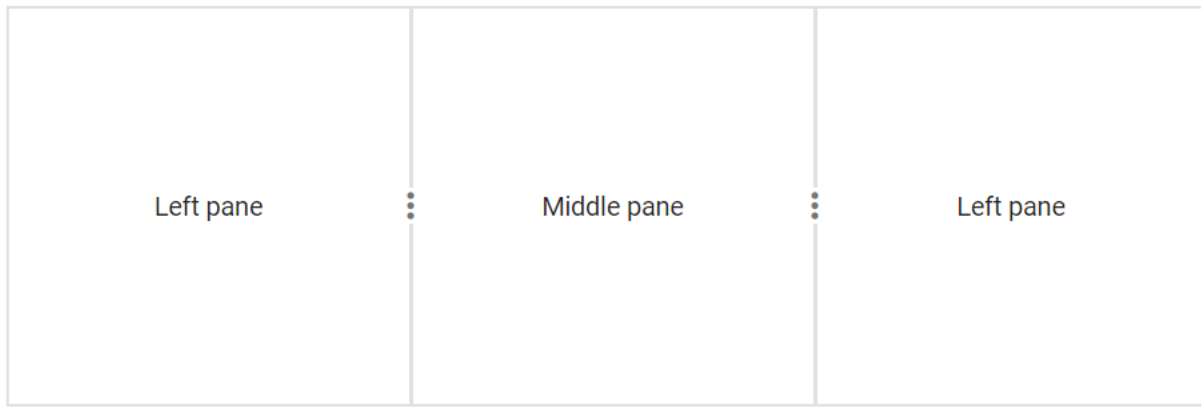
ASPX-CS

```

@using Syncfusion.Blazor.Layouts
<SfSplitter Height="200px" Width="600px">
<SplitterPanels>
<SplitterPane Size="200px">
<ContentTemplate>
<div> Left pane </div>
</ContentTemplate>
</SplitterPane>
<SplitterPane Size="200px">
<ContentTemplate>
<div> Middle pane </div>
</ContentTemplate>
</SplitterPane>
<SplitterPane Size="200px">

```

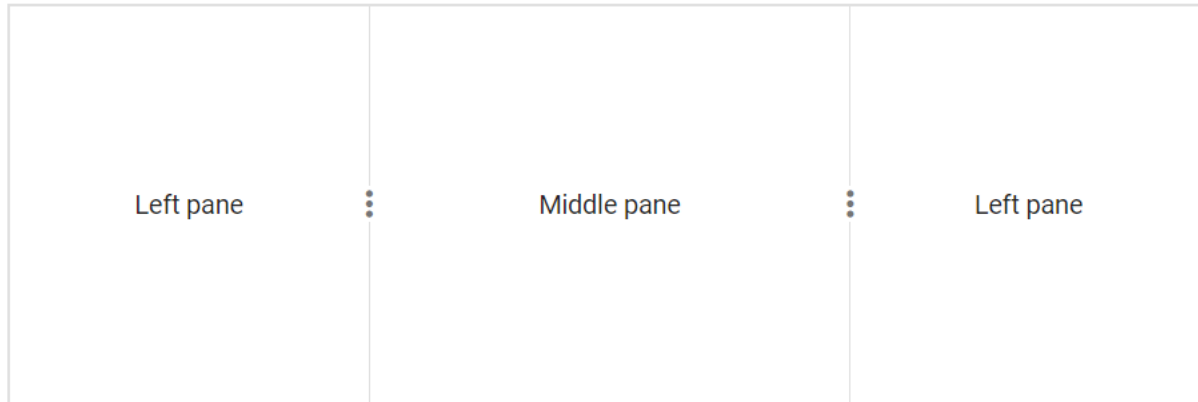
```
<ContentTemplate>
<div> Right pane </div>
</ContentTemplate>
</SplitterPane>
</SplitterPanes>
</SfSplitter>
<style>
.e-pane {
text-align: center;
align-items: center;
display: grid;
}
</style>
```



Splitter pane Size in **Percentage**.

ASPX-CS

```
@using Syncfusion.Blazor.Layouts
<SfSplitter Height="200px" Width="600px">
<SplitterPanes>
<SplitterPane Size="30%">
<ContentTemplate>
<div> Left pane </div>
</ContentTemplate>
</SplitterPane>
<SplitterPane Size="40%">
<ContentTemplate>
<div> Middle pane </div>
</ContentTemplate>
</SplitterPane>
<SplitterPane Size="30%">
<ContentTemplate>
<div> Right pane </div>
</ContentTemplate>
</SplitterPane>
</SplitterPanes>
</SfSplitter>
```



Pane Content in Blazor Splitter Component

This section explains how to provide plain text content or HTML markup to splitter pane.

HTML Markup

Splitter is a layout based container component. The pane contents can be rendered from existing HTML markups. Converting HTML markup as splitter pane is easy way to add the panes content dynamically.

ASPX-CS

```
@using Syncfusion.Blazor.Layouts
<SfSplitter Height="200px" Width="600px">
  <SplitterPan>
    <SplitterPane Content="@Content1" Size="200px"></SplitterPane>
    <SplitterPane Content="@Content2" Size="200px"></SplitterPane>
    <SplitterPane Content="@Content3" Size="200px"></SplitterPane>
  </SplitterPan>
</SfSplitter>
<style>
.content {
padding: 10px;
}
</style>
@code {
private string Content1 = "<div class='content'>" +
"<h3 class='h3'>Grid </h3>" +
"The ASP.NET DataGrid control, or DataTable is a feature-rich control used
to display data in a tabular format.</div>";
private string Content2 = "<div class='content'>" +
"<h3 class='h3'>Schedule </h3>" +
"The ASP.NET Scheduler, a.k.a. event calendar, facilitates almost all
calendar features, thus allowing users to manage their time efficiently
</div>";
private string Content3 = "<div class='content'>" +
"<h3 class='h3'>Chart </h3>" +
"ASP.NET charts, a well-crafted easy-to-use charting package, is used to add
beautiful charts in web and mobile applications </div>";
}
```

<p>Grid</p> <p>The ASP.NET DataGrid control, or DataTable is a feature-rich control used to display data in a tabular format.</p>	<p>Schedule</p> <p>The ASP.NET Scheduler, a.k.a. event calendar, facilitates almost all calendar features, thus allowing users to manage their time efficiently</p>	<p>Chart</p> <p>ASP.NET charts, a well-crafted easy-to-use charting package, is used to add beautiful charts in web and mobile applications</p>
--	--	--

Blazor UI components

Any Blazor components can be rendered along with their native and control events within splitter as pane content.

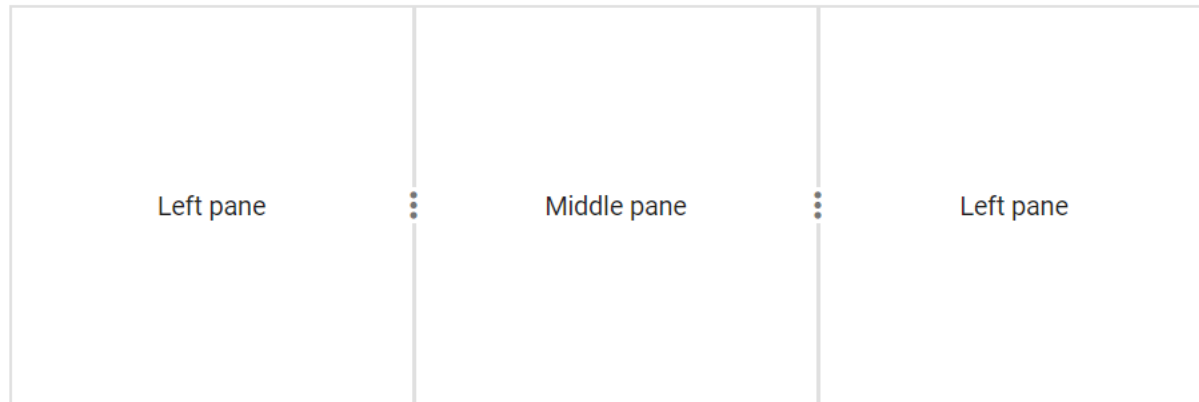
Refer [Listview within splitter](#) example.

Plain content

The plain text can be added as a pane contents using either inner HTML or **Content** API

ASPX-CS

```
@using Syncfusion.Blazor.Layouts
<SfSplitter Height="200px" Width="600px">
  <SplitterPanels>
    <SplitterPane Content="@Content1" Size="200px"></SplitterPane>
    <SplitterPane Content="@Content2" Size="200px"></SplitterPane>
    <SplitterPane Content="@Content3" Size="200px"></SplitterPane>
  </SplitterPanels>
</SfSplitter>
<style>
.e-pane {
text-align: center;
align-items: center;
display: grid;
}
</style>
@code {
private string Content1 = "Left pane";
private string Content2 = "Middle pane";
private string Content3 = "Left pane";
}
```



Integrate other Blazor component inside the pane of the Splitter

Another Blazor component can be rendered inside the split pane using following solutions:

Solution 1

The Blazor component can be directly rendered as content to split pane.

Solution 2

The Blazor component can be integrated by rendering it as a separate page. (Each page is considered as a separate component in Blazor.)

The following example demonstrates the above two solutions. The first pane renders Grid as a direct child component, and the second pane renders Tab component from a separate page(SplitterContent.razor).

Index.razor

ASPX-CS

```
@using Syncfusion.Blazor.Grids
@using Syncfusion.Blazor.Layouts
<SfSplitter Height="220px" Width="100%" SeparatorSize="4">
  <SplitterPaness>
    <SplitterPane Size="50%" Min="300px">
      <SfGrid DataSource="@Orders">
        <GridColumn>
          <GridColumn Field=@nameof(Order.OrderID) HeaderText="Order ID"
            TextAlign="TextAlign.Right" Width="120"></GridColumn>
          <GridColumn Field=@nameof(Order.CustomerID) HeaderText="Customer Name"
            Width="150"></GridColumn>
          <GridColumn Field=@nameof(Order.OrderDate) HeaderText=" Order Date"
            Format="yMd" Type="ColumnType.Date" TextAlign="TextAlign.Right"
            Width="130"></GridColumn>
        </GridColumn>
      </SfGrid>
    </SplitterPane>
    <SplitterPane Size="50%" Min="300px">
      <BlazorServer.Pages.Splitter.UGSamples.PaneContent.SplitterContent></BlazorS
        erver.Pages.Splitter.UGSamples.PaneContent.SplitterContent>
    </SplitterPane>
  </SplitterPaness>
</SfSplitter>
@code{
```

```
private List<Order> Orders { get; set; }
protected override void OnInitialized()
{
    Orders = Enumerable.Range(1, 10).Select(x => new Order()
    {
        OrderID = 1000 + x,
        CustomerID = (new string[] { "ALFKI", "ANANTR", "ANTON", "BLONP", "BOLID"
    })[new Random().Next(5)],
        Freight = 2.1 * x,
        OrderDate = DateTime.Now.AddDays(-x),
    }).ToList();
}
public class Order
{
    public int? OrderID { get; set; }
    public string CustomerID { get; set; }
    public DateTime? OrderDate { get; set; }
    public double? Freight { get; set; }
}
}
```

Other SplitterContent.razor page.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<div>
<SfTab ID="tab">
<TabItems>
<TabItem>
<ContentTemplate>
<TabHeader Text="Twitter"></TabHeader>
</ContentTemplate>
<ContentTemplate>
<p>Twitter is an online social networking service that enables users to send
and read short 140-character messages called tweets. Registered users can
read and post tweets, but those who are unregistered can only read them.
Users access Twitter through the website interface, SMS or mobile device app
Twitter Inc. is based in SanFrancisco and has more than 25 offices around
the world. Twitter was created in March 2006 by Jack Dorsey, Evan Williams,
Biz Stone, and Noah Glass and launched in July 2006. The service rapidly
gained worldwide popularity, with more than 100 million users posting 340
million tweets a day in 2012. The service also handled 1.6 billionsearch
queries per day.</p>
</ContentTemplate>
</TabItem>
<TabItem>
<ContentTemplate>
<TabHeader Text="Facebook"></TabHeader>
</ContentTemplate>
<ContentTemplate>
<p>
Facebook is an online social networking service headquartered in Menlo Park,
California. Its website was launched on February 4, 2004, by Mark Zuckerberg
with his Harvard College roommates and fellow students EduardoSaverin,
Andrew McCollum, Dustin Moskovitz and Chris Hughes. The founders had
```


initially limited the website membership to Harvard students, but later expanded it to colleges in the Boston area, the Ivy League, and Stanford University. It gradually added support for students at various other universities and later to high-school students.

```
</p>
</ContentTemplate>
</TabItem>
<TabItem>
<ContentTemplate>
<TabHeader Text="Whatsapp"></TabHeader>
</ContentTemplate>
<ContentTemplate>
<p>WhatsApp Messenger is a proprietary cross-platform instant messaging client for smartphones that operates under a subscription business model. It uses the Internet to send text messages, images, video, user location and audio media messages to other users using standard cellular mobile numbers. As of February 2016, WhatsApp had a userbase of up to one billion,[10] making it the most globally popular messaging application. WhatsApp Inc., based in Mountain View, California, was acquired by Facebook Inc. on February 19, 2014, for approximately US $19.3 billion.</p>
</ContentTemplate>
</TabItem>
</TabItems>
</SfTab>
</div>
<style>
.e-tab .e-content .e-item {
padding: 10px;
}
</style>
```

Order ID	Customer Name	Order Date	TWITTER	FACEBOOK	WHATSAPP
1001	BLONP	9/17/2019	<p>Twitter is an online social networking service that enables users to send and read short 140-character messages called tweets. Registered users can read and post tweets, but those who are unregistered can only read them. Users access Twitter through the website interface, SMS or mobile device app. Twitter Inc. is based in San Francisco and has more than 25 offices around the world. Twitter was created in March 2006 by Jack Dorsey, Evan Williams, Biz Stone, and Noah Glass and launched in July 2006. The service rapidly gained worldwide popularity, with more than 100 million users posting 340 million tweets a day in 2012. The service also handled 1.6 billion search queries per day.</p>		
1002	BLONP	9/16/2019			
1003	ANANTR	9/15/2019			
1004	BOLID	9/14/2019			
1005	ANTON	9/13/2019			

Pane Settings in the Blazor Splitter Component

This section explains the pane settings behavior.

Pane visibility

You can show or hide the Splitter panes using the **Visible** property based on the application's demand like initial load or dynamic cases. The **Visible** property is enabled by default in the Blazor splitter.

In the following code example, the **Visible** property binds to the second **SplitterPane** to show/hide the pane on **CheckBox** state change.

ASPX-CS

```
@using Syncfusion.Blazor.Layouts
```

```
<button class="e-btn" @onclick="@ToggleMiddlePane"> Toggle Middle Pane
</button>
<SfSplitter Height="240px" Width="700px" SeparatorSize="2">
<SplitterPanes>
<SplitterPane Size="30%" Collapsible="true">
<ContentTemplate>
<div> Left Pane </div>
</ContentTemplate>
</SplitterPane>
<SplitterPane Size="30%" Collapsible="true"
Visible="@this.PaneMiddleVisibility">
<ContentTemplate>
<div> Middle Pane </div>
</ContentTemplate>
</SplitterPane>
<SplitterPane Collapsible="true">
<ContentTemplate>
<div> Right Pane </div>
</ContentTemplate>
</SplitterPane>
</SplitterPanes>
</SfSplitter>
@code {
public bool PaneMiddleVisibility { get; set; } = false;
public void ToggleMiddlePane()
{
this.PaneMiddleVisibility = !this.PaneMiddleVisibility;
}
}
```

Split Panes in Blazor Splitter Component

This section explains split-panes behaviors.

Horizontal layout

By default, splitter will be rendered in horizontal orientation. Splitter container will be split as panes in horizontal flow direction with vertical separator.

ASPX-CS

```
@using Syncfusion.Blazor.Layouts
<SfSplitter Height="200px" Width="600px">
<SplitterPanes>
<SplitterPane Size="200px">
<ContentTemplate>
<div>
<div class="content">
<h3 class="h3">Grid </h3>
The ASP.NET DataGrid control, or DataTable is a feature-rich control used to
display data in a tabular format.
</div>
</div>
</ContentTemplate>
</SplitterPane>
<SplitterPane Size="200px">
<ContentTemplate>
<div>
```

```
<div class="content">
<h3 class="h3">Schedule </h3>
The ASP.NET Scheduler, a.k.a. event calendar, facilitates almost all
calendar features, thus allowing users to manage their time efficiently
</div>
</div>
</ContentTemplate>
</SplitterPane>
<SplitterPane Size="200px">
<ContentTemplate>
<div>
<div class="content">
<h3 class="h3">Chart </h3>
ASP.NET charts, a well-crafted easy-to-use charting package, is used to add
beautiful charts in web and mobile applications
</div>
</div>
</ContentTemplate>
</SplitterPane>
</SplitterPanels>
</SfSplitter>
<style>
.content {
padding: 10px;
}
</style>
```

Grid	Schedule	Chart
The ASP.NET DataGrid control, or DataTable is a feature-rich control used to display data in a tabular format.	The ASP.NET Scheduler, a.k.a. event calendar, facilitates almost all calendar features, thus allowing users to manage their time efficiently	ASP.NET charts, a well-crafted easy-to-use charting package, is used to add beautiful charts in web and mobile applications

Vertical layout

By setting value to **Orientation** API as **Vertical**, splitter will be rendered in vertical orientation. Splitter container will be split as panes in vertical flow direction with horizontal separator.

ASPX-CS

```
@using Syncfusion.Blazor.Layouts
<SfSplitter Height="305px" Width="600px" Orientation="Orientation.Vertical">
<SplitterPanels>
<SplitterPane Size="100px">
<ContentTemplate>
<div>
<div class="content">
<h3 class="h3">Grid </h3>
```

The ASP.NET DataGrid control, or DataTable is a feature-rich control used to display data in a tabular format.

```
</div>
</div>
</ContentTemplate>
</SplitterPane>
<SplitterPane Size="100px">
<ContentTemplate>
<div>
<div class="content">
<h3 class="h3">Schedule </h3>
```

The ASP.NET Scheduler, a.k.a. event calendar, facilitates almost all calendar features, thus allowing users to manage their time efficiently

```
</div>
</div>
</ContentTemplate>
</SplitterPane>
<SplitterPane Size="100px">
<ContentTemplate>
<div>
<div class="content">
<h3 class="h3">Chart </h3>
```

ASP.NET charts, a well-crafted easy-to-use charting package, is used to add beautiful charts in web and mobile applications

```
</div>
</div>
</ContentTemplate>
</SplitterPane>
</SplitterPanels>
</SfSplitter>
<style>
.content {
padding: 9px;
}
</style>
```

<p>Grid</p> <p>The ASP.NET DataGrid control, or DataTable is a feature-rich control used to display data in a tabular format.</p>	
<p>Schedule</p> <p>The ASP.NET Scheduler, a.k.a. event calendar, facilitates almost all calendar features, thus allowing users to manage their time efficiently</p>	
<p>Chart</p> <p>ASP.NET charts, a well-crafted easy-to-use charting package, is used to add beautiful charts in web and mobile applications</p>	

Multiple panes can also be rendered in splitter with both **Horizontal/Vertical** orientations.

Separator

By default, pane separator will be rendered with **1px** width or height. The separator size can be customized by using **SeparatorSize** API.

For horizontal orientation, it will be considered as separator width. For vertical orientation, it will be considered as separator height.

ASPX-CS

```
@using Syncfusion.Blazor.Layouts
<SfSplitter Height="250px" Width="600px" SeparatorSize="5">
  <SplitterPanes>
    <SplitterPane Size="200px">
      <ContentTemplate>
        <div>
          <div class="content">
            <h3 class="h3">Grid </h3>
            The ASP.NET DataGrid control, or DataTable is a feature-rich control used to
            display data in a tabular format.
          </div>
        </div>
      </ContentTemplate>
    </SplitterPane>
    <SplitterPane Size="200px">
      <ContentTemplate>
        <div>
          <div class="content">
            <h3 class="h3">Schedule </h3>
            The ASP.NET Scheduler, a.k.a. event calendar, facilitates almost all
            calendar features, thus allowing users to manage their time efficiently
          </div>
        </div>
      </ContentTemplate>
    </SplitterPane>
  </SplitterPanes>
</SfSplitter>
```

```

</SplitterPane>
<SplitterPane Size="200px">
<ContentTemplate>
<div>
<div class="content">
<h3 class="h3">Chart </h3>
ASP.NET charts, a well-crafted easy-to-use charting package, is used to add
beautiful charts in web and mobile applications
</div>
</div>
</ContentTemplate>
</SplitterPane>
</SplitterPanes>
</SfSplitter>
<style>
.content {
padding: 10px;
}
</style>

```

Grid	Schedule	Chart
The ASP.NET DataGrid control, or DataTable is a feature-rich control used to display data in a tabular format.	The ASP.NET Scheduler, a.k.a. event calendar, facilitates almost all calendar features, thus allowing users to manage their time efficiently	ASP.NET charts, a well-crafted easy-to-use charting package, is used to add beautiful charts in web and mobile applications

Nested Splitter

Splitter provides support to render the nested pane to achieve the complex layouts. The same `SplitterPane` tag can be used for splitter pane and nested splitter.

Also the nested splitter can be rendered using direct child of the splitter pane. For this, nested splitter should have `100%` width and height to match with the parent pane dimensions.

ASPX-CS

```

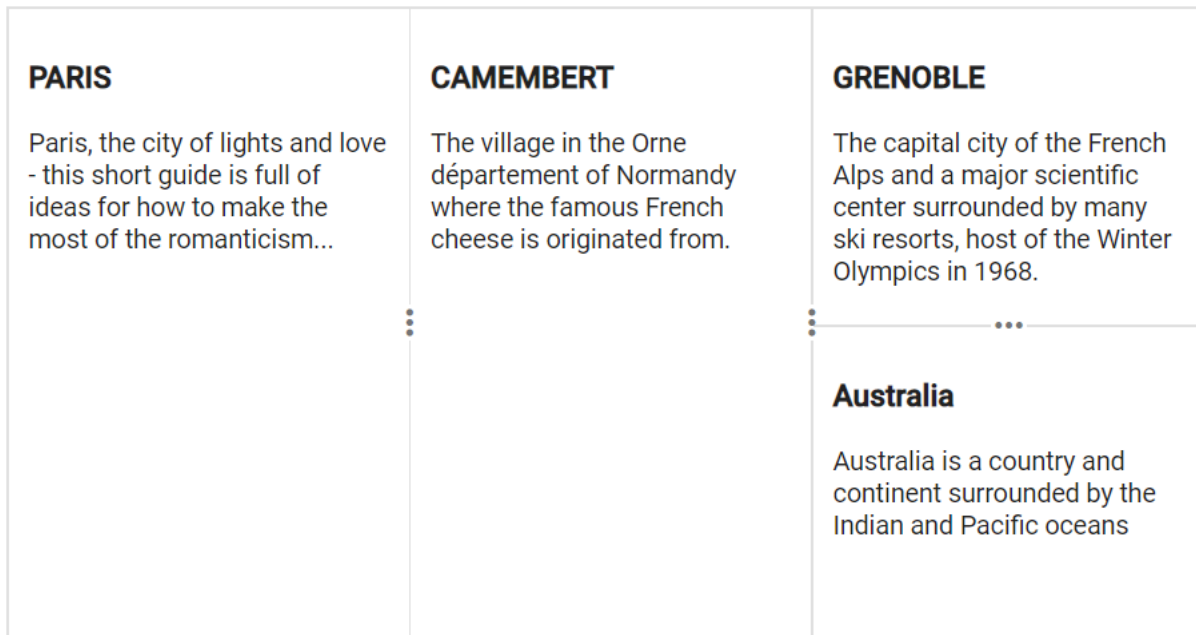
@using Syncfusion.Blazor.Layouts
<SfSplitter Height="316px" Width="600px">
<SplitterPanes>
<SplitterPane Size="200px">
<ContentTemplate>
<div>
<div class="content">
<h3 class="h3">PARIS </h3>

```

```

Paris, the city of lights and love - this short guide is full of ideas for
how to make the most of the romanticism...
</div>
</div>
</ContentTemplate>
</SplitterPane>
<SplitterPane Size="200px">
<ContentTemplate>
<div>
<div class="content">
<h3 class="h3">CAMEMBERT </h3>
The village in the Orne département of Normandy where the famous French
cheese is originated from.
</div>
</div>
</ContentTemplate>
</SplitterPane>
<SplitterPane Size="200px">
<div>
<SfSplitter ID="innerSplitter" Orientation="Orientation.Vertical">
<SplitterPanels>
<SplitterPane Size="150px" Min="20%">
<ContentTemplate>
<div>
<div class="content">
<h3 class="h3">GRENOBLE </h3>
The capital city of the French Alps and a major scientific center surrounded
by many ski resorts, host of the Winter Olympics in 1968.
</div>
</div>
</ContentTemplate>
</SplitterPane>
<SplitterPane Size="100px" Min="20%">
<ContentTemplate>
<div>
<div class="content">
<h3 class="h3">Australia </h3>
Australia is a country and continent surrounded by the Indian and Pacific
oceans
</div>
</div>
</ContentTemplate>
</SplitterPane>
</SplitterPanels>
</SfSplitter>
</div>
</SplitterPane>
</SplitterPanels>
</SfSplitter>
<style>
.content {
padding: 10px;
}
#innerSplitter {
border: none;
}
</style>

```



Add or remove pane

The panes can be added programmatically but it will be complex. For this, use the `AddPane/RemovePane` methods to add and remove the panes dynamically in the splitter.

Add pane

The panes can be added dynamically in the splitter by passing `PaneProperties` along with index to the `AddPane` method.

ASPX-CS

```
@using Syncfusion.Blazor.Layouts
@using Syncfusion.Blazor.Buttons
<SfSplitter @ref="SplitterObj" Height="200px" Width="600px">
  <SplitterPanes>
    <SplitterPane Size="190px">
      <ContentTemplate>
        <div>
          <div class='content'>
            Pane 1
          </div>
        </div>
      </ContentTemplate>
    </SplitterPane>
    <SplitterPane Size="190px">
      <ContentTemplate>
        <div>
          <div class='content'>
            Pane 2
          </div>
        </div>
      </ContentTemplate>
    </SplitterPane>
  </SplitterPanes>
```



```

</SfSplitter>
<SfButton Content="Add Pane" @onclick="@Add"></SfButton>
<style>
.content {
padding: 9px;
}
#defaultBtn {
margin-top: 15px;
}
</style>
@code {
SfSplitter SplitterObj;
private SplitterPane AddingPane = new SplitterPane() {
Size = "190px",
Content = "New Pane",
Min = "30px",
Max = "250px"
};
private async Task Add()
{
await this.SplitterObj.AddPaneAsync(AddingPane, 1);
}
}

```

Remove pane

The split panes can be removed dynamically by passing the pane index to **RemovePane** method.

ASPX-CS

```

@using Syncfusion.Blazor.Layouts
@using Syncfusion.Blazor.Buttons
<SfSplitter @ref="SplitterObj" Height="200px" Width="600px">
<SplitterPanels>
<SplitterPane Size="200px">
<ContentTemplate>
<div>
<div class='content'>
Pane 1
</div>
</div>
</ContentTemplate>
</SplitterPane>
<SplitterPane Size="200px">
<ContentTemplate>
<div>
<div class='content'>
Pane 2
</div>
</div>
</ContentTemplate>
</SplitterPane>
<SplitterPane Size="200px">
<ContentTemplate>
<div>
<div class='content'>
Pane 3

```

```

</div>
</div>
</ContentTemplate>
</SplitterPane>
</SplitterPanels>
</SfSplitter>
<SfButton Content="Remove Pane" @onclick="@Remove"></SfButton>
<style>
.content {
padding: 9px;
}
#defaultBtn {
margin-top: 15px;
}
</style>
@code {
SfSplitter SplitterObj;
private async Task Remove()
{
await this.SplitterObj.RemovePaneAsync(1);
}
}

```

See Also

- [Resizable split panes](#)
- [Collapsible panes](#)
- [Define size to a panes](#)
- [Specify content to a panes](#)

Expand and Collapse in Blazor Splitter Component

Collapsible panes

The Splitter panes can be configured with built-in expand and collapse functionalities. By default, the collapsible behavior is disabled. Enable the **Collapsible** behavior in the SplitterPane property to show or hide the expand or collapse icons in the panes. The panes can be dynamically expanded and collapsed by the corresponding icons.

The following code shows how to enable collapsible behavior.

ASPX-CS

```

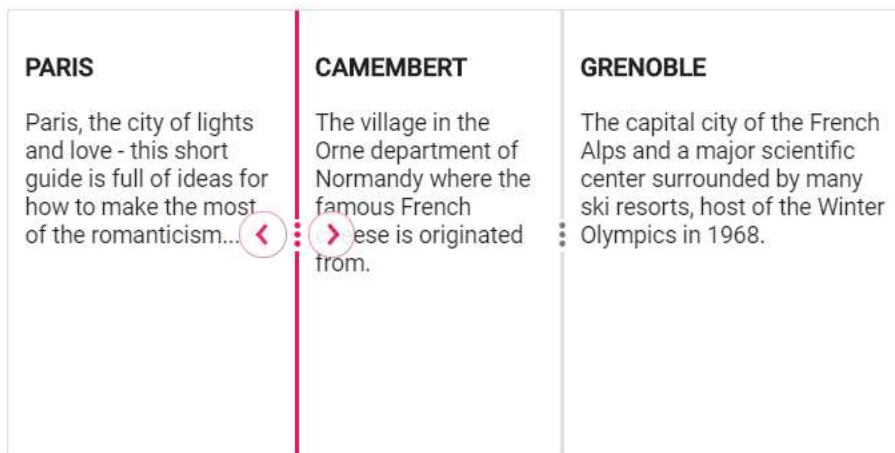
@using Syncfusion.Blazor.Layouts
<SfSplitter Height="200px" Width="600px" SeparatorSize=2>
<SplitterPanels>
<SplitterPane Collapsible="true">
<ContentTemplate>
<div>
<div class="content">
<h3 class="h3">PARIS </h3>
Paris, the city of lights and love - this short guide is full of ideas for
how to make the most of the romanticism...
</div>
</div>
</ContentTemplate>

```

```

</SplitterPane>
<SplitterPane Collapsible="true">
  <ContentTemplate>
    <div>
      <div class="content">
        <h3 class="h3">CAMEMBERT </h3>
        The village in the Orne department of Normandy where the famous French
        cheese is originated from.
      </div>
    </div>
  </ContentTemplate>
</SplitterPane>
<SplitterPane Collapsible="true">
  <ContentTemplate>
    <div>
      <div class="content">
        <h3 class="h3">GRENOBLE </h3>
        The capital city of the French Alps and a major scientific center surrounded
        by many ski resorts, host of the Winter Olympics in 1968.
      </div>
    </div>
  </ContentTemplate>
</SplitterPane>
</SplitterPanels>
</SfSplitter>
<style>
  .content {
    padding: 10px;
  }
</style>

```



Programmatically control the expand and collapse action

The Splitter provides public method to control the expand and collapse behavior programmatically using the `Expand` and `Collapse` methods.

ASPX-CS

```

@using Syncfusion.Blazor.Layouts
@using Syncfusion.Blazor.Buttons

```

```

<SfSplitter @ref="SplitterObj" Height="200px" Width="600px" SeparatorSize=2>
  <SplitterPanels>
    <SplitterPane Collapsible="true">
      <ContentTemplate>
        <div>
          <div class="content">
            <h3 class="h3">PARIS </h3>
            Paris, the city of lights and love - this short guide is full of ideas for
            how to make the most of the romanticism...
          </div>
        </div>
      </ContentTemplate>
    </SplitterPane>
    <SplitterPane Collapsible="true">
      <ContentTemplate>
        <div>
          <div class="content">
            <h3 class="h3">CAMEMBERT </h3>
            The village in the Orne department of Normandy where the famous French
            cheese is originated from.
          </div>
        </div>
      </ContentTemplate>
    </SplitterPane>
    <SplitterPane Collapsible="true">
      <ContentTemplate>
        <div>
          <div class="content">
            <h3 class="h3">GRENOBLE </h3>
            The capital city of the French Alps and a major scientific center surrounded
            by many ski resorts, host of the Winter Olympics in 1968.
          </div>
        </div>
      </ContentTemplate>
    </SplitterPane>
  </SplitterPanels>
</SfSplitter>
<div id="button-container">
  <SfButton @onclick="@Expand"> Expand </SfButton>
  <SfButton @onclick="@Collapse"> Collapse </SfButton>
</div>
<style>
.content {
padding: 10px;
}
#button-container {
display: flex;
margin: 20px 22% 40px;
}
</style>
@code {
SfSplitter SplitterObj;
private async Task Expand()
{
await this.SplitterObj.ExpandAsync(0);
}
private async Task Collapse()

```

```
{
    await this.SplitterObj.CollapseAsync(0);
}
}
```

PARIS Paris, the city of lights and love - this short guide is full of ideas for how to make the most of the romanticism...	CAMEMBERT The village in the Orne department of Normandy where the famous French cheese is originated from.	GRENOBLE The capital city of the French Alps and a major scientific center surrounded by many ski resorts, host of the Winter Olympics in 1968.
---	---	---

EXPAND

COLLAPSE

Specify initial state to panes

Specific panes can be rendered with collapsed state on page load. Specify a Boolean value to the **Collapsed** property to control this behavior. The following code explains how to collapse panes on rendering (the second panes renders with collapsed state).

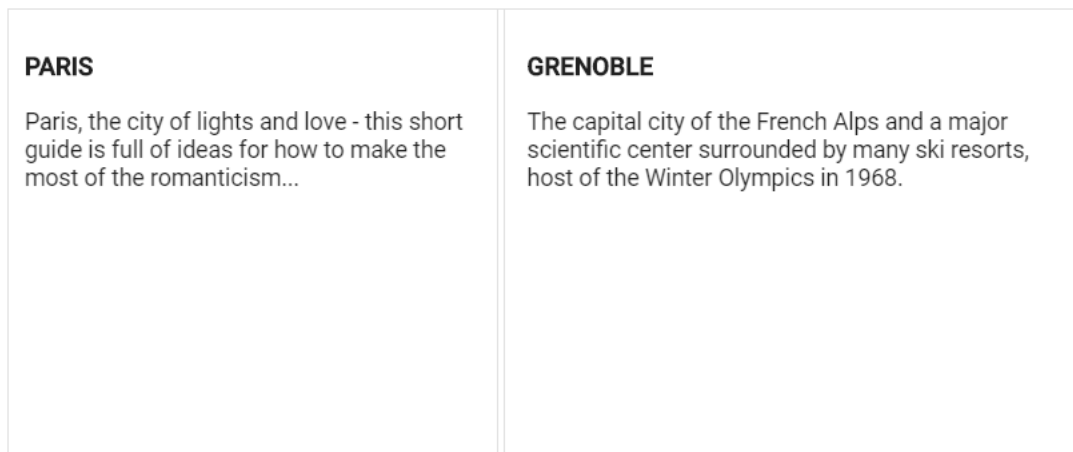
ASPX-CS

```
@using Syncfusion.Blazor.Layouts
<SfSplitter Height="200px" Width="600px" SeparatorSize=2>
    <SplitterPanels>
        <SplitterPane Collapsible="true">
            <ContentTemplate>
                <div>
                    <div class="content">
                        <h3 class="h3">PARIS </h3>
                        Paris, the city of lights and love - this short guide is full of ideas for
                        how to make the most of the romanticism...
                    </div>
                </div>
            </ContentTemplate>
        </SplitterPane>
        <SplitterPane Collapsible="true" @bind-Collapsed="@collapsed">
            <ContentTemplate>
                <div>
                    <div class="content">
                        <h3 class="h3">CAMEMBERT </h3>
                        The village in the Orne department of Normandy where the famous French
                        cheese is originated from.
                    </div>
                </div>
            </ContentTemplate>
        </SplitterPane>
    </SplitterPanels>
</SfSplitter>
```

```

</ContentTemplate>
</SplitterPane>
<SplitterPane Collapsible="true">
<ContentTemplate>
<div>
<div class="content">
<h3 class="h3">GRENOBLE </h3>
The capital city of the French Alps and a major scientific center surrounded
by many ski resorts, host of the Winter Olympics in 1968.
</div>
</div>
</ContentTemplate>
</SplitterPane>
</SplitterPanels>
</SfSplitter>
<style>
.content {
padding: 10px;
}
</style>
@code {
private bool collapsed { get; set; } = true;
}

```



See Also

- [Resizable split panes](#)

Two way Binding in Blazor Splitter Component

The splitter `SplitterPane Collapsed` property supports the two-way binding and it can be achieved by using the `bind-Collapsed` attribute. If the component value has been changed, it will affect all the places where we bind the variable for the `bind-value` attribute.

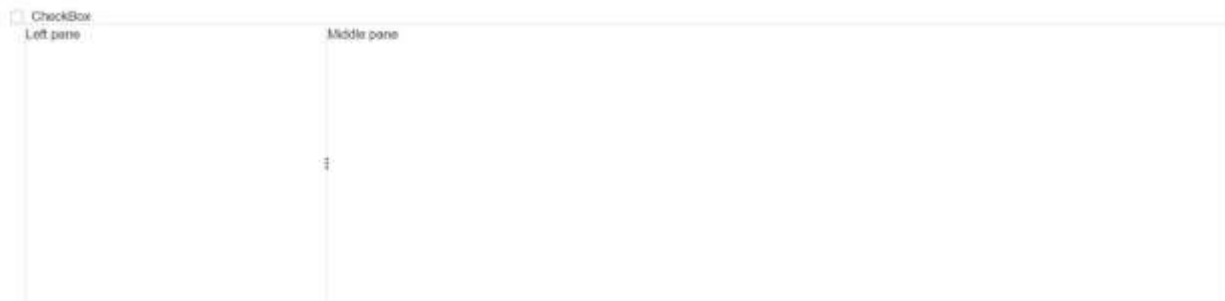
In the following example, if either the value is changed in checkbox or splitter first pane collapsed state, it will reflect in both the checkbox and splitter pane.

ASPX-CS

```

@using Syncfusion.Blazor.Layouts
@using Syncfusion.Blazor.Buttons
<div>
<div class="row">
<SfCheckBox @bind-Checked="@collapsed" Label="CheckBox"></SfCheckBox>
</div>
<SfSplitter Height="300px" Width="100%">
<SplitterPanes>
<SplitterPane Size="25%" Min="60px" Collapsible="true" @bind-
Collapsed="@collapsed">
<div>
<div class="contents">
<div>Left pane</div>
</div>
</div>
</SplitterPane>
<SplitterPane Size="50%" Min="60px" Collapsible="true">
<div>
<div class='contents'>
<div>Right pane</div>
</div>
</div>
</SplitterPane>
</SplitterPanes>
</SfSplitter>
</div>
@code {
public bool collapsed { get; set; } = false;
}

```



See Also

- [Resizable split panes](#)

Resize in Blazor Splitter Component

By default, resizing will be enabled for split panes. Resizing gripper element will be added to the separator to make the resize easy.

Horizontal splitter allows to resize in horizontal directions. Vertical splitter allows to resize in vertical directions.

While resizing, previous and next panes will adjust its dimensions automatically.

Min and Max validation

Splitter allows to set the minimum and maximum sizes for each pane. Resizing will not be occurred over the minimum and maximum values.

ASPX-CS

```
@using Syncfusion.Blazor.Layouts
<SfSplitter Height="200px" Width="600px" SeparatorSize=4>
  <SplitterPanes>
    <SplitterPane Size="200px" Min="20%" Max="40%">
      <ContentTemplate>
        <div>
          <div class="content">
            <h3 class="h3">Grid </h3>
            The ASP.NET DataGrid control, or DataTable is a feature-rich control used to
            display data in a tabular format.
          </div>
        </div>
      </ContentTemplate>
    </SplitterPane>
    <SplitterPane Size="200px" Min="30%" Max="60%">
      <ContentTemplate>
        <div>
          <div class="content">
            <h3 class="h3">Schedule </h3>
            The ASP.NET Scheduler, a.k.a. event calendar, facilitates almost all
            calendar features, thus allowing users to manage their time efficiently
          </div>
        </div>
      </ContentTemplate>
    </SplitterPane>
    <SplitterPane>
      <ContentTemplate>
        <div>
          <div class="content">
            <h3 class="h3">Chart </h3>
            ASP.NET charts, a well-crafted easy-to-use charting package, is used to add
            beautiful charts in web and mobile applications
          </div>
        </div>
      </ContentTemplate>
    </SplitterPane>
  </SplitterPanes>
</SfSplitter>
<style>
  .content {
    padding: 10px;
  }
</style>
```


Grid	Schedule	Chart
The ASP.NET DataGrid control, or DataTable is a feature-rich control used to display data in a tabular format.	The ASP.NET Scheduler, a.k.a. event calendar, facilitates almost all calendar features, thus allowing users to manage their time efficiently	ASP.NET charts, a well-crafted easy-to-use charting package, is used to add beautiful charts in web and mobile applications

Prevent resizing

The resizing for the pane can be disabled by setting `false` to the `Resizable` API within `SplitterPane`.

Splitter resizing will be enabled only when the target of the adjacent pane's `Resizable` api should also be in `true` state.

ASPX-CS

```
@using Syncfusion.Blazor.Layouts
<SfSplitter Height="200px" Width="600px" SeparatorSize=4>
  <SplitterPanes>
    <SplitterPane Size="200px" Min="20%" Max="40%" Resizable="false">
      <ContentTemplate>
        <div>
          <div class="content">
            <h3 class="h3">Grid </h3>
            The ASP.NET DataGrid control, or DataTable is a feature-rich control used to
            display data in a tabular format.
          </div>
        </div>
      </ContentTemplate>
    </SplitterPane>
    <SplitterPane Size="200px" Min="30%" Max="60%">
      <ContentTemplate>
        <div>
          <div class="content">
            <h3 class="h3">Schedule </h3>
            The ASP.NET Scheduler, a.k.a. event calendar, facilitates almost all
            calendar features, thus allowing users to manage their time efficiently
          </div>
        </div>
      </ContentTemplate>
    </SplitterPane>
    <SplitterPane>
      <ContentTemplate>
        <div>
          <div class="content">
            <h3 class="h3">Chart </h3>
            ASP.NET charts, a well-crafted easy-to-use charting package, is used to add
            beautiful charts in web and mobile applications
          </div>
        </div>
      </ContentTemplate>
    </SplitterPane>
  </SplitterPanes>
</SfSplitter>
```

```

</ContentTemplate>
</SplitterPane>
</SplitterPanels>
</SfSplitter>
<style>
.content {
padding: 10px;
}
</style>

```

Grid	Schedule	Chart
The ASP.NET DataGrid control, or DataTable is a feature-rich control used to display data in a tabular format.	The ASP.NET Scheduler, a.k.a. event calendar, facilitates almost all calendar features, thus allowing users to manage their time efficiently	ASP.NET charts, a well-crafted easy-to-use charting package, is used to add beautiful charts in web and mobile applications

Refresh content on resizing

While resizing the panes, the pane contents can be refreshed by using either `OnResizeStart`, `Resizing` or `OnResizeStop`.

Customize the resize grip and cursor

The resize gripper icon and cursor can be customized in css level.

ASPX-CS

```

@using Syncfusion.Blazor.Layouts
<SfSplitter Height="200px" Width="600px" SeparatorSize=3>
  <SplitterPanels>
    <SplitterPane Size="200px" Min="20%" Max="40%">
      <ContentTemplate>
        <div>
          <div class="content">
            <h3 class="h3">Grid </h3>
            The ASP.NET DataGrid control, or DataTable is a feature-rich control used to
            display data in a tabular format.
          </div>
        </div>
      </ContentTemplate>
    </SplitterPane>
    <SplitterPane Size="200px" Min="30%" Max="60%">
      <ContentTemplate>
        <div>
          <div class="content">
            <h3 class="h3">Schedule </h3>
            The ASP.NET Scheduler, a.k.a. event calendar, facilitates almost all
            calendar features, thus allowing users to manage their time efficiently
          </div>

```

```

</div>
</ContentTemplate>
</SplitterPane>
<SplitterPane>
<ContentTemplate>
<div>
<div class="content">
<h3 class="h3">Chart </h3>
ASP.NET charts, a well-crafted easy-to-use charting package, is used to add
beautiful charts in web and mobile applications
</div>
</div>
</ContentTemplate>
</SplitterPane>
</SplitterPanels>
</SfSplitter>
<style>
.content {
padding: 10px;
}
.e-splitter .e-split-bar .e-resize-handler:before {
content: "\e934";
font-family: 'e-icons';
font-size: 11px;
transform: rotate(90deg);
}
.e-splitter .e-split-bar.e-split-bar-horizontal.e-resizable-split-bar,
.e-splitter .e-split-bar.e-split-bar-horizontal.e-resizable-split-bar::after
{
cursor: e-resize;
}
</style>

```

Grid	Schedule	Chart
The ASP.NET DataGrid control, or DataTable is a feature-rich control used to display data in a tabular format.	The ASP.NET Scheduler, a.k.a. event calendar, facilitates almost all calendar features, thus allowing users to manage their time efficiently	ASP.NET charts, a well-crafted easy-to-use charting package, is used to add beautiful charts in web and mobile applications

See Also

- [Collapsible panes](#)

State Management in Blazor Splitter Component

State management allows users to save and load Splitter state. The splitter will use user-provided state to render instead of its properties provided declaratively.

The following properties can be saved and loaded into each **SplitterPane** later.

Property |

Collapsed |

Min |

Max |

Size |

Enabling persistence in Splitter

State persistence allows the Splitter to retain the current splitter panes state in the browser local storage for state maintenance. This action is handled through the `EnablePersistence` property which is set to false by default. When it is set to true, some properties of the `SplitterPane` will be retained even after refreshing the page.

The state will be persisted based on **ID** property. So, it is recommended to explicitly set the **ID** property for Splitter.

ASPX-CS

```
@using Syncfusion.Blazor.Layouts
<SfSplitter ID="Splitter" Height="300px" Width="100%"
EnablePersistence="true">
  <SplitterPanes>
    <SplitterPane @bind-Min="@PaneMin" @bind-Size="@PanelSize"
Collapsible="true">
      <div>
        <div style="text-align: center">
          <div>Left pane</div>
        </div>
      </div>
    </SplitterPane>
    <SplitterPane @bind-Min="@PaneMin" @bind-Size="@Pane2Size"
Collapsible="true">
      <div>
        <div style="text-align: center">
          <div>Middle pane</div>
        </div>
      </div>
    </SplitterPane>
    <SplitterPane @bind-Min="@PaneMin" Collapsible="true">
      <div>
        <div style="text-align: center">
          <div>Last Pane</div>
        </div>
      </div>
    </SplitterPane>
  </SplitterPanes>
</SfSplitter>

@code {
private string PanelSize { get; set; } = "25%";
private string Pane2Size { get; set; } = "50%";
private string PaneMin { get; set; } = "60px";
}
```

Different Layouts in Blazor Splitter Component

By using splitter control, the different layouts can be created with multiple and nested panes.

Code editor style layout

Step 1:

Create the element with two child to render the outer splitter.

ASPX-CS

```
<SfSplitter Height="400px" Orientation="Orientation.Vertical">
  <SplitterPanes>
    <SplitterPane Size="53%" Min="30%">
      <SfSplitter Height="220px" Width="100%">
        <SplitterPanes>
          <SplitterPane Size="29%" Min="23%"></SplitterPane>
          <SplitterPane Size="20%" Min="15%"></SplitterPane>
          <SplitterPane Size="35%" Min="35%"></SplitterPane>
        </SplitterPanes>
      </SfSplitter>
    </SplitterPane>
    <SplitterPane Size="53%" Min="30%">
      <ContentTemplate>
        <div class="code-editor-content">
          <h3 class="h3">Preview of sample</h3>
          <div class="splitter-image">
            
          </div>
        </div>
      </ContentTemplate>
    </SplitterPane>
  </SplitterPanes>
</SfSplitter>
```

Step 2 :

Render the first pane of vertical splitter as a horizontal splitter.

ASPX-CS

```
@using Syncfusion.Blazor.Layouts
<SfSplitter Height="400px" Orientation="Orientation.Vertical">
  <SplitterPanes>
    <SplitterPane Size="53%" Min="30%">
      <SfSplitter Height="220px" Width="100%">
        <SplitterPanes>
          <SplitterPane Size="29%" Min="23%">
            <ContentTemplate>
              <div class="code-editor-content">
                <h3 class="h3">HTML</h3>
                <div class="code-preview">
                  &#60;<span>!DOCTYPE html></span>
                </div>&#60;<span>html></span></div>
                <div>&#60;<span>body></span></div>
                &#60;<span>div</span> id="custom-image">

```

```

<div style="margin-left: 5px">&#60;<span>img</span>
src="src/albert.png"></div>
<div>&#60;<span>div</span>&#62;</div>
<div>&#60;<span>/body</span></div>
<div>&#60;<span>/html</span></div>
</div>
</div>
</ContentTemplate>
</SplitterPane>
<SplitterPane Size="20%" Min="15%">
<ContentTemplate>
<div class="code-editor-content">
<h3 class="h3">CSS</h3>
<div class="code-preview">
<span>img {</span>
<div id="code-text">margin:<span>0 auto;</span></div>
<div id="code-text">display:<span>flex;</span></div>
<div id="code-text">height:<span>70px;</span></div>
<span>    }</span>
</div>
</div>
</ContentTemplate>
</SplitterPane>
<SplitterPane Size="35%" Min="35%">
<ContentTemplate>
<div class="code-editor-content">
<h3 class="h3">JavaScript</h3>
<div class="code-preview">
<span>var</span> image = document.getElementById("custom-image");
<div>image.addEventListener("click", function() {</div>
<div style="padding-left: 20px;">// Code block for click action</div>
<span> }</span>
</div>
</div>
</ContentTemplate>
</SplitterPane>
</SplitterPanels>
</SfSplitter>
</SplitterPane>
<SplitterPane Size="53%" Min="30%">
<ContentTemplate>
<div class="code-editor-content">
<h3 class="h3">Preview of sample</h3>
<div class="splitter-image">

</div>
</div>
</ContentTemplate>
</SplitterPane>
</SplitterPanels>
</SfSplitter>

```

CSS

```

<style>
#code-text {
margin-left: 5px;
}
#target {
margin: 20px auto;
max-width: 600px;
}
.code-preview {
margin-top: 15px;
font-size: 12px;
padding: 6px;
}
.control-section {
min-height: 370px;
margin-bottom: 15px;
margin-top: 10px;
}
.h3 {
font-size: 14px;
margin: 4px;
padding: 5px;
}
.code-editor-content {
padding: 12px;
}
.splitter-image {
margin: 0 auto;
display: flex;
height: 115px;
margin-top: 10px;
}
</style>

```

Once the above configurations has been completed, the output will be displayed like [this](#)

Outlook style layout

Step 1:

Create the element with three panes and place the elements within the pane to render **Treeview**, **Listview** and **RichTextEditor**.

ASPX-CS

```

@using Syncfusion.Blazor.Lists
@using Syncfusion.Blazor.Inputs
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.Layouts
@using Syncfusion.Blazor.Navigations
@using Syncfusion.Blazor.RichTextEditor
<SfSplitter Height="493px" Width="100%" CssClass="outlook-splitter">
<SplitterEvents OnResizeStop="@ResizeStop"></SplitterEvents>
<SplitterPanes>
<SplitterPane Size="28%" Min="27%">
<ContentTemplate>
<div>

```

```

<div class="outlook-layout-content">
  <SfTreeView TValue="TreeData">
    <TreeViewFieldsSettings Id="Id" Text="Name" ParentID="Pid"
    HasChildren="HasChild" Selected="Selected" Expanded="Expanded"
    DataSource="@localData" />
    <TreeViewTemplates>
      <NodeTemplate>
        <div>
          <div class="treeviewdiv">
            <div style="float:left">
              <span class="treeName">@((context as TreeData).Name) </span>
            </div>
            @{
              @if (((context as TreeData).Count) != 0)
              {
                <div style="margin-right: 5px; float:right">
                  <span class="treeCount e-badge e-badge-primary">
                    @((context as TreeData).Count)
                  </span>
                </div>
              }
            }
          </div>
        </div>
      </NodeTemplate>
    </TreeViewTemplates>
  </SfTreeView>
</div>
</div>
</ContentTemplate>
</SplitterPane>
<SplitterPane Size="33%" Min="23%">
  <ContentTemplate>
    <div>
      <div class="outlook-layout-content">
        <SfListView DataSource="@dataSource" CssClass="e-list-template">
          <ListViewFieldSettings Text="Name" GroupBy="Order" />
          <ListViewTemplates TValue="DataModel">
            <Template>
              @{
                <div class="settings e-list-wrapper e-list-multi-line e-list-avatar">
                  <span class="e-list-item-header">@context.Name</span>
                  <div class="e-list-content">
                    <div class="status">@context.Content1</div>
                    <div id="list-message">@context.Content2</div>
                  </div>
                </div>
              }
            </Template>
            <GroupTemplate>
              <div class="e-list-wrapper">
                <span class="e-list-item-content"></span>
              </div>
            </GroupTemplate>
          </ListViewTemplates>
        </SfListView>
      </div>
    </div>
  </ContentTemplate>
</SplitterPane>

```



```

</div>
</ContentTemplate>
</SplitterPane>
<SplitterPane Size="37%" Min="30%">
<ContentTemplate>
<div>
<div class="outlook-layout-content">
<div style="width: 100%; padding: 15px;">
<table>
<tr>
<td><button class="e-btn e-flat e-outline">To...</button></td>
<td><SfTextBox /></td>
</tr>
<tr>
<td><button class="e-btn e-flat e-outline">Cc...</button></td>
<td><SfTextBox /></td>
</tr>
<tr>
<td><div id="subject-text">Subject</div></td>
<td><SfTextBox /></td>
</tr>
</table>
</div>
<div class="forum">
<div id="createpostholder">
<SfRichTextEditor @ref="RichTextEditorObj" Height="262px">
<RichTextEditorEvents Created="Created" />
</SfRichTextEditor>
<div id="buttonSection">
<SfButton IsPrimary="true">Send</SfButton>
<SfButton>Discard</SfButton>
</div>
</div>
</div>
</div>
</div>
</ContentTemplate>
</SplitterPane>
</SplitterPanels>
</SfSplitter>

```

Step 2 :

Place the `TreeViewTemplates` inside the `SfTreeView` to render the treeview template.

ASPX-CS

```

<TreeViewTemplates>
<NodeTemplate>
<div>
<div class="treeviewdiv">
<div style="float:left">
<span class="treeName">@((context as TreeData).Name)</span>
</div>
@{
@if (((context as TreeData).Count) != 0)
{

```

```
<div style="margin-right: 5px; float:right">
<span class="treeCount e-badge e-badge-primary">
@((context as TreeData).Count)
</span>
</div>
}
}
</div>
</div>
</NodeTemplate>
</TreeViewTemplates>
```

Step 3 :

Define the components DataSource in `@code` section.

ASPX-CS

```
@code {
private SfRichTextEditor RichTextEditorObj;
public class DataModel
{
public string Name { get; set; }
public string Content1 { get; set; }
public string Content2 { get; set; }
public string Id { get; set; }
public int Order { get; set; }
}
public class TreeData
{
public int Id { get; set; }
public int? Pid { get; set; }
public string Name { get; set; }
public bool HasChild { get; set; }
public bool Expanded { get; set; }
public int Count { get; set; }
public bool Selected { get; set; }
}
private List<TreeData> localData = new List<TreeData>()
{
new TreeData { Id = 1, Name = "Favorites", HasChild = true, },
new TreeData { Id = 2, Pid = 1, Name = "Sales Reports", Count = 4 },
new TreeData { Id = 3, Pid = 1, Name = "Sent Items" },
new TreeData { Id = 4, Pid = 1, Name = "Marketing Reports", Count = 6 },
new TreeData { Id = 5, HasChild = true, Name = "Andrew Fuller", Expanded =
true },
new TreeData { Id = 6, Pid = 5, Name = "Inbox", Selected = true, Count = 20
},
new TreeData { Id = 7, Pid = 5, Name = "Drafts", Count = 5 },
new TreeData { Id = 8, Pid = 5, Name = "Deleted Items" },
new TreeData { Id = 9, Pid = 5, Name = "Sent Items" },
new TreeData { Id = 10, Pid = 5, Name = "Sales Reports", Count = 4 },
new TreeData { Id = 11, Pid = 5, Name = "Marketing Reports", Count = 6 },
new TreeData { Id = 12, Pid = 5, Name = "Outbox" },
new TreeData { Id = 13, Pid = 5, Name = "Junk Email" },
new TreeData { Id = 14, Pid = 5, Name = "RSS Feed" },
new TreeData { Id = 15, Pid = 5, Name = "Trash" }
```

```
};
private List<DataModel> dataSource = new List<DataModel>()
{
    new DataModel { Name= "Selma Tally", Content1="Apology marketing email",
    Content2="Hello Ananya Singleton", Id = "1", Order = 0},
    new DataModel { Name= "Illa Russo", Content1="Annual conference",
    Content2="Hi jeani Moresa", Id = "4", Order = 0},
    new DataModel { Name= "Camden Macmellon", Content1="Reference request-
    Camden hester", Content2="Hello Kerry Best", Id = "3", Order = 0},
    new DataModel { Name= "Garth Owen", Content1="Application for job Title",
    Content2="Hello Illa Russo", Id = "2", Order = 0},
    new DataModel { Name= "Ursula Patterson", Content1="Programmaer Position
    Applicant", Content2="Hello Kerry best", Id = "5", Order = 0},
};
private async Task Created()
{
    await this.RichTextEditorObj.RefreshUIAsync();
}
private async Task ResizeStop()
{
    await this.RichTextEditorObj.RefreshUIAsync();
}
}
```

CSS

```
<style>
#discard {
margin-left: 7px;
}
.outlook-layout-content table {
width: 100%;
}
.outlook-layout-content td {
padding: 2px;
}
.control-section {
min-height: 370px;
}
.outlook-layout-content .e-treeview .e-list-text {
width: 100%;
}
#groupedList.e-listview .e-list-group-item {
height: 0;
}
.outlook-splitter .settings.e-list-wrapper.e-list-multi-line.e-list-avatar {
padding: 15px;
}
#buttonSection {
padding: 7px;
}
.outlook-layout-content #createpostholder {
padding-right: 4px;
padding-left: 3px;
}
.outlook-splitter #tree ul.e-list-parent.e-ul {
```

```
padding: 0 0 0 16px;
}
</style>
```

Once the above configurations has been completed, the output will be displayed like [this](#).

See Also

- [Multiple panes in Splitter](#)

Resize Icon Template

The Splitter allows to customize the resize icon of the separator using the template, where any image or other templates can be rendered as resize icon.

ASPX-CS

```
@using Syncfusion.Blazor.Layouts
<div>Horizontal Splitter</div>
<SfSplitter Height="240px" Width="100%">
  <SplitterTemplates>
    <Separator>
      <div style="height: 10px; border: 2px solid red"></div>
    </Separator>
  </SplitterTemplates>
  <SplitterPanes>
    <SplitterPane Collapsible="true">
      <ContentTemplate>
        <div> Left Pane </div>
      </ContentTemplate>
    </SplitterPane>
    <SplitterPane Collapsible="true">
      <ContentTemplate>
        <div> Middle Pane </div>
      </ContentTemplate>
    </SplitterPane>
    <SplitterPane Collapsible="true">
      <ContentTemplate>
        <div> Right Pane </div>
      </ContentTemplate>
    </SplitterPane>
  </SplitterPanes>
</SfSplitter>
```



Style and appearance in Blazor Splitter 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 split bar

Use the following CSS to customize the split bar properties.

Horizontal split bar

CSS

```
/* default split bar color */
.e-splitter .e-split-bar.e-split-bar-horizontal{
background: blue;
}
/* split bar color in hover and active state */
.e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover,
.e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-active {
background: green;
}
```

Vertical split bar

CSS

```
/* default split bar color */
.e-splitter .e-split-bar.e-split-bar-vertical {
background: blue;
}
/* split bar color in hover and active state */
.e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover,
.e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-active {
background: green;
}
```

Customizing the split bar resize handle

Use the following CSS to customize the split bar resize handle.

Horizontal split bar resize handle

CSS

```
/* default split bar resize handle color */
.e-splitter .e-split-bar.e-split-bar-horizontal .e-resize-handler {
color: rgba(20, 27, 233, 0.54);
}
/* default split bar resize handle color in hover and active state */
.e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover .e-resize-handler {
color: green;
}
```

Vertical split bar resize handle

CSS

```
/* default split bar resize handle color */
.e-splitter .e-split-bar.e-split-bar-vertical .e-resize-handler {
```

```
color: rgba(20, 27, 233, 0.54);
}
/* default split bar resize handle color in hover and active state */
.e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover .e-resize-
handler {
color: green;
}
```

Customizing the split bar arrows

Use the following CSS to customize the split bar arrows.

Horizontal split bar resize arrows

CSS

```
/* split bar arrows */
.e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover .e-
navigate-arrow.e-arrow-left::before, .e-splitter .e-split-bar.e-split-bar-
horizontal.e-split-bar-active .e-navigate-arrow.e-arrow-left::before, .e-
splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover .e-navigate-
arrow.e-arrow-left::after, .e-splitter .e-split-bar.e-split-bar-
horizontal.e-split-bar-active .e-navigate-arrow.e-arrow-left::after, .e-
splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover .e-navigate-
arrow.e-arrow-right::before, .e-splitter .e-split-bar.e-split-bar-
horizontal.e-split-bar-active .e-navigate-arrow.e-arrow-right::before, .e-
splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover .e-navigate-
arrow.e-arrow-right::after, .e-splitter .e-split-bar.e-split-bar-
horizontal.e-split-bar-active .e-navigate-arrow.e-arrow-right::after {
background-color: green;
}
/* split bar arrows - circular border */
.e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover .e-
navigate-arrow.e-arrow-left, .e-splitter .e-split-bar.e-split-bar-
horizontal.e-split-bar-hover .e-navigate-arrow.e-arrow-right {
border-color: rgba(33, 227, 22, 0.5);
}
```

Vertical split bar resize arrows

CSS

```
/* split bar arrows */
.e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover .e-navigate-
arrow.e-arrow-up::before, .e-splitter .e-split-bar.e-split-bar-vertical.e-
split-bar-active .e-navigate-arrow.e-arrow-up::before, .e-splitter .e-split-
bar.e-split-bar-vertical.e-split-bar-hover .e-navigate-arrow.e-arrow-
up::after, .e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-active
.e-navigate-arrow.e-arrow-up::after, .e-splitter .e-split-bar.e-split-bar-
vertical.e-split-bar-hover .e-navigate-arrow.e-arrow-down::before, .e-
splitter .e-split-bar.e-split-bar-vertical.e-split-bar-active .e-navigate-
arrow.e-arrow-down::before, .e-splitter .e-split-bar.e-split-bar-vertical.e-
split-bar-hover .e-navigate-arrow.e-arrow-down::after, .e-splitter .e-split-
bar.e-split-bar-vertical.e-split-bar-active .e-navigate-arrow.e-arrow-
down::after {
background-color: green;
}
/* split bar arrows - circular border */
```

```
.e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover .e-navigate-arrow.e-arrow-up, .e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover .e-navigate-arrow.e-arrow-down {  
border-color: rgba(33, 227, 22, 0.5);  
}
```

Events in Blazor Splitter Component

This section explains the list of events of the splitter component which will be triggered for appropriate splitter actions.

Created

Created event triggers after creating the splitter component with its panes.

ASPX-CS

```
@using Syncfusion.Blazor.Layouts  
<SfSplitter>  
<SplitterEvents Created="@CreatedHandler" ></SplitterEvents>  
</SfSplitter>  
@code{  
public void CreatedHandler(Object args)  
{  
    // Here you can customize your code  
}  
}
```

OnResizeStart

OnResizeStart event triggers when the split pane is started to resize.

ASPX-CS

```
@using Syncfusion.Blazor.Layouts  
<SfSplitter>  
<SplitterEvents OnResizeStart="@OnResizeStartHandler" ></SplitterEvents>  
</SfSplitter>  
@code{  
public void OnResizeStartHandler(ResizeEventArgs args)  
{  
    // Here you can customize your code  
}  
}
```

OnResizeStop

OnResizeStop event triggers when the resizing of split pane is stopped.

ASPX-CS

```
@using Syncfusion.Blazor.Layouts  
<SfSplitter>  
<SplitterEvents OnResizeStop="@OnResizeStopHandler" ></SplitterEvents>  
</SfSplitter>  
@code{  
public void OnResizeStopHandler(ResizingEventArgs args)  
{  
}
```

```
// Here you can customize your code
}  
}
```

Resizing

Resizing event triggers when a split pane is being resized.

ASPX-CS

```
@using Syncfusion.Blazor.Layouts  
<SfSplitter>  
<SplitterEvents Resizing="@ResizingHandler" ></SplitterEvents>  
</SfSplitter>  
@code{  
public void ResizingHandler(ResizingEventArgs args)  
{  
    // Here you can customize your code  
}  
}
```

OnCollapse

OnCollapse event triggers before the panes get collapsed.

ASPX-CS

```
@using Syncfusion.Blazor.Layouts  
<SfSplitter>  
<SplitterEvents OnCollapse="@OnCollapseHandler" ></SplitterEvents>  
</SfSplitter>  
@code{  
public void OnCollapseHandler(BeforeExpandEventArgs args)  
{  
    // Here you can customize your code  
}  
}
```

OnExpand

OnExpand event triggers before the panes get expanded.

ASPX-CS

```
@using Syncfusion.Blazor.Layouts  
<SfSplitter>  
<SplitterEvents OnExpand="@OnExpandHandler" ></SplitterEvents>  
</SfSplitter>  
@code{  
public void OnExpandHandler(BeforeExpandEventArgs args)  
{  
    // Here you can customize your code  
}  
}
```


Collapsed

Collapsed event triggers after the panes get collapsed.

ASPX-CS

```

@using Syncfusion.Blazor.Layouts
<SfSplitter>
<SplitterEvents Collapsed="@CollapsedHandler" >/SplitterEvents>
</SfSplitter>
@code{
public void CollapsedHandler(ExpandedEventArgs args)
{
// Here you can customize your code
}
}

```

Expanded

Expanded event triggers after the panes get expanded.

ASPX-CS

```

@using Syncfusion.Blazor.Layouts
<SfSplitter>
<SplitterEvents Expanded="@ExpandedHandler" >/SplitterEvents>
</SfSplitter>
@code{
public void ExpandedHandler(ExpandedEventArgs args)
{
// Here you can customize your code
}
}

```

Keyboard interaction

The following key shortcuts can be used to access the Splitter without interruptions:

| Keyboard shortcuts | Actions |

| --- | --- |

| Tab | Helps in focusing the splitter on the page and switching between the consecutive splitter bars. |

| Shift + Tab | Helps in focusing the previous splitter bar element on the splitter. |

| Right arrow | Helps in moving the active Horizontal orientated splitter bar to its Right side. |

| Left arrow | Helps in moving the active Horizontal orientated splitter bar to its Left side. |

| Up arrow | Helps in moving the active Vertical orientated splitter bar to its Up side. |

| Down arrow | Helps in moving the active Vertical orientated splitter bar to its Down side. |

| Enter | Helps to toggle between Expand and Collapse actions of the splitter bar when it is active. |

Stock Chart

Getting Started with Blazor Stock Chart Component

This section briefly explains how to include a Stock Chart component in the Blazor server-side application. Refer to [Getting Started with Syncfusion Blazor for server-side in Visual Studio 2019](#) page for introduction and configuring common specifications.

To get start quickly with Blazor Stock Chart component, check on this video.

{% youtube

"youtube:https://www.youtube.com/watch?v=AxnqK2BnapM"%}

Importing Syncfusion Blazor Stock Chart component in the application

1. Install **Syncfusion.Blazor** NuGet package to the application by using the **NuGet Package Manager**.
2. The client-side resources can be added through [CDN](#) or from [NuGet](#) package in the **HEAD** element of the `~/Pages/_Host.cshtml` page.

HTML

```
<head>
<link href="_content/Syncfusion.Blazor.Themes/bootstrap4.css"
rel="stylesheet" />
<!--CDN-->
@*<link href="https://cdn.syncfusion.com/blazor/{{ site.blazorversion
}}/styles/bootstrap4.css" rel="stylesheet" />*@
</head>
```

For Internet Explorer 11, kindly refer the polyfills. Refer the [documentation](#) for more information.

HTML

```
<head>
<link href="https://cdn.syncfusion.com/blazor/{{ site.blazorversion
}}/styles/bootstrap4.css" rel="stylesheet" />
<script
src="https://github.com/Daddoon/Blazor.Polyfill/releases/download/3.0.1/blaz
or.polyfill.min.js"></script>
</head>
```

3. Now add the lodash script **mandatorily** to the **HEAD** element of the `/Pages/Host.cshtml` page, since it is used in the component.

HTML

```
<head>
<link href="https://cdn.syncfusion.com/blazor/{{ site.blazorversion
}}/styles/bootstrap4.css" rel="stylesheet" />
<script
src="https://github.com/Daddoon/Blazor.Polyfill/releases/download/3.0.1/blaz
or.polyfill.min.js"></script>
```

```
<script
src="https://cdnjs.cloudflare.com/ajax/libs/lodash.js/4.17.20/lodash.min.js"
></script>
</head>
```

Adding component package to the application

Open the `~/Imports.razor` file and include the **Syncfusion.Blazor.Charts** namespace.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
```

Adding SyncfusionBlazor service in Startup.cs

Open the **Startup.cs** file and add services required by Syncfusion components using **services.AddSyncfusionBlazor()** method. Add this method in the **ConfigureServices** function as follows.

CSHARP

```
using Syncfusion.Blazor;
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void ConfigureServices(IServiceCollection services)
        {
            ....
            ....
            services.AddSyncfusionBlazor();
        }
    }
}
```

To enable custom client-side source loading from CRG or CDN, please refer to the section about [custom resources in Blazor application](#).

Adding Stock Chart component

To initialize the Stock Chart component, add the below code to **Index.razor** view page under `~/Pages` folder. In a new application, if **Index.razor** page has any default content template, then those content can be completely removed and following code can be added.

ASPX-CS

```
@page "/"
<SfStockChart>
</SfStockChart>
```

Populate Stock Chart with Data

To bind data for the Stock Chart component, assign a **IEnumerable** object to the [DataSource](#) property. It can also be provided as an instance of the [DataManager](#).

ASPX-CS

```
@code{
public class StockChartData
{
public DateTime Date { get; set; }
public Double Open { get; set; }
public Double Low { get; set; }
public Double Close { get; set; }
public Double High { get; set; }
public Double Volume { get; set; }
}
public List<StockChartData> StockDetails = new List<StockChartData>
{
new StockChartData { Date = new DateTime(2012, 04, 02), Open = 85.9757, High
= 90.6657, Low = 85.7685, Close = 90.5257, Volume = 660187068},
new StockChartData { Date = new DateTime(2012, 04, 09), Open = 89.4471, High
= 92, Low = 86.2157, Close = 86.4614, Volume = 912634864},
new StockChartData { Date = new DateTime(2012, 04, 16), Open = 87.1514, High
= 88.6071, Low = 81.4885, Close = 81.8543, Volume = 1221746066},
new StockChartData { Date = new DateTime(2012, 04, 23), Open = 81.5157, High
= 88.2857, Low = 79.2857, Close = 86.1428, Volume = 965935749},
new StockChartData { Date = new DateTime(2012, 04, 30), Open = 85.4, High =
85.4857, Low = 80.7385, Close = 80.75, Volume = 615249365},
new StockChartData { Date = new DateTime(2012, 05, 07), Open = 80.2143, High
= 82.2685, Low = 79.8185, Close = 80.9585, Volume = 541742692},
new StockChartData { Date = new DateTime(2012, 05, 14), Open = 80.3671, High
= 81.0728, Low = 74.5971, Close = 75.7685, Volume = 708126233},
new StockChartData { Date = new DateTime(2012, 05, 21), Open = 76.3571, High
= 82.3571, Low = 76.2928, Close = 80.3271, Volume = 682076215},
new StockChartData { Date = new DateTime(2012, 05, 28), Open = 81.5571, High
= 83.0714, Low = 80.0743, Close = 80.1414, Volume = 480059584},
};
}
```

Now set the `StockDetails` to `DataSource` property. By default, Stock Chart will be rendered based on the provided `Date` and `High` fields value from datasource without any mapping.

ASPX-CS

```
<SfStockChart>
<StockChartSeriesCollection>
<StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Candle"
XName="Date" YName="Close" High="High" Low="Low" Open="Open" Close="Close"
Volume="Volume"></StockChartSeries>
</StockChartSeriesCollection>
</SfStockChart>
```

On successful compilation of the application, the Syncfusion Blazor Stock Chart component will render in the web browser as shown below.



Adding Title

A title using [Title](#) property can be added in the Stock Chart, to provide quick information to the user about the data plotted in the component.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfStockChart Title="AAPL Historical">
  <StockChartSeriesCollection>
    <StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Candle"
      XName="Date" YName="Close" High="High" Low="Low" Open="Open" Close="Close"
      Volume="Volume"></StockChartSeries>
  </StockChartSeriesCollection>
</SfStockChart>

@code {
  public class ChartData
  {
    public DateTime Date { get; set; }
    public Double Open { get; set; }
    public Double Low { get; set; }
    public Double Close { get; set; }
    public Double High { get; set; }
    public Double Volume { get; set; }
  }
  public List<ChartData> StockDetails = new List<ChartData>
  {
    new ChartData { Date = new DateTime(2012, 04, 02), Open= 85.9757, High =
    90.6657,Low = 85.7685, Close = 90.5257,Volume = 660187068},
    new ChartData { Date = new DateTime(2012, 04, 09), Open= 89.4471, High =
    92,Low = 86.2157, Close = 86.4614,Volume = 912634864},
    new ChartData { Date = new DateTime(2012, 04, 16), Open= 87.1514, High =
    88.6071,Low = 81.4885, Close = 81.8543,Volume = 1221746066},
    new ChartData { Date = new DateTime(2012, 04, 23), Open= 81.5157, High =
    88.2857,Low = 79.2857, Close = 86.1428,Volume = 965935749},
    new ChartData { Date = new DateTime(2012, 04, 30), Open= 85.4, High =
    85.4857,Low = 80.7385, Close = 80.75,Volume = 615249365},
  }
}
```

```

new ChartData { Date = new DateTime(2012, 05, 07), Open= 80.2143, High = 82.2685,Low = 79.8185, Close = 80.9585,Volume = 541742692},
new ChartData { Date = new DateTime(2012, 05, 14), Open= 80.3671, High = 81.0728,Low = 74.5971, Close = 75.7685,Volume = 708126233},
new ChartData { Date = new DateTime(2012, 05, 21), Open= 76.3571, High = 82.3571,Low = 76.2928, Close = 80.3271,Volume = 682076215},
new ChartData { Date = new DateTime(2012, 05, 28), Open= 81.5571, High = 83.0714,Low = 80.0743, Close = 80.1414,Volume = 480059584}
};
}

```



Adding Crosshair

The crosshair is a vertical and horizontal line on the view that shows the value of an axis when the mouse or touch is in a certain position. The crosshair lines can be enabled by using [Enable](#) property in the [StockChartCrosshairSettings](#). Likewise tooltip label for an axis can be enabled by using [Enable](#) property in the [StockChartTooltipSettings](#).

ASPX-CS

```

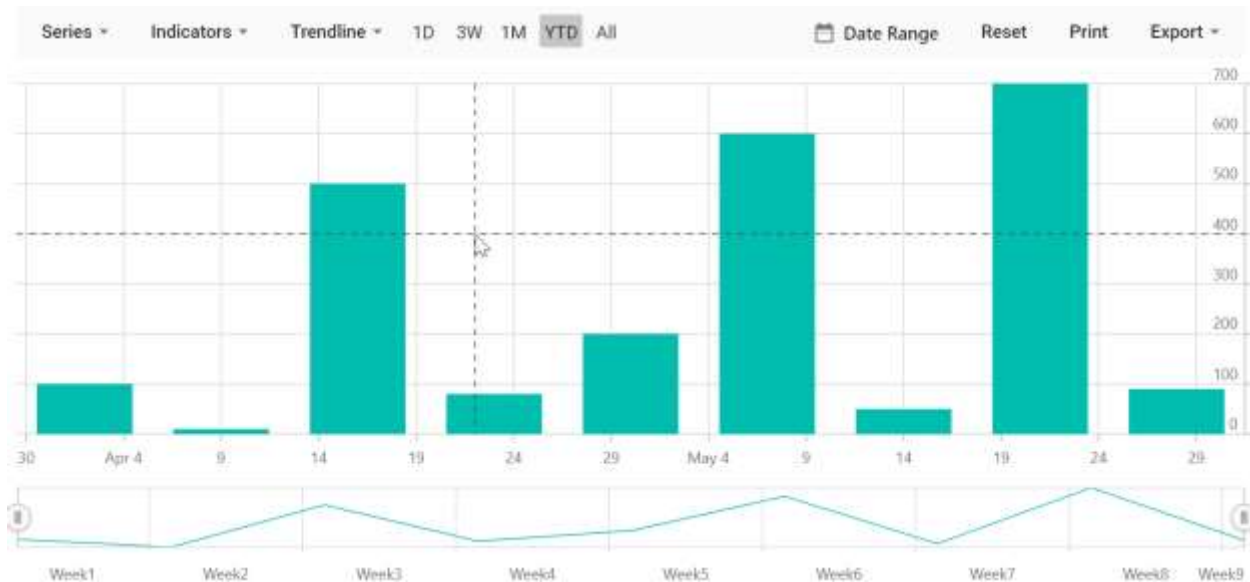
@using Syncfusion.Blazor.Charts
<SfStockChart>
  <StockChartCrosshairSettings Enable="true"></StockChartCrosshairSettings>
  <StockChartSeriesCollection>
    <StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Column"
      XName="Date" YName="Y"></StockChartSeries>
  </StockChartSeriesCollection>
</SfStockChart>
@code {
  public class ChartData
  {
    public DateTime Date { get; set; }
    public Double Y { get; set; }
  }
  public List<ChartData> StockDetails = new List<ChartData>
  {
    new ChartData { Date = new DateTime(2012, 04, 02), Y= 100},

```

```

new ChartData { Date = new DateTime(2012, 04, 09), Y= 10 },
new ChartData { Date = new DateTime(2012, 04, 16), Y= 500},
new ChartData { Date = new DateTime(2012, 04, 23), Y= 80 },
new ChartData { Date = new DateTime(2012, 04, 30), Y= 200},
new ChartData { Date = new DateTime(2012, 05, 07), Y= 600},
new ChartData { Date = new DateTime(2012, 05, 14), Y= 50 },
new ChartData { Date = new DateTime(2012, 05, 21), Y= 700},
new ChartData { Date = new DateTime(2012, 05, 28), Y= 90 }
};
}

```



Adding Trackball

The trackball is used to track a closest data point to the mouse or touch position. The trackball marker indicates the closest point and trackball tooltip displays the information about the point. It can be enabled by setting the [Enable](#) property in the [StockChartCrosshairSettings](#) to **true** and [Shared](#) property in the [StockChartTooltipSettings](#) to **true** in component.

ASPX-CS

```

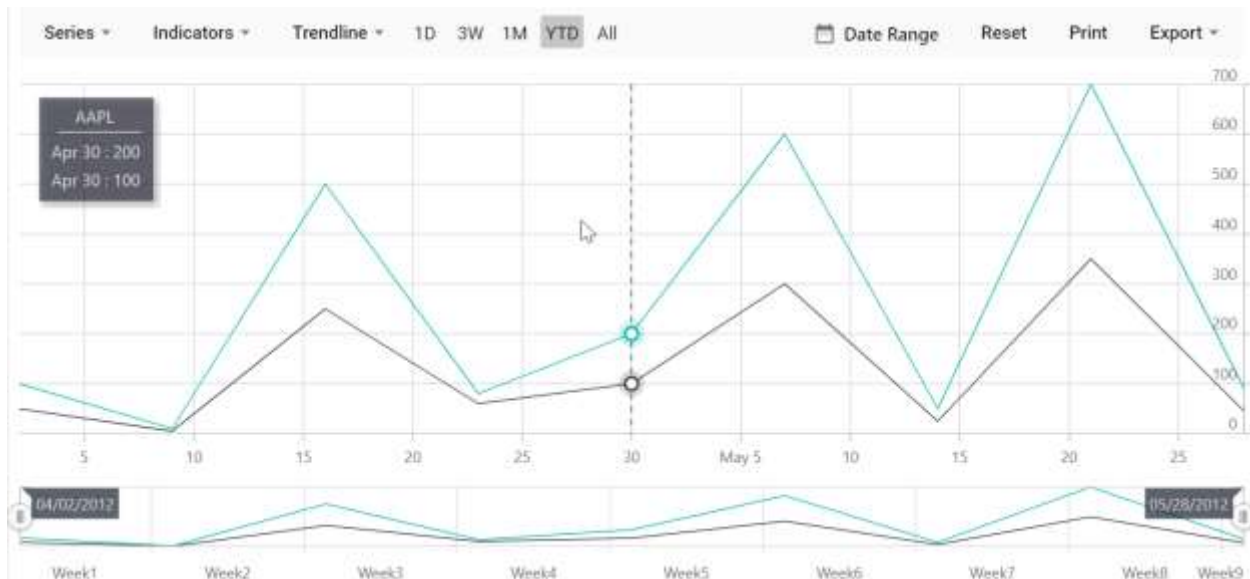
@using Syncfusion.Blazor.Charts
<SfStockChart>
  <StockChartCrosshairSettings Enable="true"
  LineType="LineType.Vertical"></StockChartCrosshairSettings>
  <StockChartTooltipSettings Enable="true" Shared="true" Format="{point.x} :
  {point.y}" Header="AAPL"></StockChartTooltipSettings>
  <StockChartSeriesCollection>
    <StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Line"
    XName="Date" YName="Y"></StockChartSeries>
    <StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Line"
    XName="Date" YName="Y1"></StockChartSeries>
  </StockChartSeriesCollection>
</SfStockChart>
@code {
  public class ChartData
  {
    public DateTime Date { get; set; }

```

```

public Double Y { get; set; }
public Double Y1 { get; set; }
}
public List<ChartData> StockDetails = new List<ChartData>
{
    new ChartData { Date = new DateTime(2012, 04, 02), Y= 100, Y1= 50},
    new ChartData { Date = new DateTime(2012, 04, 09), Y= 10 , Y1= 5},
    new ChartData { Date = new DateTime(2012, 04, 16), Y= 500, Y1= 250},
    new ChartData { Date = new DateTime(2012, 04, 23), Y= 80 , Y1= 60},
    new ChartData { Date = new DateTime(2012, 04, 30), Y= 200, Y1= 100},
    new ChartData { Date = new DateTime(2012, 05, 07), Y= 600, Y1= 300},
    new ChartData { Date = new DateTime(2012, 05, 14), Y= 50 , Y1= 25},
    new ChartData { Date = new DateTime(2012, 05, 21), Y= 700, Y1= 350},
    new ChartData { Date = new DateTime(2012, 05, 28), Y= 90 , Y1= 45}
};
}

```



The fully working sample can be found [here](#).

See also

- [Getting Started with Syncfusion Blazor for Client-Side in .NET Core CLI](#)
- [Getting Started with Syncfusion Blazor for Server-Side in Visual Studio 2019 Preview](#)
- [Getting Started with Syncfusion Blazor for Server-Side in .NET Core CLI](#)

<!-- markdownlint-disable MD036 -->

Working with Data in Blazor Stock Chart Component

Stock Chart can visualize data bound from local or remote data.

Local Data

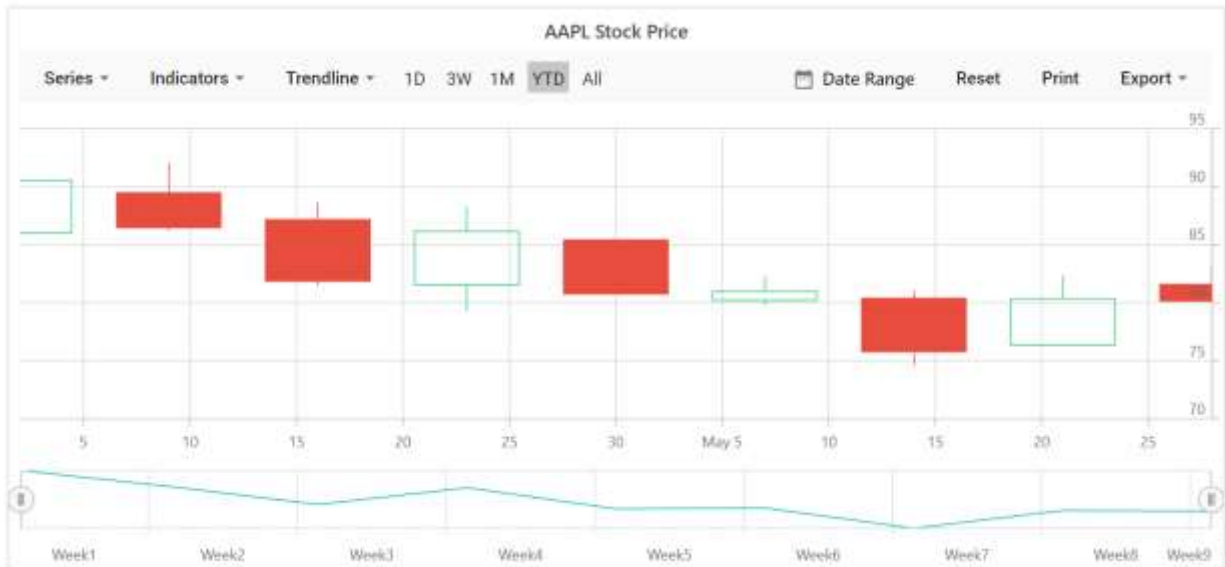
To bind list binding to the stock chart, a `IEnumerable` object can be assigned to the [DataSource](#) property. The list data source can also be provided as an instance of the [SfDataManager](#) or by using [SfDataManager](#) component. Now map the fields in list to

[XName](#), [High](#), [Low](#), [Open](#) and [Close](#)

properties.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfStockChart Title="AAPL Stock Price">
<StockChartSeriesCollection>
<StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Candle"
XName="Date" High="High" Low="Low" Open="Open" Close="Close" Volume="Volume"
Name="Google"></StockChartSeries>
</StockChartSeriesCollection>
</SfStockChart>
@code {
public class ChartData
{
public DateTime Date;
public Double Open;
public Double Low;
public Double Close;
public Double High;
public Double Volume;
}
public List<ChartData> StockDetails = new List<ChartData>
{
new ChartData { Date = new DateTime(2012, 04, 02), Open= 85.9757, High =
90.6657,Low = 85.7685, Close = 90.5257,Volume = 660187068},
new ChartData { Date = new DateTime(2012, 04, 09), Open= 89.4471, High =
92,Low = 86.2157, Close = 86.4614,Volume = 912634864},
new ChartData { Date = new DateTime(2012, 04, 16), Open= 87.1514, High =
88.6071,Low = 81.4885, Close = 81.8543,Volume = 1221746066},
new ChartData { Date = new DateTime(2012, 04, 23), Open= 81.5157, High =
88.2857,Low = 79.2857, Close = 86.1428,Volume = 965935749},
new ChartData { Date = new DateTime(2012, 04, 30), Open= 85.4, High =
85.4857,Low = 80.7385, Close = 80.75,Volume = 615249365},
new ChartData { Date = new DateTime(2012, 05, 07), Open= 80.2143, High =
82.2685,Low = 79.8185, Close = 80.9585,Volume = 541742692},
new ChartData { Date = new DateTime(2012, 05, 14), Open= 80.3671, High =
81.0728,Low = 74.5971, Close = 75.7685,Volume = 708126233},
new ChartData { Date = new DateTime(2012, 05, 21), Open= 76.3571, High =
82.3571,Low = 76.2928, Close = 80.3271,Volume = 682076215},
new ChartData { Date = new DateTime(2012, 05, 28), Open= 81.5571, High =
83.0714,Low = 80.0743, Close = 80.1414,Volume = 480059584}
};
}
```

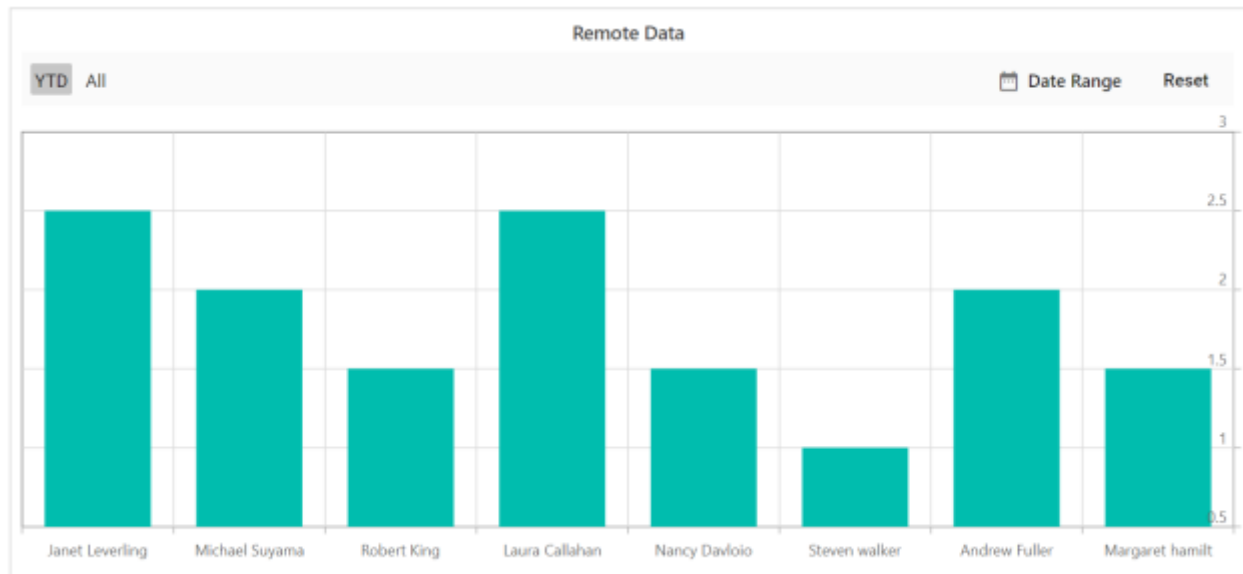


Remote Data

To bind remote data to stock chart component, assign service data as an instance of **SfDataManager** to the **DataSource** property or by using **EjsDataManager** component. To interact with remote data source, provide the endpoint **Url**.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
@using Syncfusion.Blazor.Data
<SfStockChart Title="Remote Data" EnableSelector="false"
TrendlineType="@TrendlineType" IndicatorType="@Indicator"
SeriesType="@SeriesType" ExportType="@ExportType">
<SfDataManager Url="https://mvc.syncfusion.com/Services/Northwnd.svc/Tasks/"
CrossDomain="true"></SfDataManager>
<StockChartPrimaryXAxis
ValueType="Syncfusion.Blazor.Charts.ValueType.Category"></StockChartPrimaryXAxis>
<StockChartPrimaryYAxis Maximum="3"></StockChartPrimaryYAxis>
<StockChartSeriesCollection>
<StockChartSeries XName="Assignee" YName="Estimate" Query="new
ej.data.Query().take(10).where('Estimate', 'lessThan', 3, false)"
Type="ChartSeriesType.Column">
</StockChartSeries>
</StockChartSeriesCollection>
</SfStockChart>
@code {
public List<TrendlineTypes> TrendlineType = new List<TrendlineTypes>();
public List<TechnicalIndicators> Indicator = new
List<TechnicalIndicators>();
public List<ChartSeriesType> SeriesType = new List<ChartSeriesType>();
public List<ExportType> ExportType = new List<ExportType>();
}
```



Entity Framework

Entity Framework acts as a modern object-database mapper for .NET. This section explains how to consume data from the **Microsoft SQL Server** database and bind it to the chart component.

Create DbContext class

The first step is to create a DbContext class called **OrderContext** for establishing connection to a Microsoft SQL Server database.

CSHARP

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.EntityFrameworkCore;
using System.ComponentModel.DataAnnotations;
using EFChart.Data;
namespace EFChart.Data
{
    public class OrderContext : DbContext
    {
        public virtual DbSet<Order> Orders { get; set; }
        protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
        {
            if (!optionsBuilder.IsConfigured)
            {
                // Configures the context to connect to a Microsoft SQL Serve database
                optionsBuilder.UseSqlServer(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename='D:\blazor\EFTreeMap\App_Data
\NORTHWND.MDF';Integrated Security=True;Connect Timeout=30");
            }
        }
    }
    public class Order
    {
        [Key]

```

```
public int? OrderID { get; set; }
[Required]
public string CustomerID { get; set; }
[Required]
public int EmployeeID { get; set; }
}
```

Create data access layer to perform data operation

Now, create a class called **OrderDataAccessLayer**, which acts as a data access layer to retrieve the records from the database table.

CSHARP

```
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using static BlazorApp1.Data.OrderContext;
using EFChart.Data;
namespace EFChart.Data
{
    public class OrderDataAccessLayer
    {
        OrderContext db = new OrderContext();
        //To Get all Orders details
        public DbSet<Order> GetAllOrders()
        {
            try
            {
                return db.Orders;
            }
            catch
            {
                throw;
            }
        }
    }
}
```

Creating Web API Controller

A Web API Controller must be created which allows the chart to directly consume data from the Entity Framework.

CSHARP

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Primitives;
```

```

using static BlazorApp1.Data.OrderContext;
using EFChart.Data;
namespace EFChart.Controller
{
    [Route("api/[controller]")]
    [ApiController]
    public class DefaultController : ControllerBase
    {
        OrderDataAccessLayer db = new OrderDataAccessLayer();
        [HttpGet]
        public object Get()
        {
            IQueryable<Order> data = db.GetAllOrders().AsQueryable();
            var count = data.Count();
            var queryString = Request.Query;
            if (queryString.Keys.Contains("$inlinecount"))
            {
                StringValues Skip;
                StringValues Take;
                int skip = (queryString.TryGetValue("$skip", out Skip)) ?
                    Convert.ToInt32(Skip[0]) : 0;
                int top = (queryString.TryGetValue("$top", out Take)) ?
                    Convert.ToInt32(Take[0]) : data.Count();
                return new { Items = data.Skip(skip).Take(top), Count = count };
            }
            else
            {
                return data;
            }
        }
    }
}

```

Add Web API Controller services in Startup.cs

Open the **Startup.cs** file and add services and endpoints required for Web API Controller as follows.

CSHARP

```

using EFChart.Data;
using Newtonsoft.Json.Serialization;
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void ConfigureServices(IServiceCollection services)
        {
            ....
            ....
            services.AddSingleton<OrderDataAccessLayer>();
            // Adds services for controllers to the specified
            // Microsoft.Extensions.DependencyInjection.IServiceCollection.
            services.AddControllers().AddNewtonsoftJson(options =>
            {
                options.SerializerSettings.ContractResolver = new DefaultContractResolver();
            });
        }
    }
}

```

```

});
}
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    ....
    ....
    app.UseEndpoints(endpoints =>
    {
        // Adds endpoints for controller actions to the
        // Microsoft.AspNetCore.Routing.IEndpointRouteBuilder
        endpoints.MapDefaultControllerRoute();
        ....
        ....
    });
}
}
}
}

```

Configure chart component

Configure the chart to bind data using either [DataSource](#) property or [SfDataManager](#).

For instance, to bind data directly from the data access layer class **OrderDataAccessLayer**, assign the [DataSource](#) property to be **OrderData.GetAllOrders()**.

ASPX-CS

```

@using EFChart.Data;
@inject OrderDataAccessLayer OrderData;
@using Syncfusion.Blazor.Charts
<SfStockChart EnableSelector="false"
DataSource="@OrderData.GetAllOrders()">
<StockChartPrimaryXAxis
ValueType="Syncfusion.Blazor.Charts.ValueType.Category"></StockChartPrimaryX
Axis>
<StockChartSeriesCollection>
<StockChartSeries Type="ChartSeriesType.Column" XName="CustomerID"
YName="OrderID"></StockChartSeries>
</StockChartSeriesCollection>
</SfStockChart>
@code{
}

```

On the other hand, to configure the chart using Web API, provide the appropriate endpoint Url within [SfDataManager](#) along with [Adaptor](#). Here, use [WebApiAdaptor](#) in-order to interact with the Web API to consume data from the Entity Framework appropriately.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
@using Syncfusion.Blazor.Data
<div class="control-section">
<div>
<SfStockChart EnableSelector="false">
<SfDataManager Url="api/Default"
Adaptor="Syncfusion.Blazor.Adaptors.WebApiAdaptor">
</SfDataManager>

```

```

<StockChartPrimaryXAxis
ValueType="Syncfusion.Blazor.Charts.ValueType.Category"></StockChartPrimaryX
Axis>
<StockChartSeriesCollection>
<StockChartSeries Type="ChartSeriesType.Column" XName="CustomerID"
YName="OrderID"></StockChartSeries>
</StockChartSeriesCollection>
</SfStockChart>
</div>
</div>
@code{
}

```

See Also

- [Series Types](#)

Stock Chart Dimensions in Blazor Stock Chart Component

Size for Container

Stock Chart can render to its container size. You can set the size via inline or CSS as demonstrated below.

ASPX-CS

```

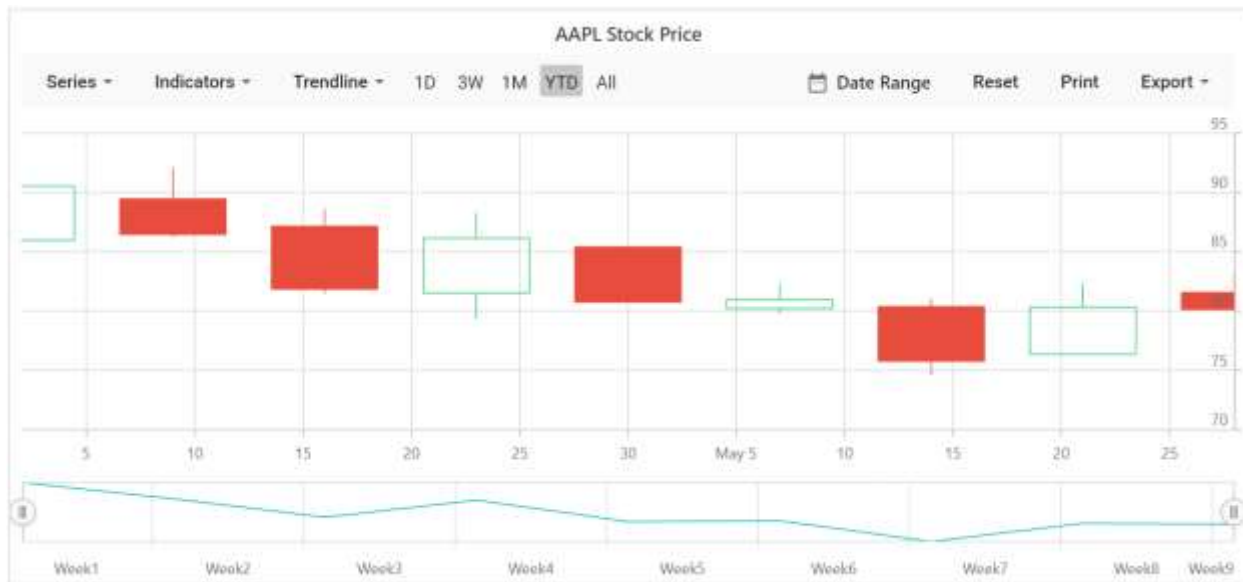
@using Syncfusion.Blazor.Charts
<div style="height:450px;width:970px">
<SfStockChart Title="AAPL Stock Price">
<StockChartSeriesCollection>
<StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Candle"
XName="Date" High="High" Low="Low" Open="Open" Close="Close"
Volume="Volume"></StockChartSeries>
</StockChartSeriesCollection>
</SfStockChart>
</div>
@code {
public class ChartData
{
public DateTime Date;
public Double Open;
public Double Low;
public Double Close;
public Double High;
public Double Volume;
}
public List<ChartData> StockDetails = new List<ChartData>
{
new ChartData { Date = new DateTime(2012, 04, 02), Open= 85.9757, High =
90.6657,Low = 85.7685, Close = 90.5257,Volume = 660187068},
new ChartData { Date = new DateTime(2012, 04, 09), Open= 89.4471, High =
92,Low = 86.2157, Close = 86.4614,Volume = 912634864},
new ChartData { Date = new DateTime(2012, 04, 16), Open= 87.1514, High =
88.6071,Low = 81.4885, Close = 81.8543,Volume = 1221746066},
new ChartData { Date = new DateTime(2012, 04, 23), Open= 81.5157, High =
88.2857,Low = 79.2857, Close = 86.1428,Volume = 965935749},
}
}

```

```

new ChartData { Date = new DateTime(2012, 04, 30), Open= 85.4, High =
85.4857,Low = 80.7385, Close = 80.75,Volume = 615249365},
new ChartData { Date = new DateTime(2012, 05, 07), Open= 80.2143, High =
82.2685,Low = 79.8185, Close = 80.9585,Volume = 541742692},
new ChartData { Date = new DateTime(2012, 05, 14), Open= 80.3671, High =
81.0728,Low = 74.5971, Close = 75.7685,Volume = 708126233},
new ChartData { Date = new DateTime(2012, 05, 21), Open= 76.3571, High =
82.3571,Low = 76.2928, Close = 80.3271,Volume = 682076215},
new ChartData { Date = new DateTime(2012, 05, 28), Open= 81.5571, High =
83.0714,Low = 80.0743, Close = 80.1414,Volume = 480059584}
};
}

```



Size for Stock Chart

The size can be set for stock chart directly through [Width](#) and [Height](#) properties.

<!-- markdownlint-disable MD036 -->

In Pixel

<!-- markdownlint-disable MD036 -->

ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfStockChart Title="AAPL Stock Price" Width="970px" Height="350px">
  <StockChartSeriesCollection>
    <StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Candle"
      XName="Date" High="High" Low="Low" Open="Open" Close="Close"
      Volume="Volume"></StockChartSeries>
  </StockChartSeriesCollection>
</SfStockChart>
@code {
  public class ChartData
  {
    public DateTime Date;
    public Double Open;

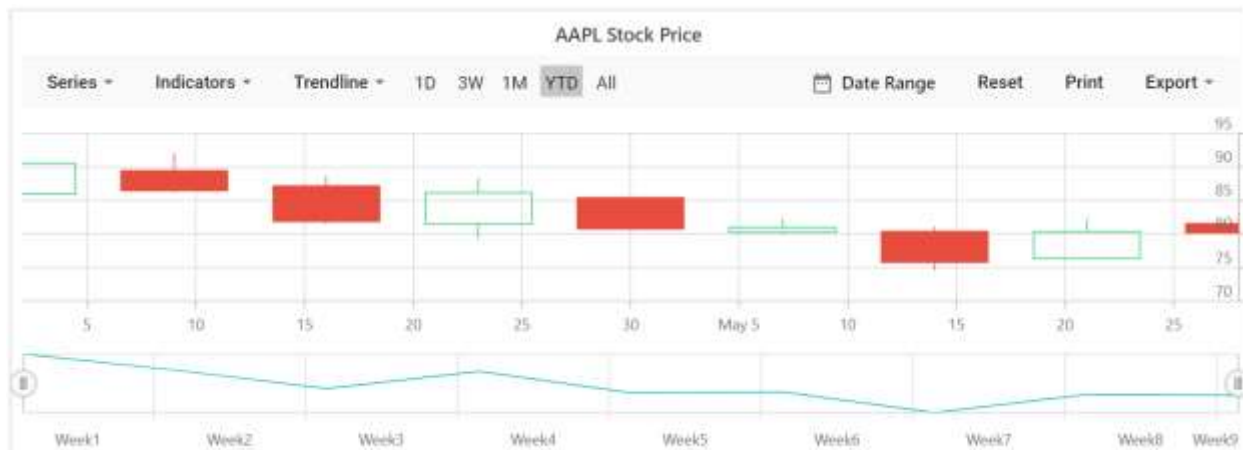
```



```

public Double Low;
public Double Close;
public Double High;
public Double Volume;
}
public List<ChartData> StockDetails = new List<ChartData>
{
    new ChartData { Date = new DateTime(2012, 04, 02), Open= 85.9757, High = 90.6657,Low = 85.7685, Close = 90.5257,Volume = 660187068},
    new ChartData { Date = new DateTime(2012, 04, 09), Open= 89.4471, High = 92,Low = 86.2157, Close = 86.4614,Volume = 912634864},
    new ChartData { Date = new DateTime(2012, 04, 16), Open= 87.1514, High = 88.6071,Low = 81.4885, Close = 81.8543,Volume = 1221746066},
    new ChartData { Date = new DateTime(2012, 04, 23), Open= 81.5157, High = 88.2857,Low = 79.2857, Close = 86.1428,Volume = 965935749},
    new ChartData { Date = new DateTime(2012, 04, 30), Open= 85.4, High = 85.4857,Low = 80.7385, Close = 80.75,Volume = 615249365},
    new ChartData { Date = new DateTime(2012, 05, 07), Open= 80.2143, High = 82.2685,Low = 79.8185, Close = 80.9585,Volume = 541742692},
    new ChartData { Date = new DateTime(2012, 05, 14), Open= 80.3671, High = 81.0728,Low = 74.5971, Close = 75.7685,Volume = 708126233},
    new ChartData { Date = new DateTime(2012, 05, 21), Open= 76.3571, High = 82.3571,Low = 76.2928, Close = 80.3271,Volume = 682076215},
    new ChartData { Date = new DateTime(2012, 05, 28), Open= 81.5571, High = 83.0714,Low = 80.0743, Close = 80.1414,Volume = 480059584}
};
}

```



In Percentage

By setting value in percentage, stock chart gets its dimension with respect to its container. For example, when the height is '50%', chart renders to half of the container height.

ASPX-CS

```

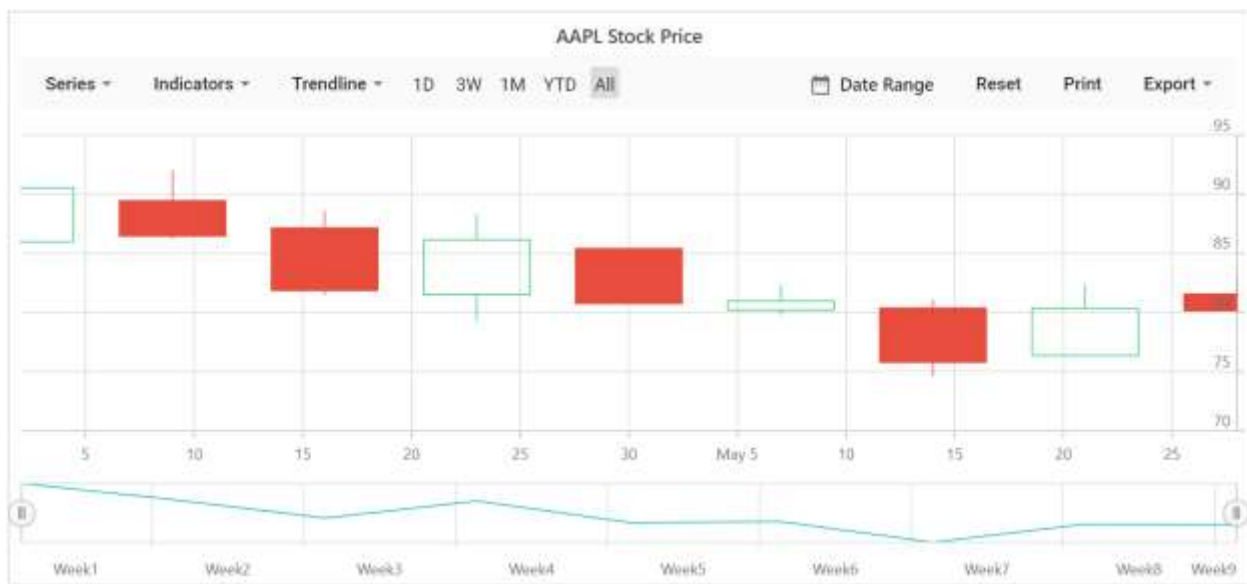
@using Syncfusion.Blazor.Charts
<SfStockChart Title="AAPL Stock Price" Width="100%" Height="90%">
  <StockChartSeriesCollection>
    <StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Candle"
      XName="Date" High="High" Low="Low" Open="Open" Close="Close"
      Volume="Volume"></StockChartSeries>
  </StockChartSeriesCollection>
</SfStockChart>

```

```

</StockChartSeriesCollection>
</SfStockChart>
@code {
public class ChartData
{
public DateTime Date;
public Double Open;
public Double Low;
public Double Close;
public Double High;
public Double Volume;
}
public List<ChartData> StockDetails = new List<ChartData>
{
new ChartData { Date = new DateTime(2012, 04, 02), Open= 85.9757, High =
90.6657,Low = 85.7685, Close = 90.5257,Volume = 660187068},
new ChartData { Date = new DateTime(2012, 04, 09), Open= 89.4471, High =
92,Low = 86.2157, Close = 86.4614,Volume = 912634864},
new ChartData { Date = new DateTime(2012, 04, 16), Open= 87.1514, High =
88.6071,Low = 81.4885, Close = 81.8543,Volume = 1221746066},
new ChartData { Date = new DateTime(2012, 04, 23), Open= 81.5157, High =
88.2857,Low = 79.2857, Close = 86.1428,Volume = 965935749},
new ChartData { Date = new DateTime(2012, 04, 30), Open= 85.4, High =
85.4857,Low = 80.7385, Close = 80.75,Volume = 615249365},
new ChartData { Date = new DateTime(2012, 05, 07), Open= 80.2143, High =
82.2685,Low = 79.8185, Close = 80.9585,Volume = 541742692},
new ChartData { Date = new DateTime(2012, 05, 14), Open= 80.3671, High =
81.0728,Low = 74.5971, Close = 75.7685,Volume = 708126233},
new ChartData { Date = new DateTime(2012, 05, 21), Open= 76.3571, High =
82.3571,Low = 76.2928, Close = 80.3271,Volume = 682076215},
new ChartData { Date = new DateTime(2012, 05, 28), Open= 81.5571, High =
83.0714,Low = 80.0743, Close = 80.1414,Volume = 480059584}
};
}

```



When you do not specify the size, it takes 450px as the height and window size as its width.

<!-- markdownlint-disable MD036 -->

Axis Types in Blazor Stock Chart Component

DateTime Axis

Date time axis uses date time scale and displays the date time values as axis labels in the specified format and set the [ValueType](#) of axis to DateTime.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfStockChart>
  <StockChartPrimaryXAxis
    ValueType="@Syncfusion.Blazor.Charts.ValueType.DateTime">
  </StockChartPrimaryXAxis>
  <StockChartSeriesCollection>
    <StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Column"
      XName="Date" YName="Y">
    </StockChartSeries>
  </StockChartSeriesCollection>
</SfStockChart>

@code{
public class ChartData
{
    public DateTime Date;
    public Double Y;
}
public List<ChartData> StockDetails = new List<ChartData>
{
    new ChartData { Date = new DateTime(2012, 04, 02), Y= 100},
    new ChartData { Date = new DateTime(2012, 04, 09), Y= 10},
    new ChartData { Date = new DateTime(2012, 04, 16), Y= 500},
    new ChartData { Date = new DateTime(2012, 04, 23), Y= 80},
    new ChartData { Date = new DateTime(2012, 04, 30), Y= 200},
    new ChartData { Date = new DateTime(2012, 05, 07), Y= 600},
    new ChartData { Date = new DateTime(2012, 05, 14), Y= 50},
    new ChartData { Date = new DateTime(2012, 05, 21), Y= 700},
    new ChartData { Date = new DateTime(2012, 05, 28), Y= 90}
};
}
```



Logarithmic Axis

<!-- markdownlint-disable MD033 -->

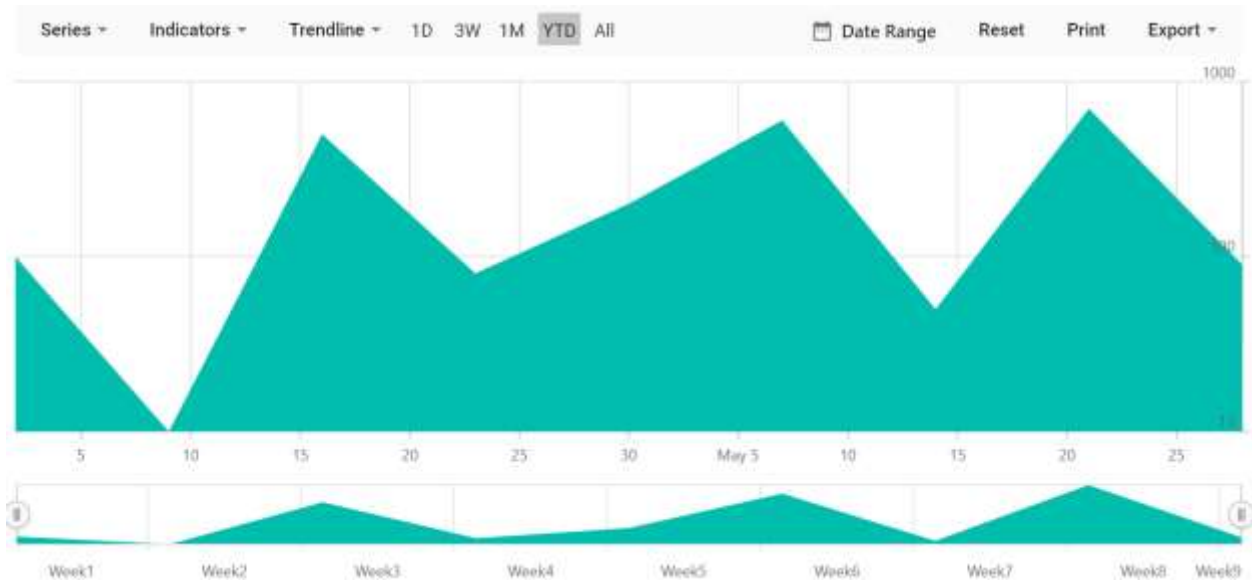
Logarithmic axis uses logarithmic scale and it is very useful in visualizing data, when it has numerical values in both lower order of magnitude (eg: 10^{-6}) and higher order of magnitude (eg: 10^6) and set the [ValueType](#) of axis to **Logarithmic**.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfStockChart>
  <StockChartPrimaryXAxis
    ValueType="@Syncfusion.Blazor.Charts.ValueType.DateTime">
  </StockChartPrimaryXAxis>
  <StockChartPrimaryYAxis
    ValueType="@Syncfusion.Blazor.Charts.ValueType.Logarithmic">
  </StockChartPrimaryYAxis>
  <StockChartSeriesCollection>
    <StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Area"
      XName="Date" YName="Y">
    </StockChartSeries>
  </StockChartSeriesCollection>
</SfStockChart>

@code{
public class ChartData
{
    public DateTime Date;
    public Double Y;
}
public List<ChartData> StockDetails = new List<ChartData>
{
    new ChartData { Date = new DateTime(2012, 04, 02), Y= 100},
    new ChartData { Date = new DateTime(2012, 04, 09), Y= 10},
    new ChartData { Date = new DateTime(2012, 04, 16), Y= 500},
    new ChartData { Date = new DateTime(2012, 04, 23), Y= 80},
    new ChartData { Date = new DateTime(2012, 04, 30), Y= 200},
    new ChartData { Date = new DateTime(2012, 05, 07), Y= 600},
}
```

```
new ChartData { Date = new DateTime(2012, 05, 14), Y= 50},
new ChartData { Date = new DateTime(2012, 05, 21), Y= 700},
new ChartData { Date = new DateTime(2012, 05, 28), Y= 90}
};
}
```



See Also

- [Axis Customization](#)

Axis Customization in Blazor Stock Chart Component

<!-- markdownlint-disable MD034 -->

Title

A title can be added to the axis using the [Title](#) property to provide quick information to the user about the data plotted in the axis. Title style can be customized using the [TitleStyle](#) property of the axis.

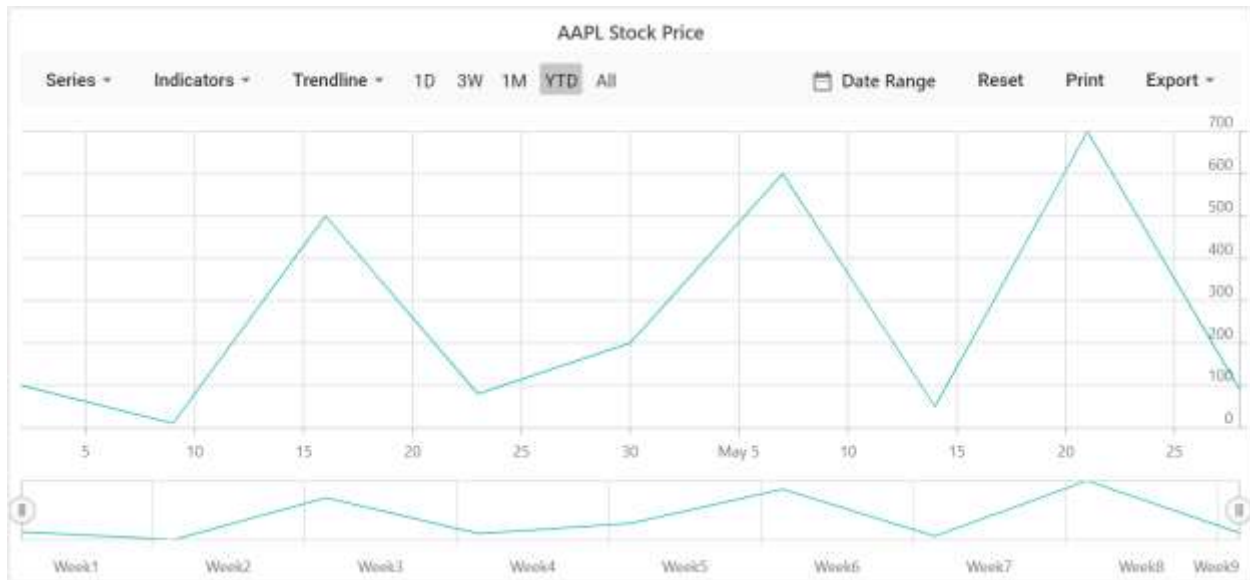
ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfStockChart Title="AAPL Stock Price">
  <StockChartSeriesCollection>
    <StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Line"
      XName="Date" YName="Y"></StockChartSeries>
  </StockChartSeriesCollection>
</SfStockChart>
@code {
  public class ChartData
  {
    public DateTime Date;
    public Double Y;
  }
  public List<ChartData> StockDetails = new List<ChartData>
  {
    new ChartData { Date = new DateTime(2012, 04, 02), Y= 100},
```

```

new ChartData { Date = new DateTime(2012, 04, 09), Y= 10},
new ChartData { Date = new DateTime(2012, 04, 16), Y= 500},
new ChartData { Date = new DateTime(2012, 04, 23), Y= 80},
new ChartData { Date = new DateTime(2012, 04, 30), Y= 200},
new ChartData { Date = new DateTime(2012, 05, 07), Y= 600},
new ChartData { Date = new DateTime(2012, 05, 14), Y= 50},
new ChartData { Date = new DateTime(2012, 05, 21), Y= 700},
new ChartData { Date = new DateTime(2012, 05, 28), Y= 90}
};
}

```



Tick Lines Customization

The **Width**, **Color** and **Size** of the minor and major tick lines can be customized using the [MajorTickLines](#) and the [MinorTickLines](#) properties in the axis.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfStockChart Title="AAPL Stock Price">
  <StockChartPrimaryXAxis>
    <StockChartAxisMajorTickLines Width="5"
    Color="blue"></StockChartAxisMajorTickLines>
    <StockChartAxisMinorTickLines Width="0"
    Color="red"></StockChartAxisMinorTickLines>
  </StockChartPrimaryXAxis>
  <StockChartSeriesCollection>
    <StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Line"
    XName="Date" YName="Y"></StockChartSeries>
  </StockChartSeriesCollection>
</SfStockChart>

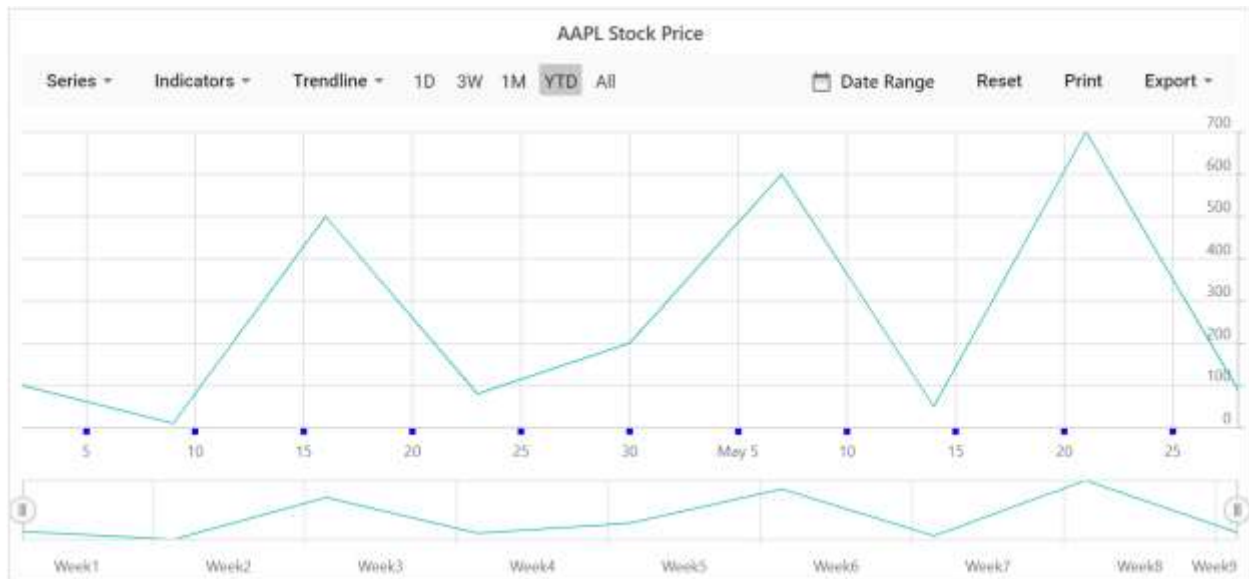
@code {
public class ChartData
{
  public DateTime Date;
  public Double Y;
}

```

```

public List<ChartData> StockDetails = new List<ChartData>
{
    new ChartData { Date = new DateTime(2012, 04, 02), Y= 100},
    new ChartData { Date = new DateTime(2012, 04, 09), Y= 10},
    new ChartData { Date = new DateTime(2012, 04, 16), Y= 500},
    new ChartData { Date = new DateTime(2012, 04, 23), Y= 80},
    new ChartData { Date = new DateTime(2012, 04, 30), Y= 200},
    new ChartData { Date = new DateTime(2012, 05, 07), Y= 600},
    new ChartData { Date = new DateTime(2012, 05, 14), Y= 50},
    new ChartData { Date = new DateTime(2012, 05, 21), Y= 700},
    new ChartData { Date = new DateTime(2012, 05, 28), Y= 90}
};
}

```



Grid Lines Customization

The **Width**, **Color** and **DashArray** of the minor and major grid lines can be customized using the [MajorGridLines](#) and the [MinorGridLines](#) properties in the axis.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfStockChart Title="AAPL Stock Price">
  <StockChartPrimaryXAxis>
    <StockChartAxisMajorGridLines Width="1"
    Color="blue"></StockChartAxisMajorGridLines>
    <StockChartAxisMinorGridLines Width="0"
    Color="red"></StockChartAxisMinorGridLines>
  </StockChartPrimaryXAxis>
  <StockChartPrimaryYAxis>
    <StockChartAxisMajorGridLines Width="1"
    Color="green"></StockChartAxisMajorGridLines>
    <StockChartAxisMinorGridLines Width="0"
    Color="red"></StockChartAxisMinorGridLines>
  </StockChartPrimaryYAxis>
  <StockChartSeriesCollection>

```

```

<StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Line"
XName="Date" YName="Y"></StockChartSeries>
</StockChartSeriesCollection>
</SfStockChart>
@code {
public class ChartData
{
public DateTime Date;
public Double Y;
}
public List<ChartData> StockDetails = new List<ChartData>
{
new ChartData { Date = new DateTime(2012, 04, 02), Y= 100},
new ChartData { Date = new DateTime(2012, 04, 09), Y= 10},
new ChartData { Date = new DateTime(2012, 04, 16), Y= 500},
new ChartData { Date = new DateTime(2012, 04, 23), Y= 80},
new ChartData { Date = new DateTime(2012, 04, 30), Y= 200},
new ChartData { Date = new DateTime(2012, 05, 07), Y= 600},
new ChartData { Date = new DateTime(2012, 05, 14), Y= 50},
new ChartData { Date = new DateTime(2012, 05, 21), Y= 700},
new ChartData { Date = new DateTime(2012, 05, 28), Y= 90}
};
}

```



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.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfStockChart Title="Multiple Axis">
<StockChartAxes>
<StockChartAxis RowIndex="0" Name="yAxis"></StockChartAxis>
</StockChartAxes>
<StockChartPrimaryXAxis>

```



```

<StockChartAxisMajorGridLines Width="0"></StockChartAxisMajorGridLines>
</StockChartPrimaryXAxis>
<StockChartPrimaryYAxis Interval="40">
<StockChartAxisLineStyle Color="Transparent"></StockChartAxisLineStyle>
<StockChartAxisMajorTickLines Width="0"
Color="Transparent"></StockChartAxisMajorTickLines>
</StockChartPrimaryYAxis>
<StockChartCrosshairSettings Enable="true"></StockChartCrosshairSettings>
<StockChartSeriesCollection>
<StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Candle"
XName="Date" High="High" Low="Low" Open="Open" Close="Close"
Name="Apple"></StockChartSeries>
<StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Column"
YAxisName="yAxis" XName="Date" YName="Low" Name="Google"></StockChartSeries>
</StockChartSeriesCollection>
</SfStockChart>
@code {
public class ChartData
{
public DateTime Date;
public Double Open;
public Double Low;
public Double Close;
public Double High;
public Double Volume;
}
public List<ChartData> StockDetails = new List<ChartData>
{
new ChartData { Date = new DateTime(2012, 04, 02), Open= 85.9757, High =
90.6657,Low = 85.7685, Close = 90.5257,Volume = 660187068},
new ChartData { Date = new DateTime(2012, 04, 09), Open= 89.4471, High =
92,Low = 86.2157, Close = 86.4614,Volume = 912634864},
new ChartData { Date = new DateTime(2012, 04, 16), Open= 87.1514, High =
88.6071,Low = 81.4885, Close = 81.8543,Volume = 1221746066},
new ChartData { Date = new DateTime(2012, 04, 23), Open= 81.5157, High =
88.2857,Low = 79.2857, Close = 86.1428,Volume = 965935749},
new ChartData { Date = new DateTime(2012, 04, 30), Open= 85.4, High =
85.4857,Low = 80.7385, Close = 80.75,Volume = 615249365},
new ChartData { Date = new DateTime(2012, 05, 07), Open= 80.2143, High =
82.2685,Low = 79.8185, Close = 80.9585,Volume = 541742692},
new ChartData { Date = new DateTime(2012, 05, 14), Open= 80.3671, High =
81.0728,Low = 74.5971, Close = 75.7685,Volume = 708126233},
new ChartData { Date = new DateTime(2012, 05, 21), Open= 76.3571, High =
82.3571,Low = 76.2928, Close = 80.3271,Volume = 682076215},
new ChartData { Date = new DateTime(2012, 05, 28), Open= 81.5571, High =
83.0714,Low = 80.0743, Close = 80.1414,Volume = 480059584}
};
}

```



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`.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfStockChart Title="Inversed Axis">
  <StockChartPrimaryXAxis IsInversed="true"></StockChartPrimaryXAxis>
  <StockChartSeriesCollection>
    <StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Candle"
      XName="Date" High="High" Low="Low" Open="Open" Close="Close"
      Name="Apple"></StockChartSeries>
  </StockChartSeriesCollection>
</SfStockChart>

@code {
  public class ChartData
  {
    public DateTime Date;
    public Double Open;
    public Double Low;
    public Double Close;
    public Double High;
    public Double Volume;
  }
  public List<ChartData> StockDetails = new List<ChartData>
  {
    new ChartData { Date = new DateTime(2012, 04, 02), Open= 85.9757, High =
    90.6657,Low = 85.7685, Close = 90.5257,Volume = 660187068},
    new ChartData { Date = new DateTime(2012, 04, 09), Open= 89.4471, High =
    92,Low = 86.2157, Close = 86.4614,Volume = 912634864},
    new ChartData { Date = new DateTime(2012, 04, 16), Open= 87.1514, High =
    88.6071,Low = 81.4885, Close = 81.8543,Volume = 1221746066},
    new ChartData { Date = new DateTime(2012, 04, 23), Open= 81.5157, High =
    88.2857,Low = 79.2857, Close = 86.1428,Volume = 965935749},
  }
}
```

```

new ChartData { Date = new DateTime(2012, 04, 30), Open= 85.4, High =
85.4857,Low = 80.7385, Close = 80.75,Volume = 615249365},
new ChartData { Date = new DateTime(2012, 05, 07), Open= 80.2143, High =
82.2685,Low = 79.8185, Close = 80.9585,Volume = 541742692},
new ChartData { Date = new DateTime(2012, 05, 14), Open= 80.3671, High =
81.0728,Low = 74.5971, Close = 75.7685,Volume = 708126233},
new ChartData { Date = new DateTime(2012, 05, 21), Open= 76.3571, High =
82.3571,Low = 76.2928, Close = 80.3271,Volume = 682076215},
new ChartData { Date = new DateTime(2012, 05, 28), Open= 81.5571, High =
83.0714,Low = 80.0743, Close = 80.1414,Volume = 480059584}
};
}

```



Opposed Position

To place an axis opposite from its original position, set [OpposedPosition](#) to true.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfStockChart Title="AAPL Stock Price">
  <StockChartPrimaryXAxis OpposedPosition="true"></StockChartPrimaryXAxis>
  <StockChartSeriesCollection>
    <StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Line"
      XName="Date" High="High" Low="Low" Open="Open"
      Close="Close"></StockChartSeries>
  </StockChartSeriesCollection>
</SfStockChart>
@code {
  public class ChartData
  {
    public DateTime Date;
    public Double Open;
    public Double Low;
    public Double Close;
    public Double High;
    public Double Volume;
  }
}

```

```

}
public List<ChartData> StockDetails = new List<ChartData>
{
    new ChartData { Date = new DateTime(2012, 04, 02), Open= 85.9757, High = 90.6657,Low = 85.7685, Close = 90.5257,Volume = 660187068},
    new ChartData { Date = new DateTime(2012, 04, 09), Open= 89.4471, High = 92,Low = 86.2157, Close = 86.4614,Volume = 912634864},
    new ChartData { Date = new DateTime(2012, 04, 16), Open= 87.1514, High = 88.6071,Low = 81.4885, Close = 81.8543,Volume = 1221746066},
    new ChartData { Date = new DateTime(2012, 04, 23), Open= 81.5157, High = 88.2857,Low = 79.2857, Close = 86.1428,Volume = 965935749},
    new ChartData { Date = new DateTime(2012, 04, 30), Open= 85.4, High = 85.4857,Low = 80.7385, Close = 80.75,Volume = 615249365},
    new ChartData { Date = new DateTime(2012, 05, 07), Open= 80.2143, High = 82.2685,Low = 79.8185, Close = 80.9585,Volume = 541742692},
    new ChartData { Date = new DateTime(2012, 05, 14), Open= 80.3671, High = 81.0728,Low = 74.5971, Close = 75.7685,Volume = 708126233},
    new ChartData { Date = new DateTime(2012, 05, 21), Open= 76.3571, High = 82.3571,Low = 76.2928, Close = 80.3271,Volume = 682076215},
    new ChartData { Date = new DateTime(2012, 05, 28), Open= 81.5571, High = 83.0714,Low = 80.0743, Close = 80.1414,Volume = 480059584}
};
}

```



Series Types in Blazor Stock Chart Component

Stock Chart supports 6 major types of series namely **Line**, **Spline**, **Hilo**, **HiloOpenClose**, **Hollow Candle** and **Candle**. By using the series dropdown button you can navigate between the above listed series types.

<!-- markdownlint-disable MD036 -->

Line

To render a line series, use series **Type** as **Line**.

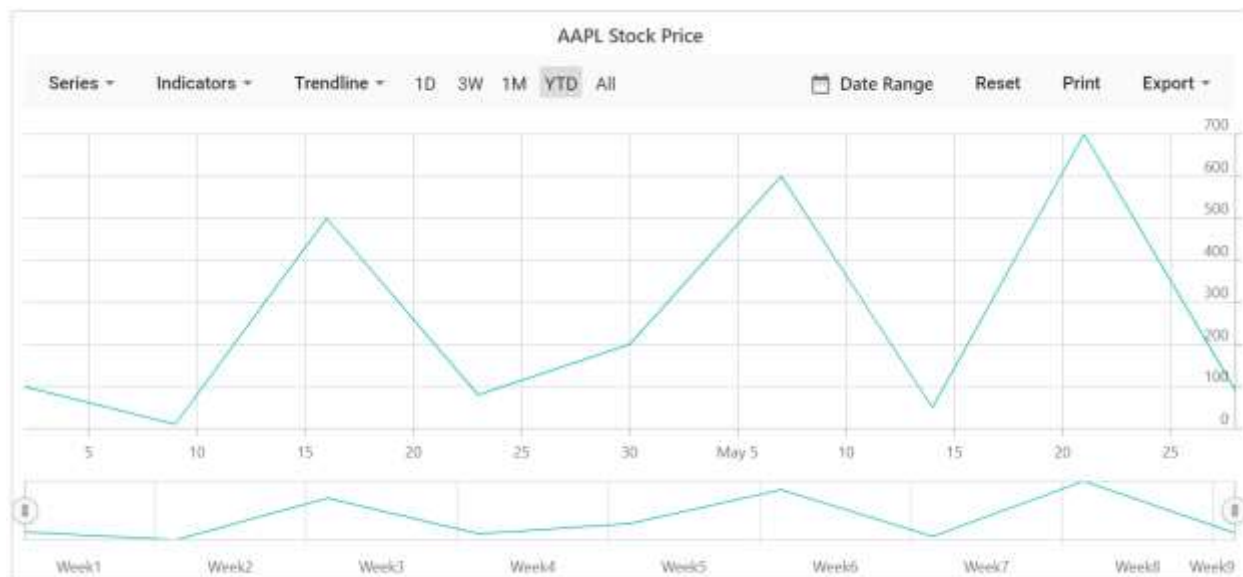
ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfStockChart Title="AAPL Stock Price">
  <StockChartSeriesCollection>
    <StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Line"
      XName="Date" YName="Y">
    </StockChartSeries>
  </StockChartSeriesCollection>
</SfStockChart>

@code {
  public class ChartData
  {
    public DateTime Date;
    public Double Y;
  }
  public List<ChartData> StockDetails = new List<ChartData>
  {
    new ChartData { Date = new DateTime(2012, 04, 02), Y= 100},
    new ChartData { Date = new DateTime(2012, 04, 09), Y= 10},
    new ChartData { Date = new DateTime(2012, 04, 16), Y= 500},
    new ChartData { Date = new DateTime(2012, 04, 23), Y= 80},
    new ChartData { Date = new DateTime(2012, 04, 30), Y= 200},
    new ChartData { Date = new DateTime(2012, 05, 07), Y= 600},
    new ChartData { Date = new DateTime(2012, 05, 14), Y= 50},
    new ChartData { Date = new DateTime(2012, 05, 21), Y= 700},
    new ChartData { Date = new DateTime(2012, 05, 28), Y= 90}
  };
}

```



Spline

To render a spline series, use series [Type](#) as **Spline**.

ASPX-CS

```

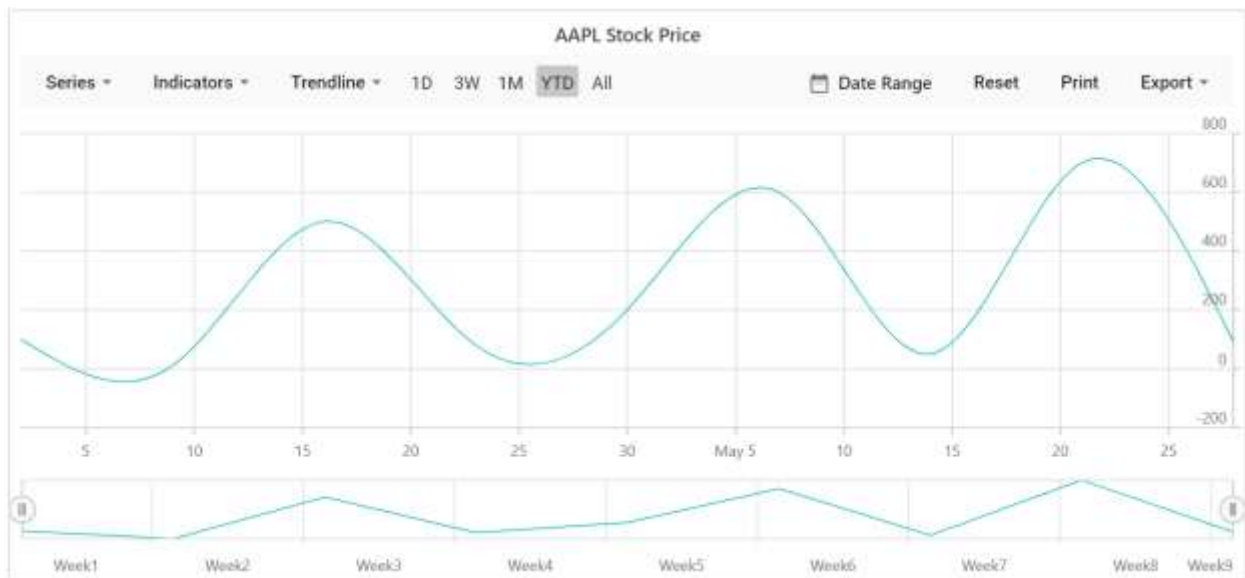
@using Syncfusion.Blazor.Charts
<SfStockChart Title="AAPL Stock Price">
  <StockChartSeriesCollection>

```

```

<StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Spline"
XName="Date" YName="Y">
</StockChartSeries>
</StockChartSeriesCollection>
</SfStockChart>
@code {
public class ChartData
{
public DateTime Date;
public Double Y;
}
public List<ChartData> StockDetails = new List<ChartData>
{
new ChartData { Date = new DateTime(2012, 04, 02), Y= 100},
new ChartData { Date = new DateTime(2012, 04, 09), Y= 10},
new ChartData { Date = new DateTime(2012, 04, 16), Y= 500},
new ChartData { Date = new DateTime(2012, 04, 23), Y= 80},
new ChartData { Date = new DateTime(2012, 04, 30), Y= 200},
new ChartData { Date = new DateTime(2012, 05, 07), Y= 600},
new ChartData { Date = new DateTime(2012, 05, 14), Y= 50},
new ChartData { Date = new DateTime(2012, 05, 21), Y= 700},
new ChartData { Date = new DateTime(2012, 05, 28), Y= 90}
};
}

```



HollowCandle

To render a hollow candle series, use series [Type](#) as `Candle` and set `EnableSolidCandle` as false.

Hilo

To render a hilo series, use series [Type](#) as `Hilo`.

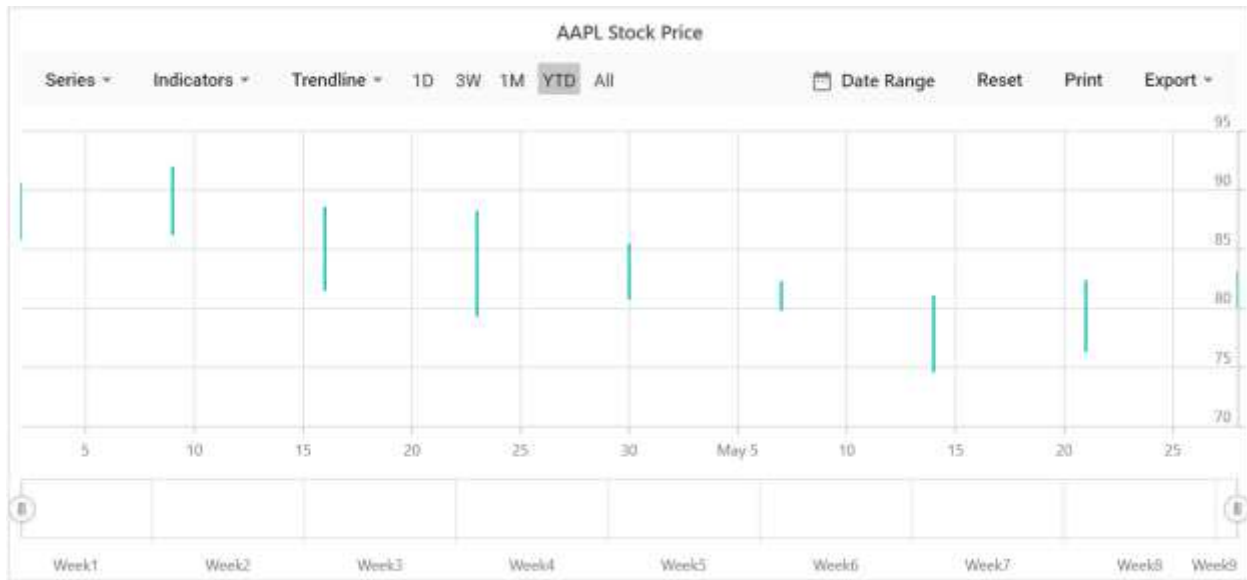
ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfStockChart Title="AAPL Stock Price">

```

```
<StockChartSeriesCollection>
<StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Hilo"
XName="Date" High="High" Low="Low">
</StockChartSeries>
</StockChartSeriesCollection>
</SfStockChart>
@code {
public class StockChartData
{
public DateTime Date;
public Double Open;
public Double Low;
public Double Close;
public Double High;
public Double Volume;
}
public List<StockChartData> StockDetails = new List<StockChartData>
{
new StockChartData { Date = new DateTime(2012, 04, 02), Open = 85.9757, High =
90.6657, Low = 85.7685, Close = 90.5257, Volume = 660187068},
new StockChartData { Date = new DateTime(2012, 04, 09), Open = 89.4471, High =
92, Low = 86.2157, Close = 86.4614, Volume = 912634864},
new StockChartData { Date = new DateTime(2012, 04, 16), Open = 87.1514, High =
88.6071, Low = 81.4885, Close = 81.8543, Volume = 1221746066},
new StockChartData { Date = new DateTime(2012, 04, 23), Open = 81.5157, High =
88.2857, Low = 79.2857, Close = 86.1428, Volume = 965935749},
new StockChartData { Date = new DateTime(2012, 04, 30), Open = 85.4, High =
85.4857, Low = 80.7385, Close = 80.75, Volume = 615249365},
new StockChartData { Date = new DateTime(2012, 05, 07), Open = 80.2143, High =
82.2685, Low = 79.8185, Close = 80.9585, Volume = 541742692},
new StockChartData { Date = new DateTime(2012, 05, 14), Open = 80.3671, High =
81.0728, Low = 74.5971, Close = 75.7685, Volume = 708126233},
new StockChartData { Date = new DateTime(2012, 05, 21), Open = 76.3571, High =
82.3571, Low = 76.2928, Close = 80.3271, Volume = 682076215},
new StockChartData { Date = new DateTime(2012, 05, 28), Open = 81.5571, High =
83.0714, Low = 80.0743, Close = 80.1414, Volume = 480059584},
};
}
```



HiLoOpenClose

To render a HiLoOpenClose series, use series [Type](#) as `HiLoOpenClose`.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfStockChart Title="AAPL Stock Price">
  <StockChartSeriesCollection>
    <StockChartSeries DataSource="@StockDetails"
      Type="ChartSeriesType.HiLoOpenClose" XName="Date" High="High" Low="Low">
    </StockChartSeries>
  </StockChartSeriesCollection>
</SfStockChart>

@code {
  public class StockChartData
  {
    public DateTime Date;
    public Double Open;
    public Double Low;
    public Double Close;
    public Double High;
    public Double Volume;
  }

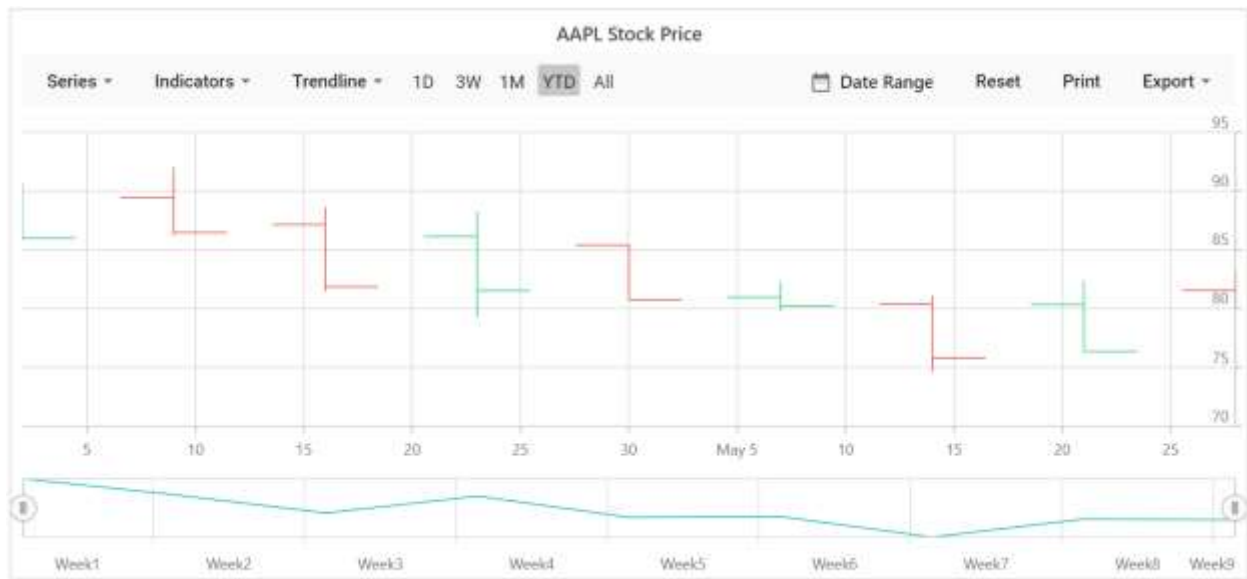
  public List<StockChartData> StockDetails = new List<StockChartData>
  {
    new StockChartData { Date = new DateTime(2012, 04, 02), Open = 85.9757, High = 90.6657, Low = 85.7685, Close = 90.5257, Volume = 660187068},
    new StockChartData { Date = new DateTime(2012, 04, 09), Open = 89.4471, High = 92, Low = 86.2157, Close = 86.4614, Volume = 912634864},
    new StockChartData { Date = new DateTime(2012, 04, 16), Open = 87.1514, High = 88.6071, Low = 81.4885, Close = 81.8543, Volume = 1221746066},
    new StockChartData { Date = new DateTime(2012, 04, 23), Open = 81.5157, High = 88.2857, Low = 79.2857, Close = 86.1428, Volume = 965935749},
    new StockChartData { Date = new DateTime(2012, 04, 30), Open = 85.4, High = 85.4857, Low = 80.7385, Close = 80.75, Volume = 615249365},
    new StockChartData { Date = new DateTime(2012, 05, 07), Open = 80.2143, High = 82.2685, Low = 79.8185, Close = 80.9585, Volume = 541742692},
  }
}
```



```

new StockChartData { Date = new DateTime(2012, 05, 14), Open = 80.3671, High
= 81.0728, Low = 74.5971, Close = 75.7685, Volume = 708126233},
new StockChartData { Date = new DateTime(2012, 05, 21), Open = 76.3571, High
= 82.3571, Low = 76.2928, Close = 80.3271, Volume = 682076215},
new StockChartData { Date = new DateTime(2012, 05, 28), Open = 81.5571, High
= 83.0714, Low = 80.0743, Close = 80.1414, Volume = 480059584},
};
}

```



Candle

To render a candle series, use series [Type](#) as **Candle**.

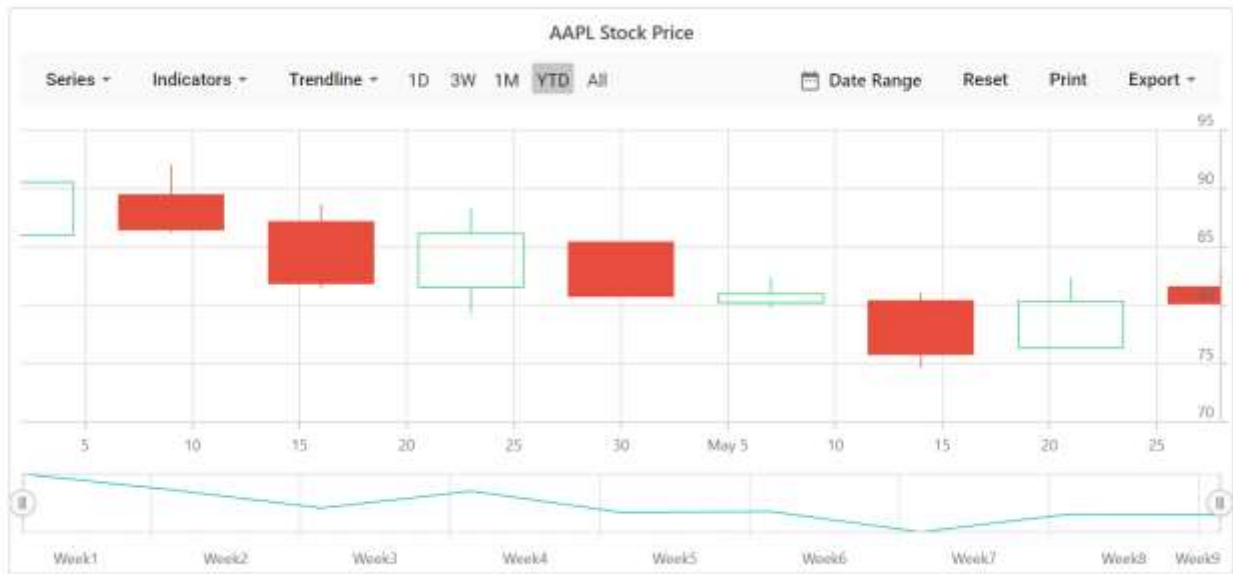
ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfStockChart Title="AAPL Stock Price">
  <StockChartSeriesCollection>
    <StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Candle"
      XName="Date" High="High" Low="Low" Open="Open" Close="Close" Volume="Volume"
      Name="Google"></StockChartSeries>
  </StockChartSeriesCollection>
</SfStockChart>
@code {
  public class ChartData
  {
    public DateTime Date;
    public Double Y;
  }
  public List<ChartData> StockDetails = new List<ChartData>
  {
    new ChartData { Date = new DateTime(2012, 04, 02), Y= 100},
    new ChartData { Date = new DateTime(2012, 04, 09), Y= 10},
    new ChartData { Date = new DateTime(2012, 04, 16), Y= 500},
    new ChartData { Date = new DateTime(2012, 04, 23), Y= 80},
    new ChartData { Date = new DateTime(2012, 04, 30), Y= 200},
    new ChartData { Date = new DateTime(2012, 05, 07), Y= 600},
  }
}

```

```
new ChartData { Date = new DateTime(2012, 05, 14), Y= 50},
new ChartData { Date = new DateTime(2012, 05, 21), Y= 700},
new ChartData { Date = new DateTime(2012, 05, 28), Y= 90}
};
}
```



Trendlines in Blazor Stock Chart Component

Trendlines are used to show the direction and speed of price. Stock Chart supports 6 types of trendlines namely **Linear**, **Exponential**, **Logarithmic**, **Polynomial**, **Power**, **Moving Average**. By using trendline dropdown button, the required trendline type can be added or removed.

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**.

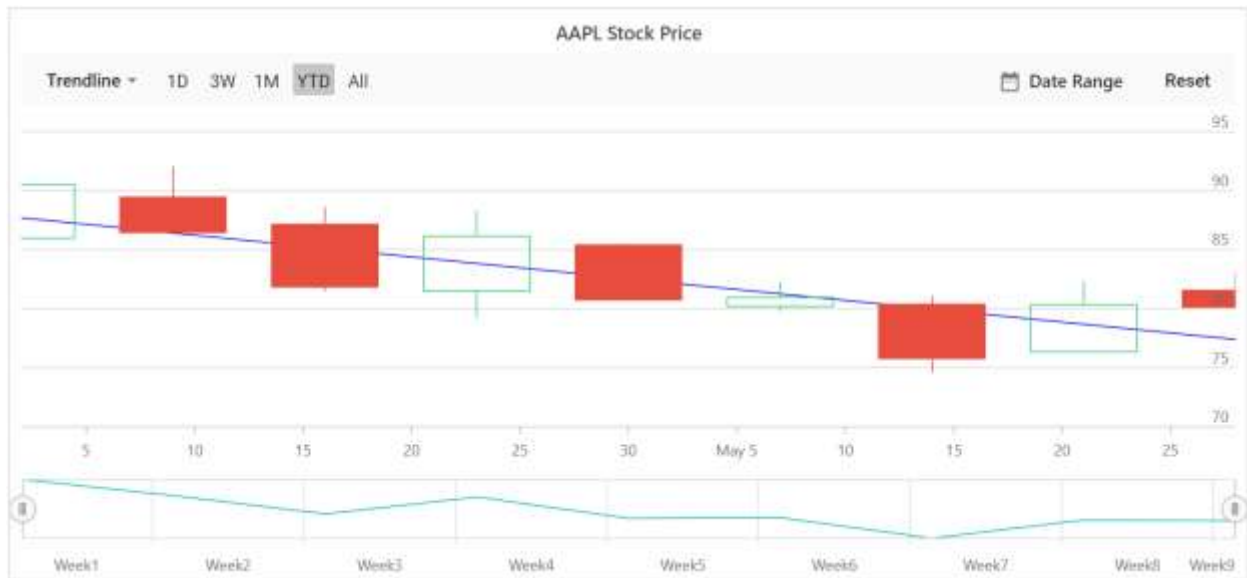
ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfStockChart Title="AAPL Stock Price" SeriesType="@SeriesType"
IndicatorType="@Indicator" ExportType="@ExportType">
<StockChartPrimaryXAxis>
<StockChartAxisMajorGridLines
Color="Transparent"></StockChartAxisMajorGridLines>
</StockChartPrimaryXAxis>
<StockChartPrimaryYAxis>
<StockChartAxisLineStyle Color="Transparent"></StockChartAxisLineStyle>
<StockChartAxisMajorTickLines Color="Transparent"
Width="0"></StockChartAxisMajorTickLines>
</StockChartPrimaryYAxis>
<StockChartSeriesCollection>
<StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Candle"
XName="Date" High="High" Low="Low" Open="Open" Close="Close" Volume="Volume"
Name="Google">
<StockChartTrendlines>
```

```

<StockChartTrendline Type="TrendlineTypes.Linear"
EnableTooltip="false"></StockChartTrendline>
</StockChartTrendlines>
</StockChartSeries>
</StockChartSeriesCollection>
</SfStockChart>
@code {
public List<TechnicalIndicators> Indicator = new
List<TechnicalIndicators>();
public List<ChartSeriesType> SeriesType = new List<ChartSeriesType>();
public List<ExportType> ExportType = new List<ExportType>();
public class StockChartData
{
public DateTime Date;
public Double Open;
public Double Low;
public Double Close;
public Double High;
public Double Volume;
}
public List<StockChartData> StockDetails = new List<StockChartData>
{
new StockChartData { Date = new DateTime(2012, 04, 02), Open = 85.9757, High
= 90.6657, Low = 85.7685, Close = 90.5257, Volume = 660187068},
new StockChartData { Date = new DateTime(2012, 04, 09), Open = 89.4471, High
= 92, Low = 86.2157, Close = 86.4614, Volume = 912634864},
new StockChartData { Date = new DateTime(2012, 04, 16), Open = 87.1514, High
= 88.6071, Low = 81.4885, Close = 81.8543, Volume = 1221746066},
new StockChartData { Date = new DateTime(2012, 04, 23), Open = 81.5157, High
= 88.2857, Low = 79.2857, Close = 86.1428, Volume = 965935749},
new StockChartData { Date = new DateTime(2012, 04, 30), Open = 85.4, High =
85.4857, Low = 80.7385, Close = 80.75, Volume = 615249365},
new StockChartData { Date = new DateTime(2012, 05, 07), Open = 80.2143, High
= 82.2685, Low = 79.8185, Close = 80.9585, Volume = 541742692},
new StockChartData { Date = new DateTime(2012, 05, 14), Open = 80.3671, High
= 81.0728, Low = 74.5971, Close = 75.7685, Volume = 708126233},
new StockChartData { Date = new DateTime(2012, 05, 21), Open = 76.3571, High
= 82.3571, Low = 76.2928, Close = 80.3271, Volume = 682076215},
new StockChartData { Date = new DateTime(2012, 05, 28), Open = 81.5571, High
= 83.0714, Low = 80.0743, Close = 80.1414, Volume = 480059584},
};
}

```



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`.

ASPX-CS

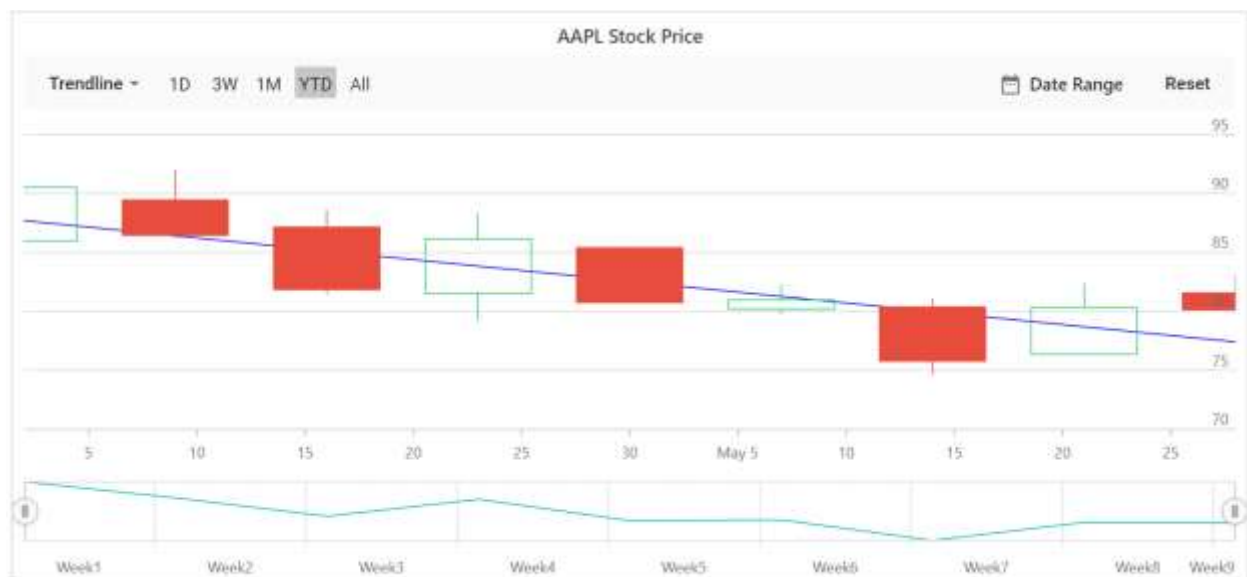
```
@using Syncfusion.Blazor.Charts
<SfStockChart Title="AAPL Stock Price" SeriesType="@SeriesType"
IndicatorType="@Indicator" ExportType="@ExportType">
  <StockChartPrimaryXAxis>
    <StockChartAxisMajorGridLines
      Color="Transparent"></StockChartAxisMajorGridLines>
    </StockChartPrimaryXAxis>
    <StockChartPrimaryYAxis>
      <StockChartAxisLineStyle Color="Transparent"></StockChartAxisLineStyle>
      <StockChartAxisMajorTickLines Color="Transparent"
        Width="0"></StockChartAxisMajorTickLines>
    </StockChartPrimaryYAxis>
    <StockChartSeriesCollection>
      <StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Candle"
        XName="Date" High="High" Low="Low" Open="Open" Close="Close" Volume="Volume"
        Name="Google">
        <StockChartTrendlines>
          <StockChartTrendline Type="TrendlineTypes.Logarithmic"
            EnableTooltip="false"></StockChartTrendline>
        </StockChartTrendlines>
      </StockChartSeries>
    </StockChartSeriesCollection>
  </SfStockChart>
@code {
  public List<TechnicalIndicators> Indicator = new
    List<TechnicalIndicators>();
  public List<ChartSeriesType> SeriesType = new List<ChartSeriesType>();
  public List<ExportType> ExportType = new List<ExportType>();
```

```

public class StockChartData
{
    public DateTime Date;
    public Double Open;
    public Double Low;
    public Double Close;
    public Double High;
    public Double Volume;
}

public List<StockChartData> StockDetails = new List<StockChartData>
{
    new StockChartData { Date = new DateTime(2012, 04, 02), Open = 85.9757, High = 90.6657, Low = 85.7685, Close = 90.5257, Volume = 660187068},
    new StockChartData { Date = new DateTime(2012, 04, 09), Open = 89.4471, High = 92, Low = 86.2157, Close = 86.4614, Volume = 912634864},
    new StockChartData { Date = new DateTime(2012, 04, 16), Open = 87.1514, High = 88.6071, Low = 81.4885, Close = 81.8543, Volume = 1221746066},
    new StockChartData { Date = new DateTime(2012, 04, 23), Open = 81.5157, High = 88.2857, Low = 79.2857, Close = 86.1428, Volume = 965935749},
    new StockChartData { Date = new DateTime(2012, 04, 30), Open = 85.4, High = 85.4857, Low = 80.7385, Close = 80.75, Volume = 615249365},
    new StockChartData { Date = new DateTime(2012, 05, 07), Open = 80.2143, High = 82.2685, Low = 79.8185, Close = 80.9585, Volume = 541742692},
    new StockChartData { Date = new DateTime(2012, 05, 14), Open = 80.3671, High = 81.0728, Low = 74.5971, Close = 75.7685, Volume = 708126233},
    new StockChartData { Date = new DateTime(2012, 05, 21), Open = 76.3571, High = 82.3571, Low = 76.2928, Close = 80.3271, Volume = 682076215},
    new StockChartData { Date = new DateTime(2012, 05, 28), Open = 81.5571, High = 83.0714, Low = 80.0743, Close = 80.1414, Volume = 480059584},
};
}

```



Exponential

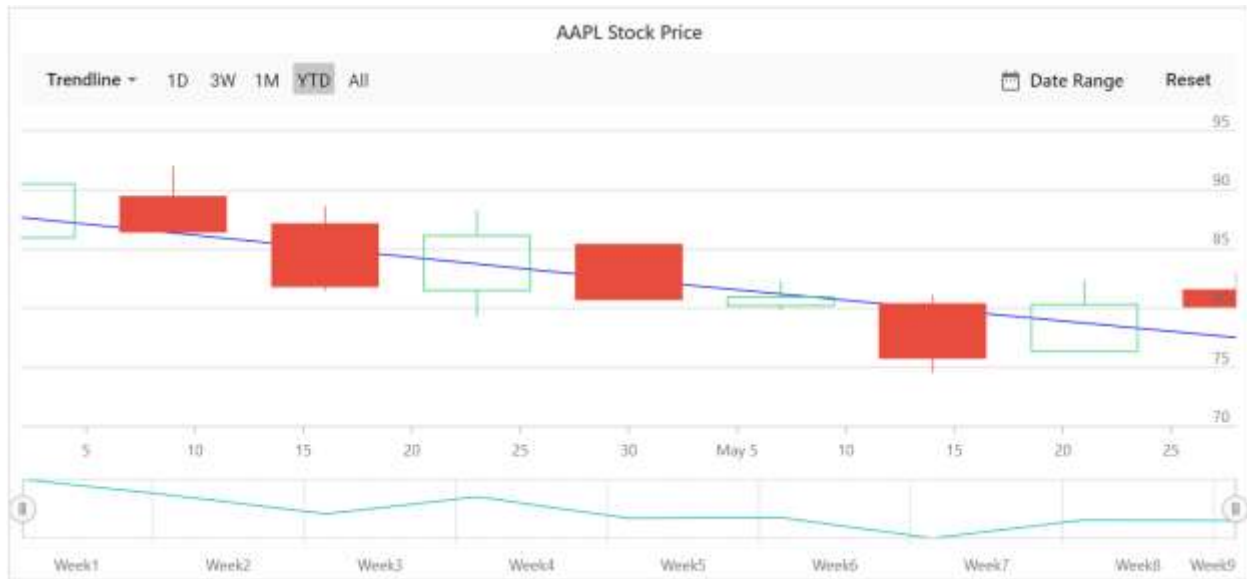
An exponential trendline is a curved line that is most useful when data values rise or fall at increasingly higher rates. You cannot create an exponential trendline, if your data contains zero or negative values.

To render an exponential trendline, use trendline [Type](#) as **Exponential**.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfStockChart Title="AAPL Stock Price" SeriesType="@SeriesType"
IndicatorType="@Indicator" ExportType="@ExportType">
  <StockChartPrimaryXAxis>
  <StockChartAxisMajorGridLines
Color="Transparent"></StockChartAxisMajorGridLines>
</StockChartPrimaryXAxis>
<StockChartPrimaryYAxis>
<StockChartAxisLineStyle Color="Transparent"></StockChartAxisLineStyle>
<StockChartAxisMajorTickLines Color="Transparent"
Width="0"></StockChartAxisMajorTickLines>
</StockChartPrimaryYAxis>
<StockChartSeriesCollection>
<StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Candle"
XName="Date" High="High" Low="Low" Open="Open" Close="Close" Volume="Volume"
Name="Google">
  <StockChartTrendlines>
  <StockChartTrendline Type="TrendlineTypes.Exponential"
EnableTooltip="false"></StockChartTrendline>
</StockChartTrendlines>
</StockChartSeries>
</StockChartSeriesCollection>
</SfStockChart>
@code {
public List<TechnicalIndicators> Indicator = new
List<TechnicalIndicators>();
public List<ChartSeriesType> SeriesType = new List<ChartSeriesType>();
public List<ExportType> ExportType = new List<ExportType>();
public class StockChartData
{
public DateTime Date;
public Double Open;
public Double Low;
public Double Close;
public Double High;
public Double Volume;
}
public List<StockChartData> StockDetails = new List<StockChartData>
{
new StockChartData { Date = new DateTime(2012, 04, 02), Open = 85.9757, High
= 90.6657, Low = 85.7685, Close = 90.5257, Volume = 660187068},
new StockChartData { Date = new DateTime(2012, 04, 09), Open = 89.4471, High
= 92, Low = 86.2157, Close = 86.4614, Volume = 912634864},
new StockChartData { Date = new DateTime(2012, 04, 16), Open = 87.1514, High
= 88.6071, Low = 81.4885, Close = 81.8543, Volume = 1221746066},
new StockChartData { Date = new DateTime(2012, 04, 23), Open = 81.5157, High
= 88.2857, Low = 79.2857, Close = 86.1428, Volume = 965935749},
new StockChartData { Date = new DateTime(2012, 04, 30), Open = 85.4, High =
85.4857, Low = 80.7385, Close = 80.75, Volume = 615249365},
new StockChartData { Date = new DateTime(2012, 05, 07), Open = 80.2143, High
= 82.2685, Low = 79.8185, Close = 80.9585, Volume = 541742692},
new StockChartData { Date = new DateTime(2012, 05, 14), Open = 80.3671, High
= 81.0728, Low = 74.5971, Close = 75.7685, Volume = 708126233},
```

```
new StockChartData { Date = new DateTime(2012, 05, 21), Open = 76.3571, High = 82.3571, Low = 76.2928, Close = 80.3271, Volume = 682076215},
new StockChartData { Date = new DateTime(2012, 05, 28), Open = 81.5571, High = 83.0714, Low = 80.0743, Close = 80.1414, Volume = 480059584},
};
}
```



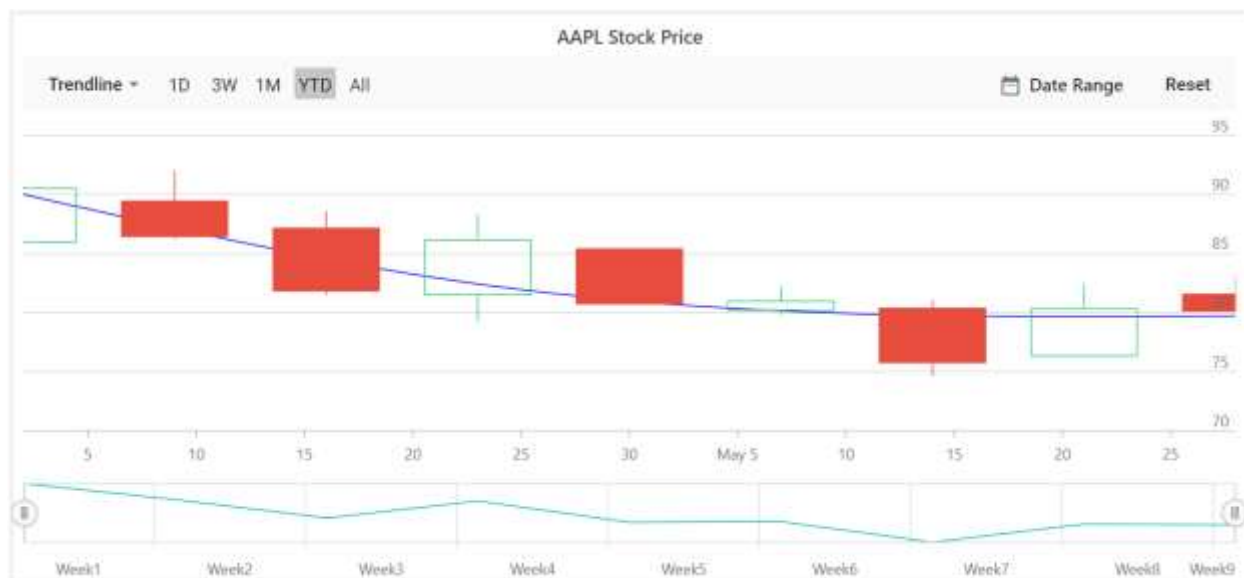
Polynomial

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.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfStockChart Title="AAPL Stock Price" SeriesType="@SeriesType"
IndicatorType="@Indicator" ExportType="@ExportType">
  <StockChartPrimaryXAxis>
    <StockChartAxisMajorGridLines
      Color="Transparent"></StockChartAxisMajorGridLines>
    </StockChartPrimaryXAxis>
    <StockChartPrimaryYAxis>
      <StockChartAxisLineStyle Color="Transparent"></StockChartAxisLineStyle>
      <StockChartAxisMajorTickLines Color="Transparent"
        Width="0"></StockChartAxisMajorTickLines>
    </StockChartPrimaryYAxis>
    <StockChartSeriesCollection>
      <StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Candle"
        XName="Date" High="High" Low="Low" Open="Open" Close="Close" Volume="Volume"
        Name="Google">
        <StockChartTrendlines>
          <StockChartTrendline Type="TrendlineTypes.Polynomial"
            EnableTooltip="false"></StockChartTrendline>
        </StockChartTrendlines>
      </StockChartSeries>
    </StockChartSeriesCollection>
  </SfStockChart>
```

```
@code {
    public List<TechnicalIndicators> Indicator = new
    List<TechnicalIndicators>();
    public List<ChartSeriesType> SeriesType = new List<ChartSeriesType>();
    public List<ExportType> ExportType = new List<ExportType>();
    public class StockChartData
    {
        public DateTime Date;
        public Double Open;
        public Double Low;
        public Double Close;
        public Double High;
        public Double Volume;
    }
    public List<StockChartData> StockDetails = new List<StockChartData>
    {
        new StockChartData { Date = new DateTime(2012, 04, 02), Open = 85.9757, High = 90.6657, Low = 85.7685, Close = 90.5257, Volume = 660187068},
        new StockChartData { Date = new DateTime(2012, 04, 09), Open = 89.4471, High = 92, Low = 86.2157, Close = 86.4614, Volume = 912634864},
        new StockChartData { Date = new DateTime(2012, 04, 16), Open = 87.1514, High = 88.6071, Low = 81.4885, Close = 81.8543, Volume = 1221746066},
        new StockChartData { Date = new DateTime(2012, 04, 23), Open = 81.5157, High = 88.2857, Low = 79.2857, Close = 86.1428, Volume = 965935749},
        new StockChartData { Date = new DateTime(2012, 04, 30), Open = 85.4, High = 85.4857, Low = 80.7385, Close = 80.75, Volume = 615249365},
        new StockChartData { Date = new DateTime(2012, 05, 07), Open = 80.2143, High = 82.2685, Low = 79.8185, Close = 80.9585, Volume = 541742692},
        new StockChartData { Date = new DateTime(2012, 05, 14), Open = 80.3671, High = 81.0728, Low = 74.5971, Close = 75.7685, Volume = 708126233},
        new StockChartData { Date = new DateTime(2012, 05, 21), Open = 76.3571, High = 82.3571, Low = 76.2928, Close = 80.3271, Volume = 682076215},
        new StockChartData { Date = new DateTime(2012, 05, 28), Open = 81.5571, High = 83.0714, Low = 80.0743, Close = 80.1414, Volume = 480059584},
    };
}
```



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](#).

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.

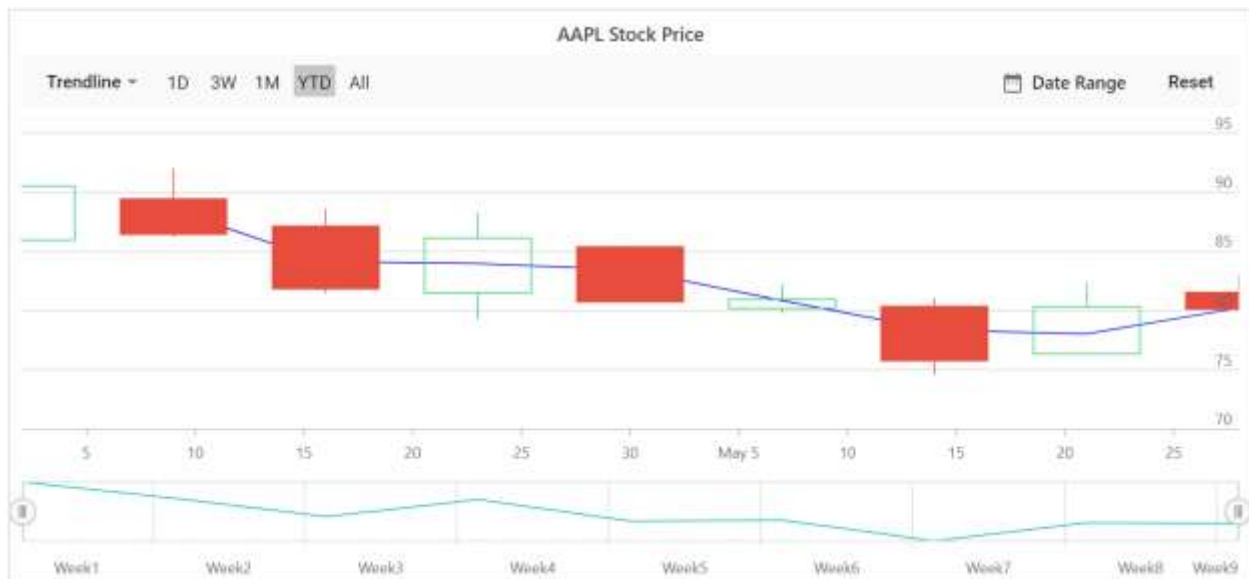
ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfStockChart Title="AAPL Stock Price" SeriesType="@SeriesType"
IndicatorType="@Indicator" ExportType="@ExportType">
  <StockChartPrimaryXAxis>
  <StockChartAxisMajorGridLines
Color="Transparent"></StockChartAxisMajorGridLines>
</StockChartPrimaryXAxis>
<StockChartPrimaryYAxis>
<StockChartAxisLineStyle Color="Transparent"></StockChartAxisLineStyle>
<StockChartAxisMajorTickLines Color="Transparent"
Width="0"></StockChartAxisMajorTickLines>
</StockChartPrimaryYAxis>
<StockChartSeriesCollection>
<StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Candle"
XName="Date" High="High" Low="Low" Open="Open" Close="Close" Volume="Volume"
Name="Google">
<StockChartTrendlines>
<StockChartTrendline Type="TrendlineTypes.MovingAverage"
EnableTooltip="false"></StockChartTrendline>
</StockChartTrendlines>
</StockChartSeries>
</StockChartSeriesCollection>
</SfStockChart>
@code {
public List<TechnicalIndicators> Indicator = new
List<TechnicalIndicators>();
public List<ChartSeriesType> SeriesType = new List<ChartSeriesType>();
public List<ExportType> ExportType = new List<ExportType>();
public class ChartData
{
public DateTime Date;
public Double Open;
public Double Low;
public Double Close;
public Double High;
public Double Volume;
}
public List<ChartData> StockDetails = new List<ChartData>
{
new ChartData { Date = new DateTime(2012, 04, 02), Open= 85.9757, High =
90.6657,Low = 85.7685, Close = 90.5257,Volume = 660187068},
new ChartData { Date = new DateTime(2012, 04, 09), Open= 89.4471, High =
92,Low = 86.2157, Close = 86.4614,Volume = 912634864},
```

```

new ChartData { Date = new DateTime(2012, 04, 16), Open= 87.1514, High =
88.6071,Low = 81.4885, Close = 81.8543,Volume = 1221746066},
new ChartData { Date = new DateTime(2012, 04, 23), Open= 81.5157, High =
88.2857,Low = 79.2857, Close = 86.1428,Volume = 965935749},
new ChartData { Date = new DateTime(2012, 04, 30), Open= 85.4, High =
85.4857,Low = 80.7385, Close = 80.75,Volume = 615249365},
new ChartData { Date = new DateTime(2012, 05, 07), Open= 80.2143, High =
82.2685,Low = 79.8185, Close = 80.9585,Volume = 541742692},
new ChartData { Date = new DateTime(2012, 05, 14), Open= 80.3671, High =
81.0728,Low = 74.5971, Close = 75.7685,Volume = 708126233},
new ChartData { Date = new DateTime(2012, 05, 21), Open= 76.3571, High =
82.3571,Low = 76.2928, Close = 80.3271,Volume = 682076215},
new ChartData { Date = new DateTime(2012, 05, 28), Open= 81.5571, High =
83.0714,Low = 80.0743, Close = 80.1414,Volume = 480059584}
};
}

```



Customization of Trendline

The [Fill](#) and [Width](#) properties are used to customize the appearance of the trendline.

ASPX-CS

```

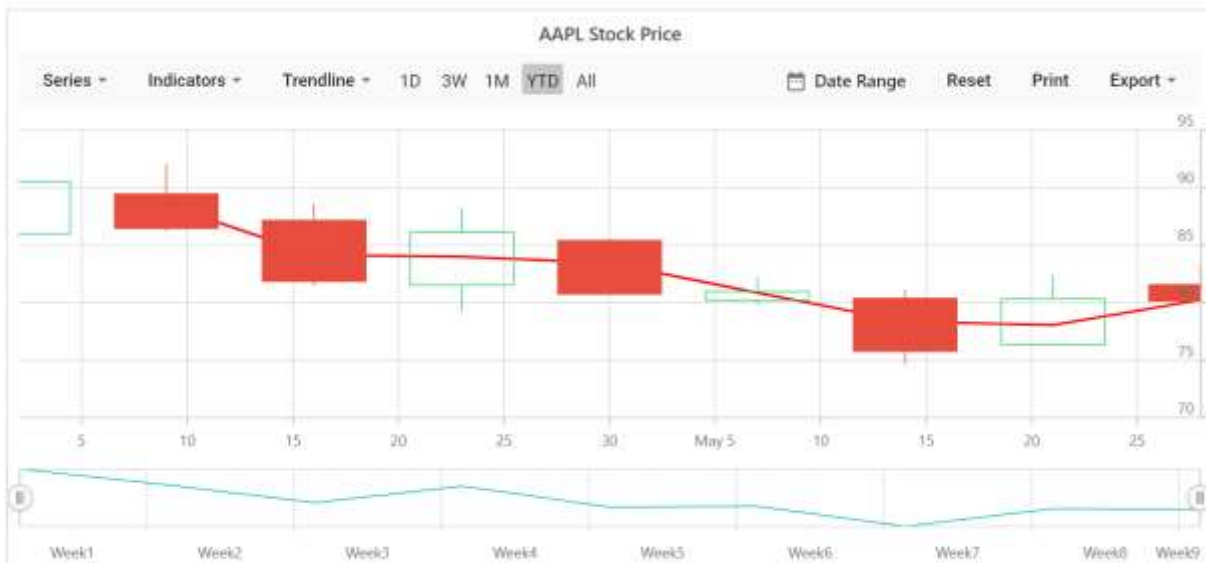
@using Syncfusion.Blazor.Charts
<SfStockChart Title="AAPL Stock Price">
  <StockChartSeriesCollection>
    <StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Candle"
      XName="Date" Name="Google">
      <StockChartTrendlines>
        <StockChartTrendline Type="TrendlineTypes.MovingAverage"
          EnableTooltip="false" Fill="red" Width="2"></StockChartTrendline>
      </StockChartTrendlines>
    </StockChartSeries>
  </StockChartSeriesCollection>
</SfStockChart>
@code {
  public class ChartData

```

```

{
    public DateTime Date;
    public Double Open;
    public Double Low;
    public Double Close;
    public Double High;
    public Double Volume;
}
public List<ChartData> StockDetails = new List<ChartData>
{
    new ChartData { Date = new DateTime(2012, 04, 02), Open= 85.9757, High = 90.6657,Low = 85.7685, Close = 90.5257,Volume = 660187068},
    new ChartData { Date = new DateTime(2012, 04, 09), Open= 89.4471, High = 92,Low = 86.2157, Close = 86.4614,Volume = 912634864},
    new ChartData { Date = new DateTime(2012, 04, 16), Open= 87.1514, High = 88.6071,Low = 81.4885, Close = 81.8543,Volume = 1221746066},
    new ChartData { Date = new DateTime(2012, 04, 23), Open= 81.5157, High = 88.2857,Low = 79.2857, Close = 86.1428,Volume = 965935749},
    new ChartData { Date = new DateTime(2012, 04, 30), Open= 85.4, High = 85.4857,Low = 80.7385, Close = 80.75,Volume = 615249365},
    new ChartData { Date = new DateTime(2012, 05, 07), Open= 80.2143, High = 82.2685,Low = 79.8185, Close = 80.9585,Volume = 541742692},
    new ChartData { Date = new DateTime(2012, 05, 14), Open= 80.3671, High = 81.0728,Low = 74.5971, Close = 75.7685,Volume = 708126233},
    new ChartData { Date = new DateTime(2012, 05, 21), Open= 76.3571, High = 82.3571,Low = 76.2928, Close = 80.3271,Volume = 682076215},
    new ChartData { Date = new DateTime(2012, 05, 28), Open= 81.5571, High = 83.0714,Low = 80.0743, Close = 80.1414,Volume = 480059584}
};
}

```



<!-- markdownlint-disable MD036 -->

Technical Indicators in Blazor Stock Chart Component

A technical indicator is a mathematical calculation based on historic price, volume or open interest information that aims to forecast financial market direction.

Stock Chart supports 10 types of technical indicators namely Accumulation Distribution, ATR, EMA, SMA, TMA, Momentum, MACD, RSI, Stochastic, Bollinger Band. By using indicator dropdown box, add and remove the required indicators types.

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`.

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`.

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`.

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 rendered at the value 100.

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.

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.

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`.

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 [OverSold](#) properties of indicators and the periodLine and signalLine is render based on stochastic formula.

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 .

Bollinger Band

<!-- markdownlint-disable MD034 -->

A Stock 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) and [StandardDeviations](#) is 2.

Tooltip in Blazor Stock Chart Component

<!-- markdownlint-disable MD036 -->

Stock Chart will display details about the points through tooltip, when the mouse is moved over the point.

Default Tooltip

By default, tooltip is not visible. Enable the tooltip by setting [Enable](#) property to true .

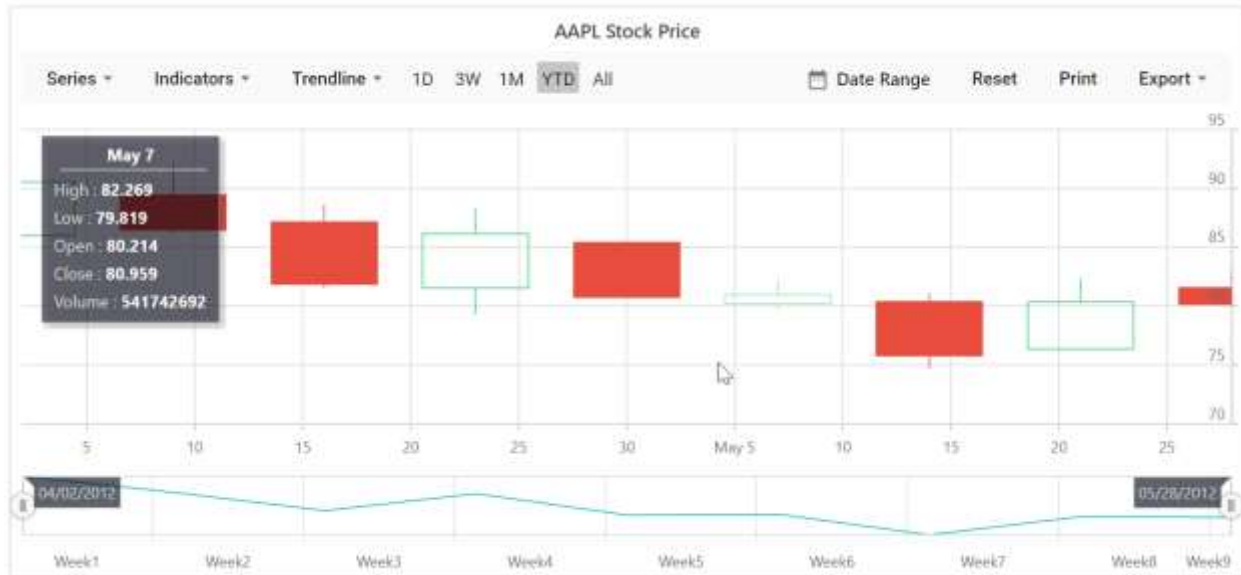
ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfStockChart Title="AAPL Stock Price">
  <StockChartTooltipSettings Enable="true">
  </StockChartTooltipSettings>
  <StockChartSeriesCollection>
  <StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Candle"
  XName="Date" High="High" Low="Low" Open="Open" Close="Close" Volume="Volume"
  Name="Google"></StockChartSeries>
  </StockChartSeriesCollection>
</SfStockChart>

@code {
public class ChartData
{
public DateTime Date;
public Double Open;
public Double Low;
public Double Close;
public Double High;
public Double Volume;
}

public List<ChartData> StockDetails = new List<ChartData>
{
new ChartData { Date = new DateTime(2012, 04, 02), Open= 85.9757, High =
90.6657,Low = 85.7685, Close = 90.5257,Volume = 660187068},
new ChartData { Date = new DateTime(2012, 04, 09), Open= 89.4471, High =
92,Low = 86.2157, Close = 86.4614,Volume = 912634864},
new ChartData { Date = new DateTime(2012, 04, 16), Open= 87.1514, High =
88.6071,Low = 81.4885, Close = 81.8543,Volume = 1221746066},
new ChartData { Date = new DateTime(2012, 04, 23), Open= 81.5157, High =
88.2857,Low = 79.2857, Close = 86.1428,Volume = 965935749},
new ChartData { Date = new DateTime(2012, 04, 30), Open= 85.4, High =
85.4857,Low = 80.7385, Close = 80.75,Volume = 615249365},
```

```
new ChartData { Date = new DateTime(2012, 05, 07), Open= 80.2143, High = 82.2685,Low = 79.8185, Close = 80.9585,Volume = 541742692},
new ChartData { Date = new DateTime(2012, 05, 14), Open= 80.3671, High = 81.0728,Low = 74.5971, Close = 75.7685,Volume = 708126233},
new ChartData { Date = new DateTime(2012, 05, 21), Open= 76.3571, High = 82.3571,Low = 76.2928, Close = 80.3271,Volume = 682076215},
new ChartData { Date = new DateTime(2012, 05, 28), Open= 81.5571, High = 83.0714,Low = 80.0743, Close = 80.1414,Volume = 480059584}
};
}
```



```
<!-- markdownlint-disable MD013 -->
```

Format the Tooltip

```
<!-- markdownlint-disable MD013 -->
```

By default, tooltip shows information of x and y value in points. In addition to that, more information can be shown in tooltip. For example the format `${point.x} : ${point.high}` shows point x and high value.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfStockChart Title="AAPL Stock Price">
  <StockChartTooltipSettings Enable="true" Format="<b>${point.x} : 
  ${point.high}</b>">
  </StockChartTooltipSettings>
  <StockChartSeriesCollection>
  <StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Candle"
  XName="Date" High="High" Low="Low" Open="Open" Close="Close" Volume="Volume"
  Name="google"></StockChartSeries>
  </StockChartSeriesCollection>
</SfStockChart>
@code {
public class ChartData
{
```

```

public DateTime Date;
public Double Open;
public Double Low;
public Double Close;
public Double High;
public Double Volume;
}
public List<ChartData> StockDetails = new List<ChartData>
{
    new ChartData { Date = new DateTime(2012, 04, 02), Open= 85.9757, High =
    90.6657,Low = 85.7685, Close = 90.5257,Volume = 660187068},
    new ChartData { Date = new DateTime(2012, 04, 09), Open= 89.4471, High =
    92,Low = 86.2157, Close = 86.4614,Volume = 912634864},
    new ChartData { Date = new DateTime(2012, 04, 16), Open= 87.1514, High =
    88.6071,Low = 81.4885, Close = 81.8543,Volume = 1221746066},
    new ChartData { Date = new DateTime(2012, 04, 23), Open= 81.5157, High =
    88.2857,Low = 79.2857, Close = 86.1428,Volume = 965935749},
    new ChartData { Date = new DateTime(2012, 04, 30), Open= 85.4, High =
    85.4857,Low = 80.7385, Close = 80.75,Volume = 615249365},
    new ChartData { Date = new DateTime(2012, 05, 07), Open= 80.2143, High =
    82.2685,Low = 79.8185, Close = 80.9585,Volume = 541742692},
    new ChartData { Date = new DateTime(2012, 05, 14), Open= 80.3671, High =
    81.0728,Low = 74.5971, Close = 75.7685,Volume = 708126233},
    new ChartData { Date = new DateTime(2012, 05, 21), Open= 76.3571, High =
    82.3571,Low = 76.2928, Close = 80.3271,Volume = 682076215},
    new ChartData { Date = new DateTime(2012, 05, 28), Open= 81.5571, High =
    83.0714,Low = 80.0743, Close = 80.1414,Volume = 480059584}
};
}

```

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.

ASPX-CS

```

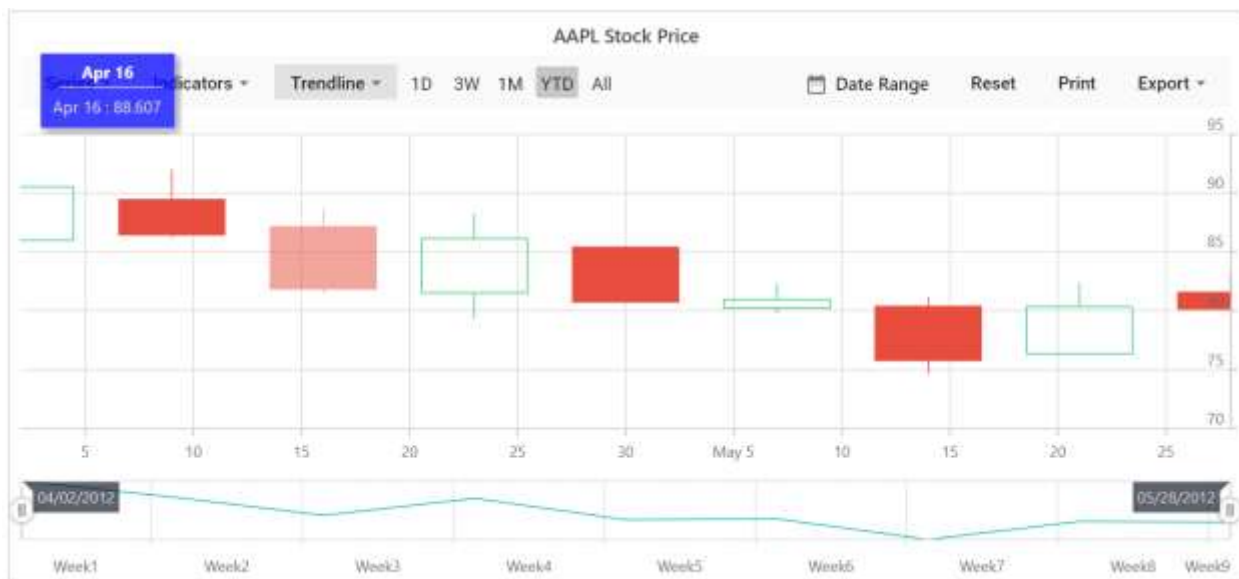
@using Syncfusion.Blazor.Charts
<SfStockChart Title="AAPL Stock Price">
<StockChartTooltipSettings Enable="true" Format="{point.x} : {point.high}"
Fill="#7bb4eb">
</StockChartTooltipSettings>
<StockChartSeriesCollection>
<StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Candle"
XName="Date" High="High" Low="Low" Open="Open" Close="Close" Volume="Volume"
Name="Google"></StockChartSeries>
</StockChartSeriesCollection>
</SfStockChart>
@code {
public class ChartData
{
    public DateTime Date;
    public Double Open;
    public Double Low;
    public Double Close;
    public Double High;
    public Double Volume;
}
}

```

```

}
public List<ChartData> StockDetails = new List<ChartData>
{
    new ChartData { Date = new DateTime(2012, 04, 02), Open= 85.9757, High = 90.6657,Low = 85.7685, Close = 90.5257,Volume = 660187068},
    new ChartData { Date = new DateTime(2012, 04, 09), Open= 89.4471, High = 92,Low = 86.2157, Close = 86.4614,Volume = 912634864},
    new ChartData { Date = new DateTime(2012, 04, 16), Open= 87.1514, High = 88.6071,Low = 81.4885, Close = 81.8543,Volume = 1221746066},
    new ChartData { Date = new DateTime(2012, 04, 23), Open= 81.5157, High = 88.2857,Low = 79.2857, Close = 86.1428,Volume = 965935749},
    new ChartData { Date = new DateTime(2012, 04, 30), Open= 85.4, High = 85.4857,Low = 80.7385, Close = 80.75,Volume = 615249365},
    new ChartData { Date = new DateTime(2012, 05, 07), Open= 80.2143, High = 82.2685,Low = 79.8185, Close = 80.9585,Volume = 541742692},
    new ChartData { Date = new DateTime(2012, 05, 14), Open= 80.3671, High = 81.0728,Low = 74.5971, Close = 75.7685,Volume = 708126233},
    new ChartData { Date = new DateTime(2012, 05, 21), Open= 76.3571, High = 82.3571,Low = 76.2928, Close = 80.3271,Volume = 682076215},
    new ChartData { Date = new DateTime(2012, 05, 28), Open= 81.5571, High = 83.0714,Low = 80.0743, Close = 80.1414,Volume = 480059584}
};
}

```



Crosshair in Blazor Stock Chart Component

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`.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfStockChart Title="AAPL Stock Price">
  <StockChartCrosshairSettings Enable="true"></StockChartCrosshairSettings>
  <StockChartSeriesCollection>

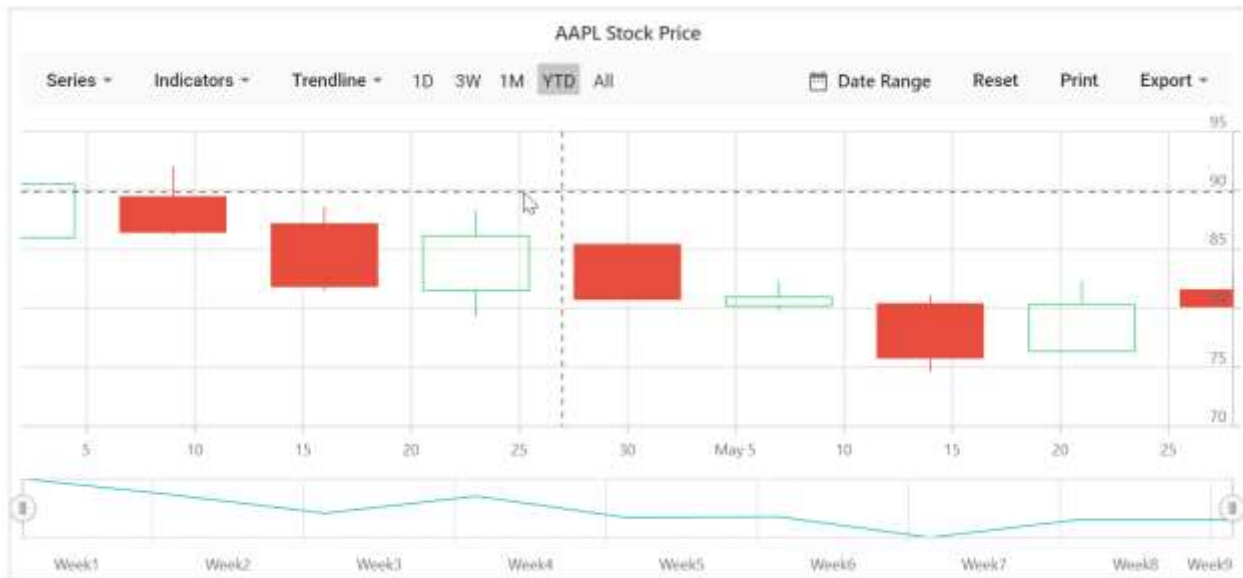
```



```

<StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Candle"
XName="Date" High="High" Low="Low" Open="Open" Close="Close"
Volume="Volume"></StockChartSeries>
</StockChartSeriesCollection>
</SfStockChart>
@code {
public class ChartData
{
public DateTime Date;
public Double Open;
public Double Low;
public Double Close;
public Double High;
public Double Volume;
}
public List<ChartData> StockDetails = new List<ChartData>
{
new ChartData { Date = new DateTime(2012, 04, 02), Open= 85.9757, High =
90.6657,Low = 85.7685, Close = 90.5257,Volume = 660187068},
new ChartData { Date = new DateTime(2012, 04, 09), Open= 89.4471, High =
92,Low = 86.2157, Close = 86.4614,Volume = 912634864},
new ChartData { Date = new DateTime(2012, 04, 16), Open= 87.1514, High =
88.6071,Low = 81.4885, Close = 81.8543,Volume = 1221746066},
new ChartData { Date = new DateTime(2012, 04, 23), Open= 81.5157, High =
88.2857,Low = 79.2857, Close = 86.1428,Volume = 965935749},
new ChartData { Date = new DateTime(2012, 04, 30), Open= 85.4, High =
85.4857,Low = 80.7385, Close = 80.75,Volume = 615249365},
new ChartData { Date = new DateTime(2012, 05, 07), Open= 80.2143, High =
82.2685,Low = 79.8185, Close = 80.9585,Volume = 541742692},
new ChartData { Date = new DateTime(2012, 05, 14), Open= 80.3671, High =
81.0728,Low = 74.5971, Close = 75.7685,Volume = 708126233},
new ChartData { Date = new DateTime(2012, 05, 21), Open= 76.3571, High =
82.3571,Low = 76.2928, Close = 80.3271,Volume = 682076215},
new ChartData { Date = new DateTime(2012, 05, 28), Open= 81.5571, High =
83.0714,Low = 80.0743, Close = 80.1414,Volume = 480059584}
};
}

```



Tooltip for axis

Tooltip label for an axis can be enabled by using [Enable](#) property of `CrosshairTooltip` in the corresponding axis.

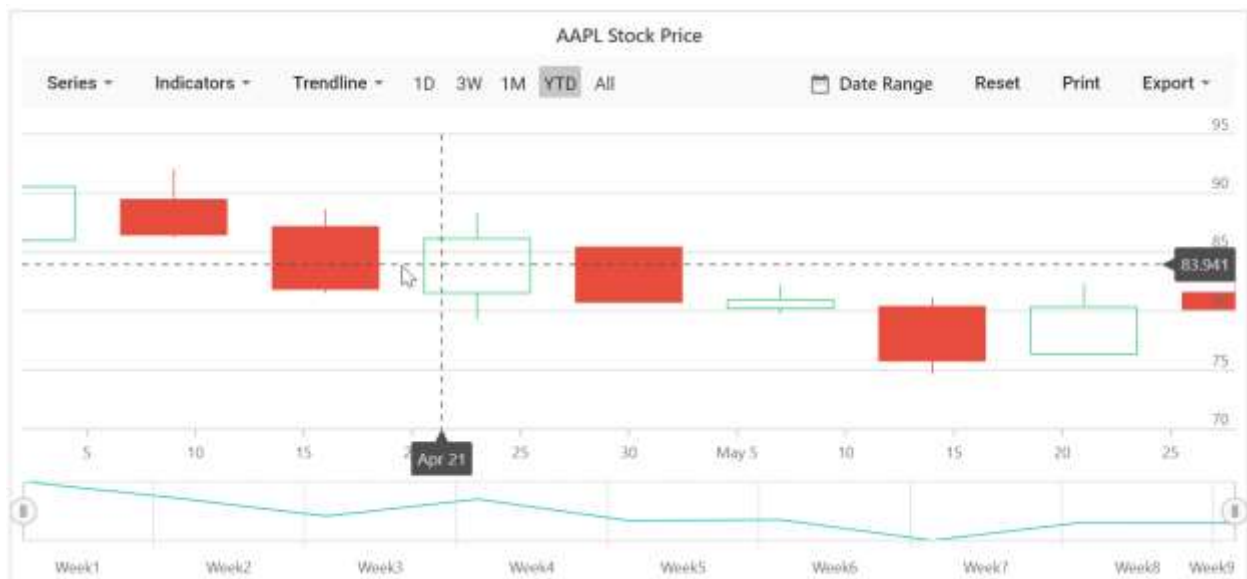
ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfStockChart Title="AAPL Stock Price">
  <StockChartCrosshairSettings Enable="true"></StockChartCrosshairSettings>
  <StockChartPrimaryYAxis>
    <StockChartAxisLineStyle Color="Transparent"></StockChartAxisLineStyle>
    <StockChartAxisMajorTickLines Color="Transparent"
      Width="0"></StockChartAxisMajorTickLines>
    <StockChartAxisCrosshairTooltip
      Enable="true"></StockChartAxisCrosshairTooltip>
  </StockChartPrimaryYAxis>
  <StockChartPrimaryXAxis>
    <StockChartAxisMajorGridLines
      Color="Transparent"></StockChartAxisMajorGridLines>
    <StockChartAxisCrosshairTooltip
      Enable="true"></StockChartAxisCrosshairTooltip>
  </StockChartPrimaryXAxis>
  <StockChartSeriesCollection>
    <StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Candle"
      XName="Date" High="High" Low="Low" Open="Open" Close="Close"
      Volume="Volume"></StockChartSeries>
  </StockChartSeriesCollection>
</SfStockChart>
@code {
  public class ChartData
  {
    public DateTime Date;
    public Double Open;
    public Double Low;
    public Double Close;
    public Double High;
    public Double Volume;
  }
}
```

```

}
public List<ChartData> StockDetails = new List<ChartData>
{
    new ChartData { Date = new DateTime(2012, 04, 02), Open= 85.9757, High = 90.6657,Low = 85.7685, Close = 90.5257,Volume = 660187068},
    new ChartData { Date = new DateTime(2012, 04, 09), Open= 89.4471, High = 92,Low = 86.2157, Close = 86.4614,Volume = 912634864},
    new ChartData { Date = new DateTime(2012, 04, 16), Open= 87.1514, High = 88.6071,Low = 81.4885, Close = 81.8543,Volume = 1221746066},
    new ChartData { Date = new DateTime(2012, 04, 23), Open= 81.5157, High = 88.2857,Low = 79.2857, Close = 86.1428,Volume = 965935749},
    new ChartData { Date = new DateTime(2012, 04, 30), Open= 85.4, High = 85.4857,Low = 80.7385, Close = 80.75,Volume = 615249365},
    new ChartData { Date = new DateTime(2012, 05, 07), Open= 80.2143, High = 82.2685,Low = 79.8185, Close = 80.9585,Volume = 541742692},
    new ChartData { Date = new DateTime(2012, 05, 14), Open= 80.3671, High = 81.0728,Low = 74.5971, Close = 75.7685,Volume = 708126233},
    new ChartData { Date = new DateTime(2012, 05, 21), Open= 76.3571, High = 82.3571,Low = 76.2928, Close = 80.3271,Volume = 682076215},
    new ChartData { Date = new DateTime(2012, 05, 28), Open= 81.5571, High = 83.0714,Low = 80.0743, Close = 80.1414,Volume = 480059584}
};
}

```



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.

ASPX-CS

```

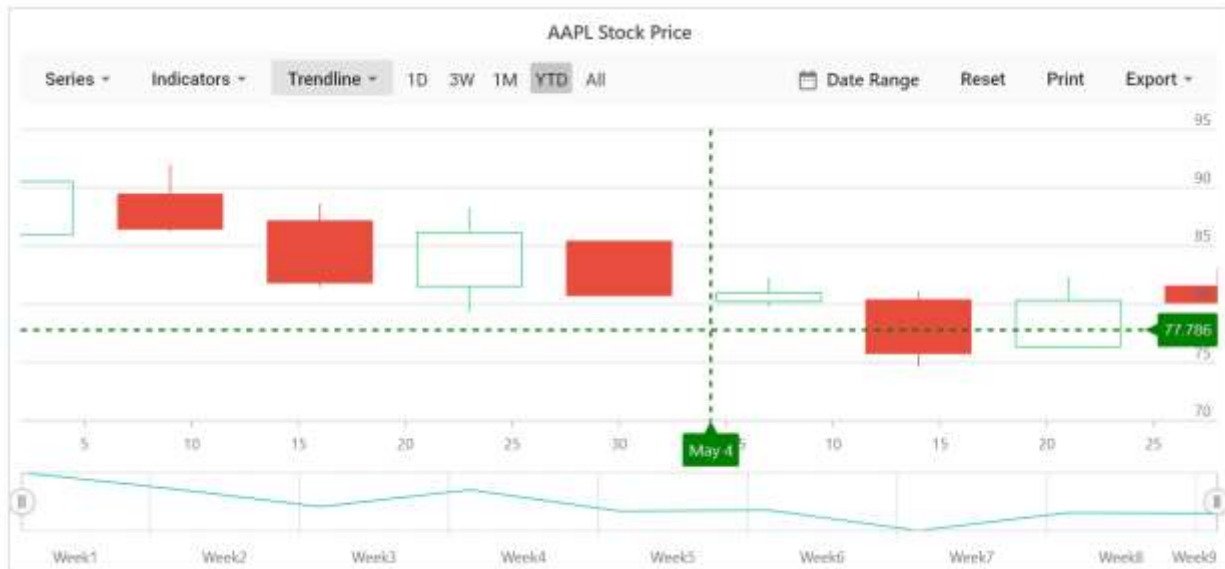
@using Syncfusion.Blazor.Charts
<SfStockChart Title="AAPL Stock Price">
  <StockChartCrosshairSettings Enable="true">
    <StockChartCrosshairLine Width="2" Color="green"></StockChartCrosshairLine>
  </StockChartCrosshairSettings>

```

```

<StockChartSeriesCollection>
  <StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Candle"
  XName="Date" High="High" Low="Low" Open="Open" Close="Close"
  Volume="Volume"></StockChartSeries>
</StockChartSeriesCollection>
<StockChartPrimaryYAxis>
  <StockChartAxisLineStyle Color="Transparent"></StockChartAxisLineStyle>
  <StockChartAxisMajorTickLines Color="Transparent"
  Width="0"></StockChartAxisMajorTickLines>
  <StockChartAxisCrosshairTooltip Enable="true"
  Fill="green"></StockChartAxisCrosshairTooltip>
</StockChartPrimaryYAxis>
<StockChartPrimaryXAxis>
  <StockChartAxisMajorGridLines
  Color="Transparent"></StockChartAxisMajorGridLines>
  <StockChartAxisCrosshairTooltip Enable="true"
  Fill="green"></StockChartAxisCrosshairTooltip>
</StockChartPrimaryXAxis>
</SfStockChart>
@code {
public class ChartData
{
public DateTime Date;
public Double Open;
public Double Low;
public Double Close;
public Double High;
public Double Volume;
}
public List<ChartData> StockDetails = new List<ChartData>
{
new ChartData { Date = new DateTime(2012, 04, 02), Open= 85.9757, High =
90.6657,Low = 85.7685, Close = 90.5257,Volume = 660187068},
new ChartData { Date = new DateTime(2012, 04, 09), Open= 89.4471, High =
92,Low = 86.2157, Close = 86.4614,Volume = 912634864},
new ChartData { Date = new DateTime(2012, 04, 16), Open= 87.1514, High =
88.6071,Low = 81.4885, Close = 81.8543,Volume = 1221746066},
new ChartData { Date = new DateTime(2012, 04, 23), Open= 81.5157, High =
88.2857,Low = 79.2857, Close = 86.1428,Volume = 965935749},
new ChartData { Date = new DateTime(2012, 04, 30), Open= 85.4, High =
85.4857,Low = 80.7385, Close = 80.75,Volume = 615249365},
new ChartData { Date = new DateTime(2012, 05, 07), Open= 80.2143, High =
82.2685,Low = 79.8185, Close = 80.9585,Volume = 541742692},
new ChartData { Date = new DateTime(2012, 05, 14), Open= 80.3671, High =
81.0728,Low = 74.5971, Close = 75.7685,Volume = 708126233},
new ChartData { Date = new DateTime(2012, 05, 21), Open= 76.3571, High =
82.3571,Low = 76.2928, Close = 80.3271,Volume = 682076215},
new ChartData { Date = new DateTime(2012, 05, 28), Open= 81.5571, High =
83.0714,Low = 80.0743, Close = 80.1414,Volume = 480059584}
};
}

```



Add 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.

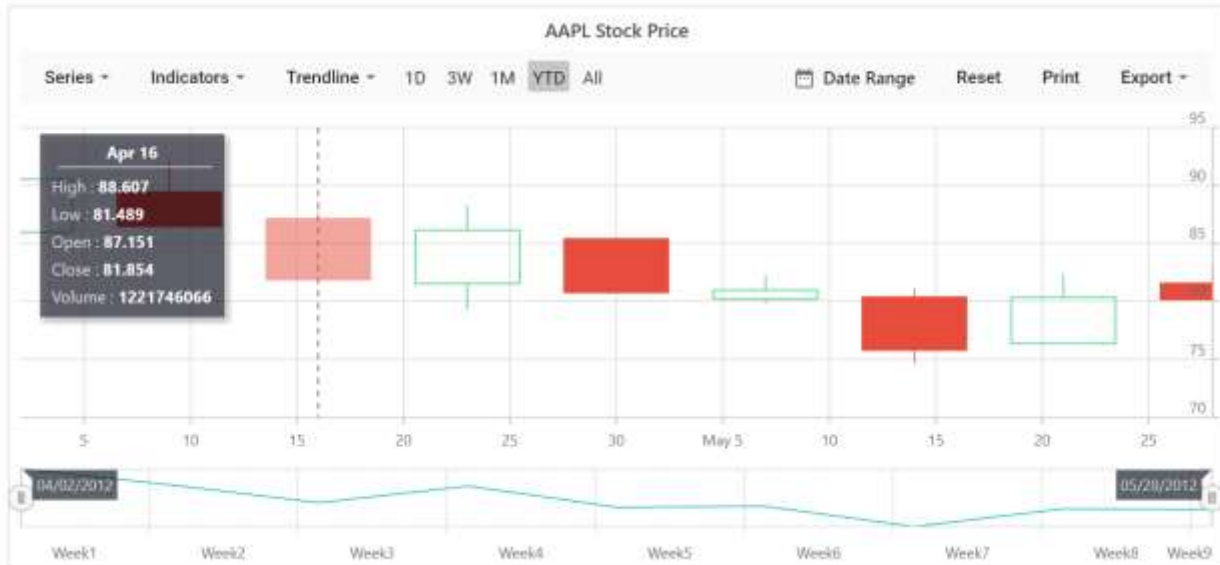
ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfStockChart Title="AAPL Stock Price">
  <StockChartCrosshairSettings Enable="true"
  LineType="LineType.Vertical"></StockChartCrosshairSettings>
  <StockChartTooltipSettings Enable="true"
  Shared="true"></StockChartTooltipSettings>
  <StockChartSeriesCollection>
    <StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Candle"
    XName="Date" High="High" Low="Low" Open="Open" Close="Close"
    Volume="Volume"></StockChartSeries>
  </StockChartSeriesCollection>
</SfStockChart>
@code {
  public class ChartData
  {
    public DateTime Date;
    public Double Open;
    public Double Low;
    public Double Close;
    public Double High;
    public Double Volume;
  }
  public List<ChartData> StockDetails = new List<ChartData>
  {
    new ChartData { Date = new DateTime(2012, 04, 02), Open= 85.9757, High =
    90.6657,Low = 85.7685, Close = 90.5257,Volume = 660187068},
```

```

new ChartData { Date = new DateTime(2012, 04, 09), Open= 89.4471, High =
92,Low = 86.2157, Close = 86.4614,Volume = 912634864},
new ChartData { Date = new DateTime(2012, 04, 16), Open= 87.1514, High =
88.6071,Low = 81.4885, Close = 81.8543,Volume = 1221746066},
new ChartData { Date = new DateTime(2012, 04, 23), Open= 81.5157, High =
88.2857,Low = 79.2857, Close = 86.1428,Volume = 965935749},
new ChartData { Date = new DateTime(2012, 04, 30), Open= 85.4, High =
85.4857,Low = 80.7385, Close = 80.75,Volume = 615249365},
new ChartData { Date = new DateTime(2012, 05, 07), Open= 80.2143, High =
82.2685,Low = 79.8185, Close = 80.9585,Volume = 541742692},
new ChartData { Date = new DateTime(2012, 05, 14), Open= 80.3671, High =
81.0728,Low = 74.5971, Close = 75.7685,Volume = 708126233},
new ChartData { Date = new DateTime(2012, 05, 21), Open= 76.3571, High =
82.3571,Low = 76.2928, Close = 80.3271,Volume = 682076215},
new ChartData { Date = new DateTime(2012, 05, 28), Open= 81.5571, High =
83.0714,Low = 80.0743, Close = 80.1414,Volume = 480059584}
};
}

```



<!-- markdownlint-disable MD036 -->

Period Selector in Blazor Stock Chart Component

The period selector allows to select a range with specified periods. By default the period selector is enabled in stock chart.

Periods

<!-- markdownlint-disable MD034 -->

Periods is an array of objects that allows users to specify the range of [Periods](#). The `Interval` property specifies the count value of the button, and the `Text` property specifies the text to be displayed on button. The `IntervalType` property allows users to customize the intervals of the buttons. The `IntervalType` property supports the following interval types:

- Auto

- Years
- Quarter
- Months
- Weeks
- Days
- Hours
- Minutes
- Seconds

ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfStockChart Title="AAPL Stock Price" IndicatorType="@IndicatorType"
TrendlineType="@TrendlineType" SeriesType="@SeriesType"
ExportType="@ExportType">
  <StockChartPeriods>
    <StockChartPeriod IntervalType=RangeIntervalType.Minutes Interval="1"
Text='1m'></StockChartPeriod>
    <StockChartPeriod IntervalType=RangeIntervalType.Minutes Interval="30"
Text='30m'></StockChartPeriod>
    <StockChartPeriod IntervalType=RangeIntervalType.Hours Interval="1"
Text='1h'></StockChartPeriod>
    <StockChartPeriod IntervalType=RangeIntervalType.Hours Interval="12"
Text='12h' Selected="true"></StockChartPeriod>
    <StockChartPeriod Text="1D"></StockChartPeriod>
  </StockChartPeriods>
  <StockChartCrosshairSettings Enable="true"></StockChartCrosshairSettings>
  <StockChartPrimaryXAxis>
    <StockChartAxisMajorGridLines
Color="Transparent"></StockChartAxisMajorGridLines>
  </StockChartPrimaryXAxis>
  <StockChartPrimaryYAxis>
    <StockChartAxisLineStyle Color="Transparent"></StockChartAxisLineStyle>
    <StockChartAxisMajorTickLines Color="Transparent"
Width="0"></StockChartAxisMajorTickLines>
  </StockChartPrimaryYAxis>
  <StockChartSeriesCollection>
    <StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Candle"
XName="Date" Name="google"></StockChartSeries>
  </StockChartSeriesCollection>
</SfStockChart>

@code {
public List<TechnicalIndicators> IndicatorType = new
List<TechnicalIndicators>() { };
public List<TrendlineTypes> TrendlineType = new List<TrendlineTypes>() { };
public List<ExportType> ExportType = new List<ExportType>() { };
public List<ChartSeriesType> SeriesType = new List<ChartSeriesType>() { };
public class ChartData
{
public DateTime Date;
public Double Open;
public Double Low;
public Double Close;
public Double High;
public Double Volume;
}
}

```

```

}
public List<ChartData> StockDetails = new List<ChartData>
{
    new ChartData { Date = new DateTime(2012, 04, 02), Open= 85.9757, High = 90.6657,Low = 85.7685, Close = 90.5257,Volume = 660187068},
    new ChartData { Date = new DateTime(2012, 04, 09), Open= 89.4471, High = 92,Low = 86.2157, Close = 86.4614,Volume = 912634864},
    new ChartData { Date = new DateTime(2012, 04, 16), Open= 87.1514, High = 88.6071,Low = 81.4885, Close = 81.8543,Volume = 1221746066},
    new ChartData { Date = new DateTime(2012, 04, 23), Open= 81.5157, High = 88.2857,Low = 79.2857, Close = 86.1428,Volume = 965935749},
    new ChartData { Date = new DateTime(2012, 04, 30), Open= 85.4, High = 85.4857,Low = 80.7385, Close = 80.75,Volume = 615249365},
    new ChartData { Date = new DateTime(2012, 05, 07), Open= 80.2143, High = 82.2685,Low = 79.8185, Close = 80.9585,Volume = 541742692},
    new ChartData { Date = new DateTime(2012, 05, 14), Open= 80.3671, High = 81.0728,Low = 74.5971, Close = 75.7685,Volume = 708126233},
    new ChartData { Date = new DateTime(2012, 05, 21), Open= 76.3571, High = 82.3571,Low = 76.2928, Close = 80.3271,Volume = 682076215},
    new ChartData { Date = new DateTime(2012, 05, 28), Open= 81.5571, High = 83.0714,Low = 80.0743, Close = 80.1414,Volume = 480059584}
};
}

```



Visibility of period selector

The [EnablePeriodSelector](#) property allows users to toggle the visibility of period selector.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfStockChart Title="AAPL Stock Price" EnablePeriodSelector="false">
  <StockChartSeriesCollection>
    <StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Candle"
      XName="Date" High="High" Low="Low" Open="Open" Close="Close" Volume="Volume"
      Name="google"></StockChartSeries>
  </StockChartSeriesCollection>

```



```

</SfStockChart>
@code {
public class ChartData
{
public DateTime Date;
public Double Open;
public Double Low;
public Double Close;
public Double High;
public Double Volume;
}
public List<ChartData> StockDetails = new List<ChartData>
{
new ChartData { Date = new DateTime(2012, 04, 02), Open= 85.9757, High =
90.6657,Low = 85.7685, Close = 90.5257,Volume = 660187068},
new ChartData { Date = new DateTime(2012, 04, 09), Open= 89.4471, High =
92,Low = 86.2157, Close = 86.4614,Volume = 912634864},
new ChartData { Date = new DateTime(2012, 04, 16), Open= 87.1514, High =
88.6071,Low = 81.4885, Close = 81.8543,Volume = 1221746066},
new ChartData { Date = new DateTime(2012, 04, 23), Open= 81.5157, High =
88.2857,Low = 79.2857, Close = 86.1428,Volume = 965935749},
new ChartData { Date = new DateTime(2012, 04, 30), Open= 85.4, High =
85.4857,Low = 80.7385, Close = 80.75,Volume = 615249365},
new ChartData { Date = new DateTime(2012, 05, 07), Open= 80.2143, High =
82.2685,Low = 79.8185, Close = 80.9585,Volume = 541742692},
new ChartData { Date = new DateTime(2012, 05, 14), Open= 80.3671, High =
81.0728,Low = 74.5971, Close = 75.7685,Volume = 708126233},
new ChartData { Date = new DateTime(2012, 05, 21), Open= 76.3571, High =
82.3571,Low = 76.2928, Close = 80.3271,Volume = 682076215},
new ChartData { Date = new DateTime(2012, 05, 28), Open= 81.5571, High =
83.0714,Low = 80.0743, Close = 80.1414,Volume = 480059584}
};
}

```



<!-- markdownlint-disable MD036 -->

Range Selector in Blazor Stock Chart Component

The range selector allows to select a range with specified periods. By default, the range selector is enabled in stock chart.

Selecting Range

The left and right thumb of RangeNavigator are used to indicate the selected range in the large collection of data. Following are the ways to select a range.

- By dragging the thumbs.
- By tapping on the labels.
- By setting the start and end through Date Range button.

Following code example shows the [EnableSelector](#) property that allows users to toggle the visibility of enable selector.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfStockChart Title="AAPL Stock Price" EnableSelector="false">
<StockChartSeriesCollection>
<StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Candle"
XName="Date" Name="google"></StockChartSeries>
</StockChartSeriesCollection>
</SfStockChart>
@code {
public class ChartData
{
public DateTime Date;
public Double Open;
public Double Low;
public Double Close;
public Double High;
public Double Volume;
}
public List<ChartData> StockDetails = new List<ChartData>
{
new ChartData { Date = new DateTime(2012, 04, 02), Open= 85.9757, High =
90.6657,Low = 85.7685, Close = 90.5257,Volume = 660187068},
new ChartData { Date = new DateTime(2012, 04, 09), Open= 89.4471, High =
92,Low = 86.2157, Close = 86.4614,Volume = 912634864},
new ChartData { Date = new DateTime(2012, 04, 16), Open= 87.1514, High =
88.6071,Low = 81.4885, Close = 81.8543,Volume = 1221746066},
new ChartData { Date = new DateTime(2012, 04, 23), Open= 81.5157, High =
88.2857,Low = 79.2857, Close = 86.1428,Volume = 965935749},
new ChartData { Date = new DateTime(2012, 04, 30), Open= 85.4, High =
85.4857,Low = 80.7385, Close = 80.75,Volume = 615249365},
new ChartData { Date = new DateTime(2012, 05, 07), Open= 80.2143, High =
82.2685,Low = 79.8185, Close = 80.9585,Volume = 541742692},
new ChartData { Date = new DateTime(2012, 05, 14), Open= 80.3671, High =
81.0728,Low = 74.5971, Close = 75.7685,Volume = 708126233},
new ChartData { Date = new DateTime(2012, 05, 21), Open= 76.3571, High =
82.3571,Low = 76.2928, Close = 80.3271,Volume = 682076215},
new ChartData { Date = new DateTime(2012, 05, 28), Open= 81.5571, High =
83.0714,Low = 80.0743, Close = 80.1414,Volume = 480059584}
};
};
```

```
}

```



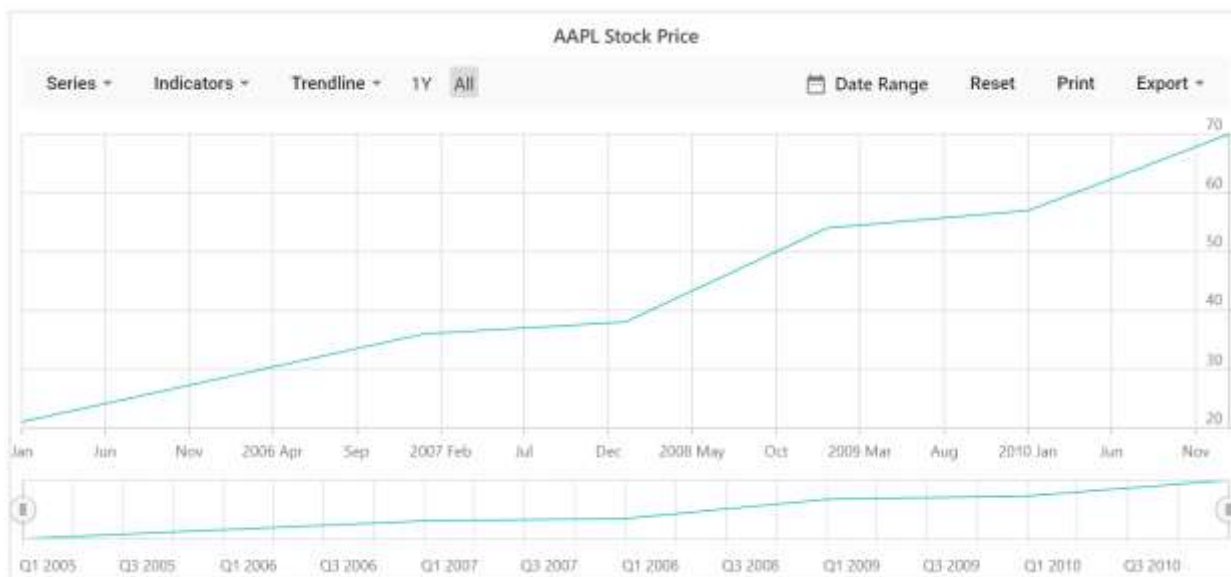
Appearance in Blazor Stock Chart Component

Stock Chart Title

Stock Chart can be given a title using [Title](#) property, to show the information about the data plotted.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfStockChart Title="AAPL Stock Price">
  <StockChartSeriesCollection>
    <StockChartSeries DataSource="@DataSource" Type="ChartSeriesType.Line"
      XName="XValue" YName="YValue"></StockChartSeries>
  </StockChartSeriesCollection>
  <StockChartPeriods>
    <StockChartPeriod Text="1Y"></StockChartPeriod>
    <StockChartPeriod Text="All" Selected="true"></StockChartPeriod>
  </StockChartPeriods>
</SfStockChart>
@code {
  public class ChartData
  {
    public DateTime XValue;
    public double YValue;
  }
  public List<ChartData> DataSource = new List<ChartData>
  {
    new ChartData { XValue = new DateTime(2005, 01, 01), YValue = 21 },
    new ChartData { XValue = new DateTime(2006, 01, 01), YValue = 24 },
    new ChartData { XValue = new DateTime(2007, 01, 01), YValue = 36 },
    new ChartData { XValue = new DateTime(2008, 01, 01), YValue = 38 },
    new ChartData { XValue = new DateTime(2009, 01, 01), YValue = 54 },
    new ChartData { XValue = new DateTime(2010, 01, 01), YValue = 57 },
    new ChartData { XValue = new DateTime(2011, 01, 01), YValue = 70 },
  };
}
```



<!-- markdownlint-disable MD036 -->

Title Customizations

The `TextStyle` property of chart title provides options to customize the `Size`, `Color`, `FontFamily`, `FontWeight`, `FontStyle`, `Opacity`, `TextAlignment` and `TextOverflow`.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfStockChart Title="AAPL Stock Price">
  <StockChartTitleStyle FontFamily="Arial" FontStyle="Italic"
    FontWeight="Regular" Color="#E27F2D" Size="20px"
    TextOverflow="TextOverflow.Wrap"></StockChartTitleStyle>
  <StockChartSeriesCollection>
    <StockChartSeries DataSource="@DataSource" Type="ChartSeriesType.Line"
      XName="XValue" YName="YValue"></StockChartSeries>
  </StockChartSeriesCollection>
  <StockChartPeriods>
    <StockChartPeriod Text="1Y"></StockChartPeriod>
    <StockChartPeriod Text="All" Selected="true"></StockChartPeriod>
  </StockChartPeriods>
</SfStockChart>

@code {
  public class ChartData
  {
    public DateTime XValue;
    public double YValue;
  }
  public List<ChartData> DataSource = new List<ChartData>
  {
    new ChartData { XValue = new DateTime(2005, 01, 01), YValue = 21 },
    new ChartData { XValue = new DateTime(2006, 01, 01), YValue = 24 },
    new ChartData { XValue = new DateTime(2007, 01, 01), YValue = 36 },
    new ChartData { XValue = new DateTime(2008, 01, 01), YValue = 38 },
    new ChartData { XValue = new DateTime(2009, 01, 01), YValue = 54 },
  }
}
```

```
new ChartData { XValue = new DateTime(2010, 01, 01), YValue = 57 },
new ChartData { XValue = new DateTime(2011, 01, 01), YValue = 70 },
};
}
```



Stock Chart Theme

Changing theme will affect background color, gridlines, tooltip colors and appearance.

Stock chart is shipped with several built-in themes such as **Material**, **Fabric**, **Bootstrap**, **HighContrastLight**, **MaterialDark**, **FabricDark**, **FabricDark**, **HighContrast** and **BootstrapDark**.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfStockChart Title="AAPL Stock Price" Theme="ChartTheme.HighContrast">
  <StockChartSeriesCollection>
    <StockChartSeries DataSource="@DataSource" Type="ChartSeriesType.Line"
      XName="XValue" YName="YValue"></StockChartSeries>
  </StockChartSeriesCollection>
  <StockChartPeriods>
    <StockChartPeriod Text="1Y"></StockChartPeriod>
    <StockChartPeriod Text="All" Selected="true"></StockChartPeriod>
  </StockChartPeriods>
</SfStockChart>
@code {
  public class ChartData
  {
    public DateTime XValue;
    public double YValue;
  }
  public List<ChartData> DataSource = new List<ChartData>
  {
    new ChartData { XValue = new DateTime(2005, 01, 01), YValue = 21 },
    new ChartData { XValue = new DateTime(2006, 01, 01), YValue = 24 },
    new ChartData { XValue = new DateTime(2007, 01, 01), YValue = 36 },
    new ChartData { XValue = new DateTime(2008, 01, 01), YValue = 38 },
  }
}
```

```
new ChartData { XValue = new DateTime(2009, 01, 01), YValue = 54 },
new ChartData { XValue = new DateTime(2010, 01, 01), YValue = 57 },
new ChartData { XValue = new DateTime(2011, 01, 01), YValue = 70 },
};
}
```



See Also

- [Axis Customization](#)

Print and Export in Blazor Stock Chart Component

The rendered stock chart can be exported to JPEG, PNG, SVG, or PDF format using the export dropdown button in the period selector toolbar. The required format can be chosen using the export dropdown button in stock-chart.

The rendered stock chart can be printed directly using print button in period selector toolbar.

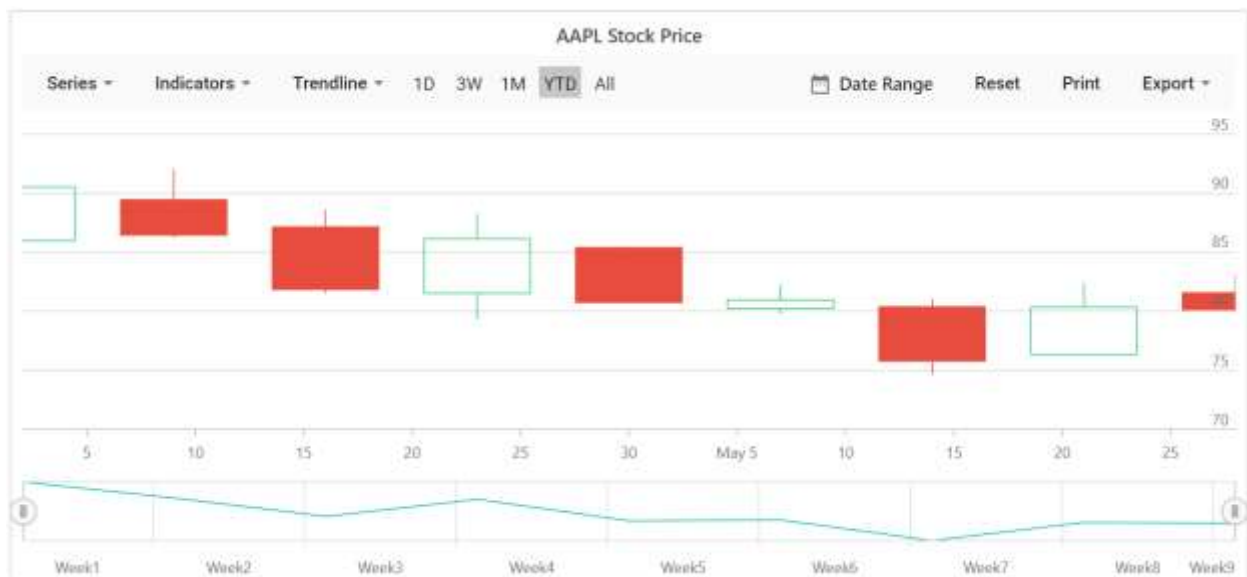
ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfStockChart Title="AAPL Stock Price">
  <StockChartSeriesCollection>
    <StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Candle"
      XName="Date" High="High" Low="Low" Open="Open" Close="Close"
      Volume="Volume"></StockChartSeries>
  </StockChartSeriesCollection>
  <StockChartPrimaryYAxis>
    <StockChartAxisLineStyle Color="Transparent"></StockChartAxisLineStyle>
    <StockChartAxisMajorTickLines Color="Transparent"
      Width="0"></StockChartAxisMajorTickLines>
  </StockChartPrimaryYAxis>
  <StockChartPrimaryXAxis>
    <StockChartAxisMajorGridLines
      Color="Transparent"></StockChartAxisMajorGridLines>
  </StockChartPrimaryXAxis>
```

```

</SfStockChart>
@code {
public class ChartData
{
public DateTime Date;
public Double Open;
public Double Low;
public Double Close;
public Double High;
public Double Volume;
}
public List<ChartData> StockDetails = new List<ChartData>
{
new ChartData { Date = new DateTime(2012, 04, 02), Open= 85.9757, High =
90.6657,Low = 85.7685, Close = 90.5257,Volume = 660187068},
new ChartData { Date = new DateTime(2012, 04, 09), Open= 89.4471, High =
92,Low = 86.2157, Close = 86.4614,Volume = 912634864},
new ChartData { Date = new DateTime(2012, 04, 16), Open= 87.1514, High =
88.6071,Low = 81.4885, Close = 81.8543,Volume = 1221746066},
new ChartData { Date = new DateTime(2012, 04, 23), Open= 81.5157, High =
88.2857,Low = 79.2857, Close = 86.1428,Volume = 965935749},
new ChartData { Date = new DateTime(2012, 04, 30), Open= 85.4, High =
85.4857,Low = 80.7385, Close = 80.75,Volume = 615249365},
new ChartData { Date = new DateTime(2012, 05, 07), Open= 80.2143, High =
82.2685,Low = 79.8185, Close = 80.9585,Volume = 541742692},
new ChartData { Date = new DateTime(2012, 05, 14), Open= 80.3671, High =
81.0728,Low = 74.5971, Close = 75.7685,Volume = 708126233},
new ChartData { Date = new DateTime(2012, 05, 21), Open= 76.3571, High =
82.3571,Low = 76.2928, Close = 80.3271,Volume = 682076215},
new ChartData { Date = new DateTime(2012, 05, 28), Open= 81.5571, High =
83.0714,Low = 80.0743, Close = 80.1414,Volume = 480059584}
};
}

```

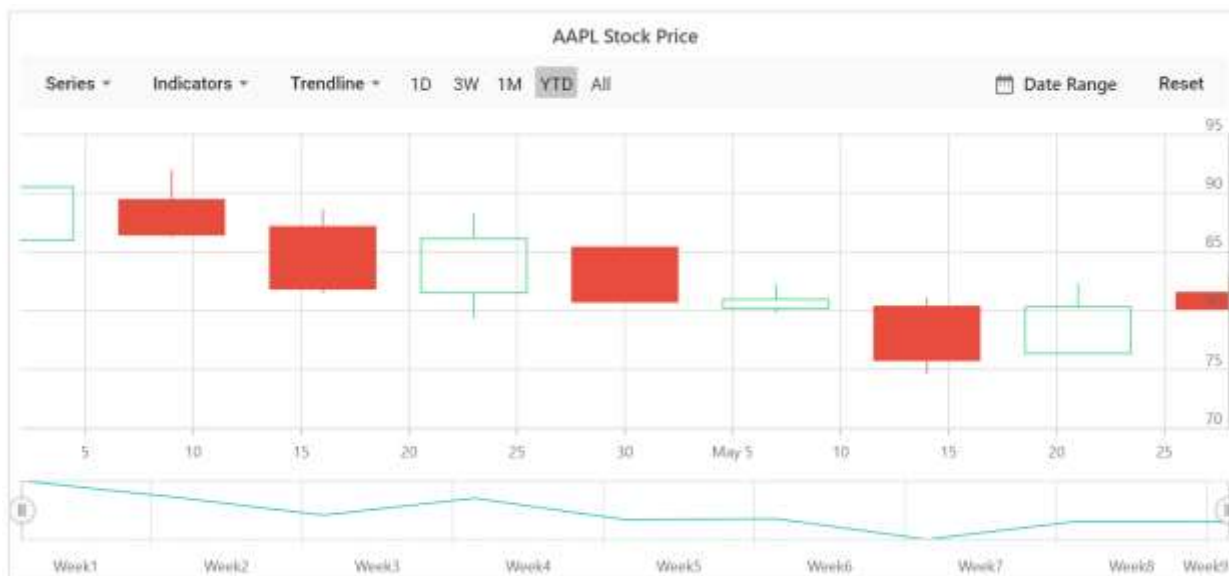


Disable Export and print

Empty the value of `ExportType` to disable the Export.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfStockChart Title="AAPL Stock Price" ExportType="new List<ExportType>() {
}">
<StockChartSeriesCollection>
<StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Candle"
XName="Date" Name="google"></StockChartSeries>
</StockChartSeriesCollection>
</SfStockChart>
@code {
public class ChartData
{
public DateTime Date;
public Double Open;
public Double Low;
public Double Close;
public Double High;
public Double Volume;
}
public List<ChartData> StockDetails = new List<ChartData>
{
new ChartData { Date = new DateTime(2012, 04, 02), Open= 85.9757, High =
90.6657,Low = 85.7685, Close = 90.5257,Volume = 660187068},
new ChartData { Date = new DateTime(2012, 04, 09), Open= 89.4471, High =
92,Low = 86.2157, Close = 86.4614,Volume = 912634864},
new ChartData { Date = new DateTime(2012, 04, 16), Open= 87.1514, High =
88.6071,Low = 81.4885, Close = 81.8543,Volume = 1221746066},
new ChartData { Date = new DateTime(2012, 04, 23), Open= 81.5157, High =
88.2857,Low = 79.2857, Close = 86.1428,Volume = 965935749},
new ChartData { Date = new DateTime(2012, 04, 30), Open= 85.4, High =
85.4857,Low = 80.7385, Close = 80.75,Volume = 615249365},
new ChartData { Date = new DateTime(2012, 05, 07), Open= 80.2143, High =
82.2685,Low = 79.8185, Close = 80.9585,Volume = 541742692},
new ChartData { Date = new DateTime(2012, 05, 14), Open= 80.3671, High =
81.0728,Low = 74.5971, Close = 75.7685,Volume = 708126233},
new ChartData { Date = new DateTime(2012, 05, 21), Open= 76.3571, High =
82.3571,Low = 76.2928, Close = 80.3271,Volume = 682076215},
new ChartData { Date = new DateTime(2012, 05, 28), Open= 81.5571, High =
83.0714,Low = 80.0743, Close = 80.1414,Volume = 480059584}
};
}
```

Events in Blazor Stock Chart Component

In this section, the list of events of Stock Chart component is provided which will be triggered for appropriate stock chart actions. The events should be provided to the Stock Chart using **StockChartEvents** component.

The following are the number of events supported for Stock Chart component.

- [Loaded](#)
- [OnPointClick](#)
- [PointMoved](#)
- [RangeChange](#)
- [OnStockChartMouseClicked](#)
- [OnStockChartMouseDown](#)
- [OnStockChartMouseLeave](#)
- [OnStockChartMouseMove](#)
- [OnStockChartMouseUp](#)

Loaded

[Loaded](#) event triggers after stock chart is rendered.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfStockChart>
  <StockChartEvents Loaded="StockChartLoaded"></StockChartEvents>
  <StockChartSeriesCollection>
    <StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Column"
      XName="Date" YName="Y">
    </StockChartSeries>
  </StockChartSeriesCollection>
</SfStockChart>

@code {
  public void StockChartLoaded(IStockChartEventArgs args)
  {
```

```
// Here you can customize your code
}
public class ChartData
{
    public DateTime Date;
    public Double Y;
}
public List<ChartData> StockDetails = new List<ChartData>
{
    new ChartData { Date = new DateTime(2012, 04, 02), Y= 100},
    new ChartData { Date = new DateTime(2012, 04, 09), Y= 10 },
    new ChartData { Date = new DateTime(2012, 04, 16), Y= 500},
    new ChartData { Date = new DateTime(2012, 04, 23), Y= 80 },
    new ChartData { Date = new DateTime(2012, 04, 30), Y= 200},
    new ChartData { Date = new DateTime(2012, 05, 07), Y= 600},
    new ChartData { Date = new DateTime(2012, 05, 14), Y= 50 },
    new ChartData { Date = new DateTime(2012, 05, 21), Y= 700},
    new ChartData { Date = new DateTime(2012, 05, 28), Y= 90 }
};
}
```

OnPointClick

[OnPointClick](#) event triggers when data point is clicked.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfStockChart>
<StockChartEvents OnPointClick="PointClick"></StockChartEvents>
<StockChartSeriesCollection>
<StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Column"
XName="Date" YName="Y">
</StockChartSeries>
</StockChartSeriesCollection>
</SfStockChart>
@code {
    public void PointClick(IPointEventArgs args)
    {
        // Here you can customize your code
    }
    public class ChartData
    {
        public DateTime Date;
        public Double Y;
    }
    public List<ChartData> StockDetails = new List<ChartData>
    {
        new ChartData { Date = new DateTime(2012, 04, 02), Y= 100},
        new ChartData { Date = new DateTime(2012, 04, 09), Y= 10 },
        new ChartData { Date = new DateTime(2012, 04, 16), Y= 500},
        new ChartData { Date = new DateTime(2012, 04, 23), Y= 80 },
        new ChartData { Date = new DateTime(2012, 04, 30), Y= 200},
        new ChartData { Date = new DateTime(2012, 05, 07), Y= 600},
        new ChartData { Date = new DateTime(2012, 05, 14), Y= 50 },
        new ChartData { Date = new DateTime(2012, 05, 21), Y= 700},
        new ChartData { Date = new DateTime(2012, 05, 28), Y= 90 }
    }
}
```

```
};
}
```

PointMoved

[PointMoved](#) event triggers when moving mouse over the data point.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfStockChart>
<StockChartEvents PointMoved="OnPointMoved"></StockChartEvents>
<StockChartSeriesCollection>
<StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Column"
XName="Date" YName="Y">
</StockChartSeries>
</StockChartSeriesCollection>
</SfStockChart>
@code {
public void OnPointMoved(IPointEventArgs args)
{
// Here you can customize your code
}
public class ChartData
{
public DateTime Date;
public Double Y;
}
public List<ChartData> StockDetails = new List<ChartData>
{
new ChartData { Date = new DateTime(2012, 04, 02), Y= 100},
new ChartData { Date = new DateTime(2012, 04, 09), Y= 10 },
new ChartData { Date = new DateTime(2012, 04, 16), Y= 500},
new ChartData { Date = new DateTime(2012, 04, 23), Y= 80 },
new ChartData { Date = new DateTime(2012, 04, 30), Y= 200},
new ChartData { Date = new DateTime(2012, 05, 07), Y= 600},
new ChartData { Date = new DateTime(2012, 05, 14), Y= 50 },
new ChartData { Date = new DateTime(2012, 05, 21), Y= 700},
new ChartData { Date = new DateTime(2012, 05, 28), Y= 90 }
};
}
```

RangeChange

[RangeChange](#) event triggers when range is changed.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfStockChart>
<StockChartEvents RangeChange="RangeChanged"></StockChartEvents>
<StockChartSeriesCollection>
<StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Column"
XName="Date" YName="Y">
</StockChartSeries>
</StockChartSeriesCollection>
</SfStockChart>
@code {
```

```

public void RangeChanged(IRangeChangeEventArgs args)
{
    // Here you can customize your code
}
public class ChartData
{
    public DateTime Date;
    public Double Y;
}
public List<ChartData> StockDetails = new List<ChartData>
{
    new ChartData { Date = new DateTime(2012, 04, 02), Y= 100},
    new ChartData { Date = new DateTime(2012, 04, 09), Y= 10 },
    new ChartData { Date = new DateTime(2012, 04, 16), Y= 500},
    new ChartData { Date = new DateTime(2012, 04, 23), Y= 80 },
    new ChartData { Date = new DateTime(2012, 04, 30), Y= 200},
    new ChartData { Date = new DateTime(2012, 05, 07), Y= 600},
    new ChartData { Date = new DateTime(2012, 05, 14), Y= 50 },
    new ChartData { Date = new DateTime(2012, 05, 21), Y= 700},
    new ChartData { Date = new DateTime(2012, 05, 28), Y= 90 }
};
}

```

OnStockChartMouseClicked

[OnStockChartMouseClicked](#) event triggers when clicking the stock chart.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfStockChart>
<StockChartEvents
OnStockChartMouseClicked="StockChartMouseClickedHandler"></StockChartEvents>
<StockChartSeriesCollection>
<StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Column"
XName="Date" YName="Y">
</StockChartSeries>
</StockChartSeriesCollection>
</SfStockChart>
@code {
public void StockChartMouseClickedHandler(IMouseEventArgs args)
{
    // Here you can customize your code
}
public class ChartData
{
    public DateTime Date;
    public Double Y;
}
public List<ChartData> StockDetails = new List<ChartData>
{
    new ChartData { Date = new DateTime(2012, 04, 02), Y= 100},
    new ChartData { Date = new DateTime(2012, 04, 09), Y= 10 },
    new ChartData { Date = new DateTime(2012, 04, 16), Y= 500},
    new ChartData { Date = new DateTime(2012, 04, 23), Y= 80 },
    new ChartData { Date = new DateTime(2012, 04, 30), Y= 200},
    new ChartData { Date = new DateTime(2012, 05, 07), Y= 600},

```

```
new ChartData { Date = new DateTime(2012, 05, 14), Y= 50 },
new ChartData { Date = new DateTime(2012, 05, 21), Y= 700},
new ChartData { Date = new DateTime(2012, 05, 28), Y= 90 }
};
}
```

OnStockChartMouseDown

[OnStockChartMouseDown](#) event triggers when mouse down over the stock chart.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfStockChart>
<StockChartEvents
OnStockChartMouseDown="StockChartMouseDownHandler"></StockChartEvents>
<StockChartSeriesCollection>
<StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Column"
XName="Date" YName="Y">
</StockChartSeries>
</StockChartSeriesCollection>
</SfStockChart>
@code {
public void StockChartMouseDownHandler(IMouseEventArgs args)
{
// Here you can customize your code
}
public class ChartData
{
public DateTime Date;
public Double Y;
}
public List<ChartData> StockDetails = new List<ChartData>
{
new ChartData { Date = new DateTime(2012, 04, 02), Y= 100},
new ChartData { Date = new DateTime(2012, 04, 09), Y= 10 },
new ChartData { Date = new DateTime(2012, 04, 16), Y= 500},
new ChartData { Date = new DateTime(2012, 04, 23), Y= 80 },
new ChartData { Date = new DateTime(2012, 04, 30), Y= 200},
new ChartData { Date = new DateTime(2012, 05, 07), Y= 600},
new ChartData { Date = new DateTime(2012, 05, 14), Y= 50 },
new ChartData { Date = new DateTime(2012, 05, 21), Y= 700},
new ChartData { Date = new DateTime(2012, 05, 28), Y= 90 }
};
}
```

OnStockChartMouseLeave

[OnStockChartMouseLeave](#) event triggers when cursor leaves the stock chart.

ASPX-CS

```
@using Syncfusion.Blazor.Charts
<SfStockChart>
<StockChartEvents
OnStockChartMouseLeave="StockChartMouseLeaveHandler"></StockChartEvents>
<StockChartSeriesCollection>
```

```

<StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Column"
XName="Date" YName="Y">
</StockChartSeries>
</StockChartSeriesCollection>
</SfStockChart>
@code {
public void StockChartMouseLeaveHandler(IMouseEventArgs args)
{
// Here you can customize your code
}
public class ChartData
{
public DateTime Date;
public Double Y;
}
public List<ChartData> StockDetails = new List<ChartData>
{
new ChartData { Date = new DateTime(2012, 04, 02), Y= 100},
new ChartData { Date = new DateTime(2012, 04, 09), Y= 10 },
new ChartData { Date = new DateTime(2012, 04, 16), Y= 500},
new ChartData { Date = new DateTime(2012, 04, 23), Y= 80 },
new ChartData { Date = new DateTime(2012, 04, 30), Y= 200},
new ChartData { Date = new DateTime(2012, 05, 07), Y= 600},
new ChartData { Date = new DateTime(2012, 05, 14), Y= 50 },
new ChartData { Date = new DateTime(2012, 05, 21), Y= 700},
new ChartData { Date = new DateTime(2012, 05, 28), Y= 90 }
};
}

```

OnStockChartMouseMove

[OnStockChartMouseMove](#) event triggers when hovering the stock chart.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfStockChart>
<StockChartEvents
OnStockChartMouseMove="StockChartMouseMoveHandler"></StockChartEvents>
<StockChartSeriesCollection>
<StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Column"
XName="Date" YName="Y">
</StockChartSeries>
</StockChartSeriesCollection>
</SfStockChart>
@code {
public void StockChartMouseMoveHandler(IMouseEventArgs args)
{
// Here you can customize your code
}
public class ChartData
{
public DateTime Date;
public Double Y;
}
public List<ChartData> StockDetails = new List<ChartData>
{

```

```

new ChartData { Date = new DateTime(2012, 04, 02), Y= 100},
new ChartData { Date = new DateTime(2012, 04, 09), Y= 10 },
new ChartData { Date = new DateTime(2012, 04, 16), Y= 500},
new ChartData { Date = new DateTime(2012, 04, 23), Y= 80 },
new ChartData { Date = new DateTime(2012, 04, 30), Y= 200},
new ChartData { Date = new DateTime(2012, 05, 07), Y= 600},
new ChartData { Date = new DateTime(2012, 05, 14), Y= 50 },
new ChartData { Date = new DateTime(2012, 05, 21), Y= 700},
new ChartData { Date = new DateTime(2012, 05, 28), Y= 90 }
};
}

```

OnStockChartMouseUp

[OnStockChartMouseUp](#) event triggers when mouse is up.

ASPX-CS

```

@using Syncfusion.Blazor.Charts
<SfStockChart>
<StockChartEvents
OnStockChartMouseUp="StockChartMouseUpHandler"></StockChartEvents>
<StockChartSeriesCollection>
<StockChartSeries DataSource="@StockDetails" Type="ChartSeriesType.Column"
XName="Date" YName="Y">
</StockChartSeries>
</StockChartSeriesCollection>
</SfStockChart>
@code {
public void StockChartMouseUpHandler(IMouseEventArgs args)
{
// Here you can customize your code
}
public class ChartData
{
public DateTime Date;
public Double Y;
}
public List<ChartData> StockDetails = new List<ChartData>
{
new ChartData { Date = new DateTime(2012, 04, 02), Y= 100},
new ChartData { Date = new DateTime(2012, 04, 09), Y= 10 },
new ChartData { Date = new DateTime(2012, 04, 16), Y= 500},
new ChartData { Date = new DateTime(2012, 04, 23), Y= 80 },
new ChartData { Date = new DateTime(2012, 04, 30), Y= 200},
new ChartData { Date = new DateTime(2012, 05, 07), Y= 600},
new ChartData { Date = new DateTime(2012, 05, 14), Y= 50 },
new ChartData { Date = new DateTime(2012, 05, 21), Y= 700},
new ChartData { Date = new DateTime(2012, 05, 28), Y= 90 }
};
}

```

Stock Events in Blazor Stock Chart Component

The stock events are used to mark specific events such as market open and close, highest or lowest price reached, year/quarter start and end on a Chart for a specific date. In this section, **SplineSeries** is used to represent selected data value. One can customize the specific data value using **StockEvents**.

Date

The [Date](#) property is used to display the stock event in the Chart at the specified time. For example, Quarter 1 ends on March 31, 2021, and the stock event date must be set to March 31, 2021. More customization options are available in the immediate sections such as **Text**, **Type**, **Description**, and so on.

Text

The text acts as a quick and short representation of the stock event, such as **Q1** for Quarter 1 and **High** for highest price over a period; it can be customized separately for each stock event using the [Text](#) property.

Type

The [Type](#) property can be used to set the background shape of a stock event, with options such as **Circle**, **Square**, **Flag**, **Text**, **Sign**, **Triangle**, **InvertedTriangle**, **ArrowUp**, **ArrowDown**, **ArrowLeft**, and **ArrowRight**.

Background

The [Background](#) property of the stock event is used to customize the color of the background shape, which accepts values in hex and rgba as a valid CSS color string.

Description

The [Description](#) property specifies the content of the stock event tooltip that appears on the Chart when the mouse is moved over the stock event. The description will be displayed as the text of the tooltip. For instance, if [Text](#) **Q1** contains a [Description](#) as **Quarter 1**, the tooltip will show **Quarter 1**.

ASPX-CS

```
@page "/"
@using Syncfusion.Blazor.Charts
@using Newtonsoft.Json
<SfStockChart Title="AAPL Stock Price" SeriesType="@SeriesValue">
  <StockChartStockEvents>
    @foreach (StockEventDetails stockEvent in StockEvents)
    {
      <StockChartStockEvent Date=@stockEvent.Date Text=@stockEvent.Text
        Description=@stockEvent.Description Type=@stockEvent.Type
        Background=@stockEvent.Background ShowOnSeries=@stockEvent.ShowOnSeries>
      <StockChartStockEventsBorder
        Color=@stockEvent.BorderColor></StockChartStockEventsBorder>
      <StockChartStockEventsTextStyle
        Color=@stockEvent.TextColor></StockChartStockEventsTextStyle>
      </StockChartStockEvent>
    }
  </StockChartStockEvents>
  <StockChartPrimaryXAxis>
    <StockChartAxisMajorGridLines Width="0"></StockChartAxisMajorGridLines>
    <StockChartAxisCrosshairTooltip
      Enable="true"></StockChartAxisCrosshairTooltip>
    </StockChartPrimaryXAxis>
    <StockChartPrimaryYAxis>
      <StockChartAxisLineStyle Width="0"></StockChartAxisLineStyle>
```



```

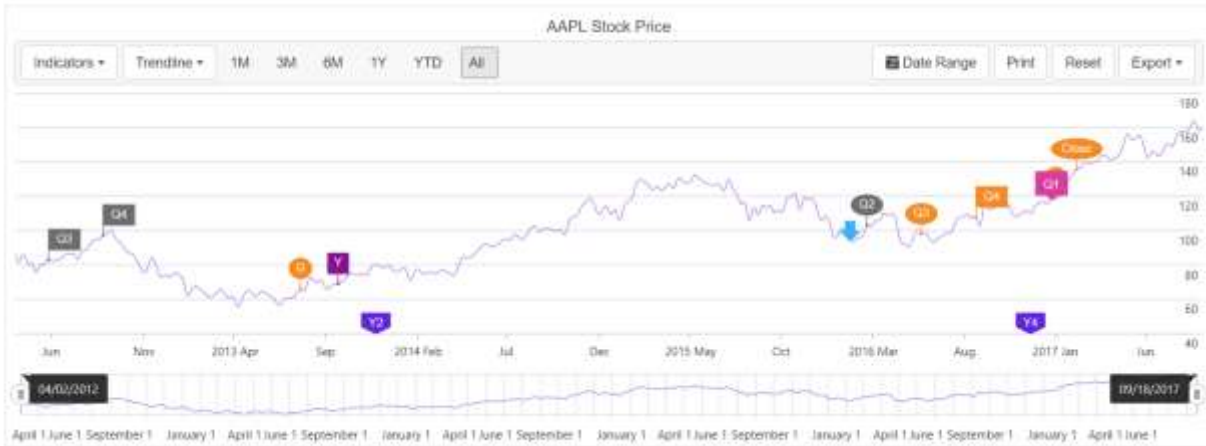
<StockChartAxisMajorTickLines Width="0"></StockChartAxisMajorTickLines>
<StockChartAxisCrosshairTooltip
Enable="true"></StockChartAxisCrosshairTooltip>
</StockChartPrimaryYAxis>
<StockChartSeriesCollection>
<StockChartSeries DataSource="@DataSource"
Type="ChartSeriesType.Spline"></StockChartSeries>
</StockChartSeriesCollection>
<StockChartChartArea>
<StockChartChartAreaBorder Width="0"></StockChartChartAreaBorder>
</StockChartChartArea>
</SfStockChart>
@code{
public class ChartData
{
public DateTime date { get; set; }
public double open { get; set; }
public double low { get; set; }
public double close { get; set; }
public double high { get; set; }
public double volume { get; set; }
}
public class StockEventDetails
{
public DateTime Date { get; set; }
public string Text { get; set; }
public string Description { get; set; }
public FlagType Type { get; set; } = FlagType.Flag;
public string Background { get; set; }
public string BorderColor { get; set; }
public string TextColor { get; set; } = "#FFFFFF";
public bool ShowOnSeries { get; set; } = true;
}
public List<ChartSeriesType> SeriesValue = new List<ChartSeriesType>();
private ChartData[] DataSource;
private List<StockEventDetails> StockEvents;
protected override async Task OnInitializedAsync()
{
GetStockEventsDetails();
await Task.Run(() =>
{
DataSource =
JsonConvert.DeserializeObject<ChartData[]>(System.IO.File.ReadAllText("./www
root/data/chart-data.json"));
});
}
private void GetStockEventsDetails()
{
StockEvents = new List<StockEventDetails>()
{
new StockEventDetails(){ Date = new DateTime(2011, 03, 01), Text ="Q2",
Description = "2012 Quarter2", Type = FlagType.Flag, Background = "#6C6D6D",
BorderColor = "#6C6D6D"},
new StockEventDetails(){ Date = new DateTime(2012, 03, 20), Text ="Open",
Description = "Markets opened", Background = "#f48a21", Type =
FlagType.Circle, BorderColor = "#f48a21"},

```

```

new StockEventDetails(){ Date = new DateTime(2012, 06, 01), Text ="Q3",
Description = "2013 Quarter3", Background = "#6C6D6D", Type = FlagType.Flag,
BorderColor = "#6C6D6D"},
new StockEventDetails(){ Date = new DateTime(2012, 09, 01), Text ="Q4",
Description = "2013 Quarter4", Background = "#6C6D6D", Type = FlagType.Flag,
BorderColor = "#6C6D6D"},
new StockEventDetails(){ Date = new DateTime(2013, 07, 30), Text ="G",
Description = "Google Stock", Background = "#f48a21", Type =
FlagType.Circle, BorderColor = "#f48a21"},
new StockEventDetails(){ Date = new DateTime(2013, 10, 01), Text ="Y",
Description = "Yahoo Stock", Background = "#841391", Type = FlagType.Square,
BorderColor = "#841391"},
new StockEventDetails(){ Date = new DateTime(2013, 12, 04), Text ="Y2",
Description = "Year 2013", Background = "#6322e0", Type = FlagType.Pin,
BorderColor = "#6322e0", ShowOnSeries = false},
new StockEventDetails(){ Date = new DateTime(2016, 03, 01), Text ="Q2",
Description = "2014 Quarter2", Background = "#6C6D6D", Type =
FlagType.Circle, BorderColor = "#6C6D6D"},
new StockEventDetails(){ Date = new DateTime(2016, 06, 01), Text ="Q3",
Description = "2014 Quarter3", Background = "#f48a21", Type =
FlagType.Circle, BorderColor = "#f48a21"},
new StockEventDetails(){ Date = new DateTime(2016, 09, 01), Text ="Q4",
Description = "2014 Quarter4", Background = "#f48a21", Type = FlagType.Flag,
BorderColor = "#f48a21"},
new StockEventDetails(){ Date = new DateTime(2016, 12, 01), Text ="Y4",
Description = "Year 2015", Background = "#6322e0", Type = FlagType.Pin,
BorderColor = "#6322e0", ShowOnSeries= false},
new StockEventDetails(){ Date = new DateTime(2016, 02, 02), Text ="End",
Description = "Markets closed", Background = "#3ab0f9", Type =
FlagType.ArrowDown, BorderColor = "#3ab0f9"},
new StockEventDetails(){ Date = new DateTime(2017, 01, 07), Text ="A",
Description = "Amazon Stock", Background = "#f48a21", Type =
FlagType.Circle, BorderColor = "#f48a21"},
new StockEventDetails(){ Date = new DateTime(2017, 01, 02), Text ="Q1",
Description = "AAPL Stock", Background = "#dd3c9f", Type = FlagType.Text,
BorderColor = "#dd3c9f"},
new StockEventDetails(){ Date = new DateTime(2017, 02, 12), Text ="Close",
Description = "Markets closed", Background = "#f48a21", Type =
FlagType.Circle, BorderColor = "#f48a21"}
};
}
}

```



Tabs

Getting Started with Blazor Tabs Component

This section briefly explains about how to include a **Tabs** in Blazor server-side application. Refer [Getting Started with Syncfusion Blazor for Server-Side in Visual Studio 2019](#) page for the introduction and configuring the common specifications.

To get start quickly with Blazor Tabs, you can check on this video:

{% youtube

"youtube:https://www.youtube.com/watch?v=8OgywAA_XHg"%}

Importing Syncfusion Blazor component in the application

You can use any one of the below standards to install the Syncfusion Blazor library in your application.

Using Syncfusion Blazor individual NuGet Packages [New standard]

Starting with Volume 4, 2020 (v18.4.0.30) release, Syncfusion provides [individual NuGet packages](#) for Syncfusion Blazor components. We highly recommend this new standard for your Blazor production applications. Refer to [this section](#) to know the benefits of the individual NuGet packages.

1. Install **Syncfusion.Blazor.Navigations** NuGet package to the application using the **NuGet Package Manager**.
2. The client-side style resources can be added through [CDN](#) or from [NuGet](#) package in the element of the `~/Pages/_Host.cshtml` page.

HTML

```
<head>
...
...
<link href="_content/Syncfusion.Blazor.Themes/bootstrap4.css"
rel="stylesheet" />
</head>
```

Waring: Syncfusion.Blazor package should not be installed along with [individual NuGet packages](#). Hence, add the above **Syncfusion.Blazor.Themes** static web assets (styles) in the application.

Using Syncfusion.Blazor NuGet Package [Old standard]

Warning: If you prefer the above new standard (individual NuGet packages), then skip this section. Using both old and new standards in the same application will throw ambiguous compilation errors.

1. Install **Syncfusion.Blazor** NuGet package to the newly created application by using the **NuGet Package Manager**.
2. Add the client-side style resources through CDN or from NuGet package in the `element` of the `~/Pages/_Host.cshtml` page.

HTML

```
<head>
....
....
<link href="_content/Syncfusion.Blazor/styles/bootstrap4.css"
rel="stylesheet" />
@*<link href="https://cdn.syncfusion.com/blazor/{{ site.blazorversion
}}/styles/bootstrap4.css" rel="stylesheet" />*@
</head>
```

For Internet Explorer 11 kindly refer the polyfills. Refer the [documentation](#) for more information.

HTML

```
<head>
...
<link href="https://cdn.syncfusion.com/blazor/{{ site.blazorversion
}}/styles/bootstrap4.css" rel="stylesheet" />
<script
src="https://github.com/Daddoon/Blazor.Polyfill/releases/download/3.0.1/blaz
or.polyfill.min.js"></script>
...
</head>
```

Adding component package to the application

Open `~/_Imports.razor` file and import the **Syncfusion.Blazor.Navigations** package.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
```

Add SyncfusionBlazor service in Startup file

Open the **Startup.cs** file and add services required by Syncfusion components using **services.AddSyncfusionBlazor()** method. Add this method in the **ConfigureServices** function as follows.

CSHARP

```
using Syncfusion.Blazor;
namespace BlazorApplication
{
    public class Startup
    {
        ....
    }
}
```

```

.....
public void ConfigureServices(IServiceCollection services)
{
    .....
    .....
    services.AddSyncfusionBlazor();
}
}
}

```

Adding Tabs component to the application

Now, add the Syncfusion Blazor Tabs component in any web page (razor) in the **Pages** folder. For example, the Tabs component is added in the **~/Pages/Index.razor** page.

ASPX-CS

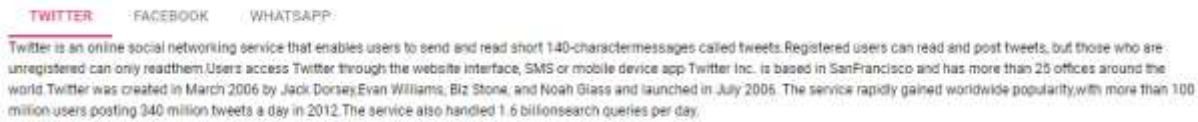
```

@using Syncfusion.Blazor.Navigations
<SfTab>
<TabItems>
<TabItem Content="Twitter is an online social networking service that
enables users to send and read short 140-character messages called
tweets. Registered users can read and post tweets, but those who are
unregistered can only read them. Users access Twitter through the website
interface, SMS or mobile device app. Twitter Inc. is based in San Francisco
and has more than 25 offices around the world. Twitter was created in March
2006 by Jack Dorsey, Evan Williams, Biz Stone, and Noah Glass and launched in
July 2006. The service rapidly gained worldwide popularity, with more than
100 million users posting 340 million tweets a day in 2012. The service also
handled 1.6 billion search queries per day.">
<ChildContent>
<TabHeader Text="Twitter"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="Facebook is an online social networking service
headquartered in Menlo Park, California. Its website was launched on February
4, 2004, by Mark Zuckerberg with his Harvard College roommates and fellow
students Eduardo Saverin, Andrew McCollum, Dustin Moskovitz and Chris
Hughes.">
<ChildContent>
<TabHeader Text="Facebook"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="WhatsApp Messenger is a proprietary cross-platform instant
messaging client for smartphones that operates under a subscription business
model. It uses the Internet to send text messages, images, video, user
location and audio media messages to other users using standard cellular
mobile numbers. As of February 2016, WhatsApp had a userbase of up to one
billion, [10] making it the most globally popular messaging
application. WhatsApp Inc., based in Mountain View, California, was acquired
by Facebook Inc. on February 19, 2014, for approximately US$19.3 billion.">
<ChildContent>
<TabHeader Text="Whatsapp"></TabHeader>
</ChildContent>
</TabItem>
</TabItems>
</SfTab>

```

Run the application

After successful compilation of the application, simply press **F5** to run the application.



Initialize Tab Content using Template

Tab provides support to render content using **ContentTemplate** property. You can give preferred content inside the **ContentTemplate** element.

ContentTemplate property supports **RenderFragment** type to render content.

The following code explains how to initialize tab content using **ContentTemplate**.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<SfTab>
<TabItems>
<TabItem>
<ChildContent>
<TabHeader Text="HTML"></TabHeader>
</ChildContent>
<ContentTemplate>
<div>HyperText Markup Language, commonly referred to as HTML, is the
standard markup language used to create web pages. Along with CSS, and
JavaScript, HTML is a cornerstone technology, used by most websites to
create visually engaging web pages, user interfaces for web applications,
and user interfaces for many mobile applications.[1] Web browsers can read
HTML files and render them into visible or audible web pages. HTML describes
the structure of a website semantically along with cues for presentation,
making it a markup language, rather than a programming language.</div>
</ContentTemplate>
</TabItem>
<TabItem>
<ChildContent>
<TabHeader Text="Java"></TabHeader>
</ChildContent>
<ContentTemplate>
<div>Java is a set of computer software and specifications developed by Sun
Microsystems, later acquired by Oracle Corporation, that provides a system
for developing application software and deploying it in a cross-platform
computing environment. Java is used in a wide variety of computing platforms
from embedded devices and mobile phones to enterprise servers and
supercomputers. While less common, Java applets run in secure, sandboxed
environments to provide many features of native applications and can be
embedded in HTML pages.</div>
</ContentTemplate>
</TabItem>
<TabItem>
<ChildContent>
<TabHeader Text="JavaScript"></TabHeader>
```

```

</ChildContent>
<ContentTemplate>@DynamicContent</ContentTemplate>
</TabItem>
</TabItems>
</SfTab>
@code{
public RenderFragment DynamicContent = builder =>
{
builder.AddContent(1, "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.[5] More recently, however, it has become common in both game
development and the creation of desktop applications.");
};
}

```

HTML

JAVA

JAVASCRIPT

HyperText Markup Language, commonly referred to as HTML, is the standard markup language used to create web pages. Along with CSS, and JavaScript, HTML is a cornerstone technology, used by most websites to create visually engaging web pages, user interfaces for web applications, and user interfaces for many mobile applications.[1] Web browsers can read HTML files and render them into visible or audible web pages. HTML describes the structure of a website semantically along with cues for presentation, making it a markup language, rather than a programming language.

Two way binding of SelectedItem

The property [SelectedItem](#) supports two way property binding, in the following code example if either the value is changed in numeric text box or selected tab item is changed, it will reflect in both the value of numeric text box and selected tab item.

ASPX-CS

```

@using Syncfusion.Blazor.Inputs
@using Syncfusion.Blazor.Navigations
Selected Tab
<SfNumericTextBox TValue="int" @bind-Value="@SelectedTab" Min="0" Max="4"
Width="200px"></SfNumericTextBox>
<SfTab @bind-SelectedItem="SelectedTab">
<TabItems>
<TabItem Content="HyperText Markup Language, commonly referred to as HTML,
is the standard markup language used to create web pages. Along with CSS,
and JavaScript, HTML is a cornerstone technology, used by most websites to
create visually engaging web pages, user interfaces for web applications,
and user interfaces for many mobile applications.[1] Web browsers can read
HTML files and render them into visible or audible web pages. HTML describes
the structure of a website semantically along with cues for presentation,
making it a markup language, rather than a programming language.">
<ChildContent>
<TabHeader Text="HTML"></TabHeader>

```

```

</ChildContent>
</TabItem>
<TabItem Content="C# is intended to be a simple, modern, general-purpose,
object-oriented programming language. Its development team is led by Anders
Hejlsberg. The most recent version is C# 5.0, which was released on August
15, 2012.">
<ChildContent>
<TabHeader Text="C Sharp (C#)"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="Java is a set of computer software and specifications
developed by Sun Microsystems, later acquired by Oracle Corporation, that
provides a system for developing application software and deploying it in a
cross-platform computing environment. Java is used in a wide variety of
computing platforms from embedded devices and mobile phones to enterprise
servers and supercomputers. While less common, Java applets run in secure,
sandboxed environments to provide many features of native applications and
can be embedded in HTML pages.">
<ChildContent>
<TabHeader Text="Java"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="The command-line compiler, VBC.EXE, is installed as part
of the freeware .NET Framework SDK. Mono also includes a command-line VB.NET
compiler. The most recent version is VB 2012, which was released on August
15, 2012.">
<ChildContent>
<TabHeader Text="VB.Net"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="Xamarin is a San Francisco, California based software
company created in May 2011[3] by the engineers that created Mono,[4] Mono
for Android and MonoTouch that are cross-platform implementations of the
Common Language Infrastructure (CLI) and Common Language Specifications
(often called Microsoft .NET). With a C#-shared codebase, developers can use
Xamarin tools to write native Android, iOS, and Windows apps with native
user interfaces and share code across multiple platforms.[5] Xamarin has
over 1 million developers in more than 120 countries around the World as of
May 2015.">
<ChildContent>
<TabHeader Text="Xamarin"></TabHeader>
</ChildContent>
</TabItem>
</TabItems>
</SfTab>
@code {
private int SelectedTab = 0;
}

```

Selected Tab 0

HTML C# SHARP(C#) JAVA VB.NET XAMARIN

HyperText Markup Language, commonly referred to as HTML, is the standard markup language used to create web pages. Along with CSS, and JavaScript, HTML is a cornerstone technology, used by most websites to create visually engaging web pages, user interfaces for web applications, and user interfaces for many mobile applications [1] Web browsers can read HTML files and render them into visible or audible web pages. HTML describes the structure of a website semantically along with cues for presentation, making it a markup language, rather than a programming language.

See Also

1. [Getting Started with Syncfusion Blazor for client-side in .NET Core CLI](#)
2. [Getting Started with Syncfusion Blazor for client-side in Visual Studio 2019](#)
3. [Getting Started with Syncfusion Blazor for server-side in .NET Core CLI](#)

Responsive Modes in Blazor Tabs Component

The following section explains about rendering tab when its width exceeds the viewable area or particularly in a given width. The available modes are as follows:

- Scrollable
- Popup

Scrollable

The default `OverflowMode` is `Scrollable`. Scrollable display mode supports displaying the Tab header items in a single line with horizontal scrolling enabled, when the item overflows to the available space.

- The right and left navigation arrow is added at the start and end of the Tab header through which user can navigate towards overflowed items of the Tab header.
- You can also see the overflowed items using touch and swipe action on the header and content section.
- By default, navigation icon in the left direction is disabled, you can see the overflowed items by moving in the right direction.
- By clicking the arrow or by holding the arrow continuously, you can see the overflowed items.



- In devices the navigation icons are not available. You can touch and swipe to see the overflowed items of the Tab header.

HTML

C SHARP(C#)

JAVA

VB.NET

XAMARIN

ASP.NET

HyperText Markup Language, commonly referred to as HTML, is the standard markup language used to create web pages. Along with CSS, and JavaScript, HTML is a cornerstone technology, used by most websites to create visually engaging web pages, user interfaces for web applications, and user interfaces for many mobile applications.[1] Web browsers can read HTML files and render them into visible or audible web pages. HTML describes the structure of a website semantically along with cues for presentation, making it a markup language, rather than a programming language.

ASPX-CS

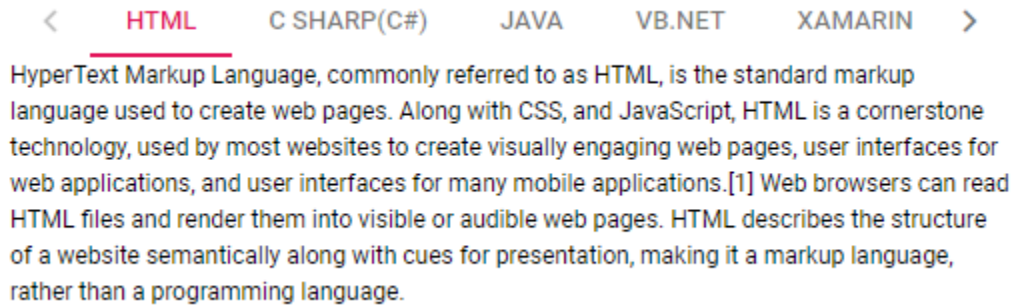
```

@using Syncfusion.Blazor.Navigations
<SfTab OverflowMode="OverflowMode.Scrollable" Width="500px">
  <TabItems>
    <TabItem Content="HyperText Markup Language, commonly referred to as HTML,
    is the standard markup language used to create web pages. Along with CSS,
    and JavaScript, HTML is a cornerstone technology, used by most websites to
    create visually engaging web pages, user interfaces for web applications,
    and user interfaces for many mobile applications.[1] Web browsers can read
    HTML files and render them into visible or audible web pages. HTML describes
    the structure of a website semantically along with cues for presentation,
    making it a markup language, rather than a programming language.">
      <ChildContent>
        <TabHeader Text="HTML"></TabHeader>
      </ChildContent>
    </TabItem>
    <TabItem Content="C# is intended to be a simple, modern, general-purpose,
    object-oriented programming language. Its development team is led by Anders
    Hejlsberg. The most recent version is C# 5.0, which was released on August
    15, 2012.">
      <ChildContent>
        <TabHeader Text="C Sharp (C#)"></TabHeader>
      </ChildContent>
    </TabItem>
    <TabItem Content="Java is a set of computer software and specifications
    developed by Sun Microsystems, later acquired by Oracle Corporation, that
    provides a system for developing application software and deploying it in a
    cross-platform computing environment. Java is used in a wide variety of
    computing platforms from embedded devices and mobile phones to enterprise
    servers and supercomputers. While less common, Java applets run in secure,
    sandboxed environments to provide many features of native applications and
    can be embedded in HTML pages.">
      <ChildContent>
        <TabHeader Text="Java"></TabHeader>
      </ChildContent>
    </TabItem>
    <TabItem Content="The command-line compiler, VBC.EXE, is installed as part
    of the freeware .NET Framework SDK. Mono also includes a command-line VB.NET
    compiler. The most recent version is VB 2012, which was released on August
    15, 2012.">
      <ChildContent>
        <TabHeader Text="VB.Net"></TabHeader>
      </ChildContent>
    </TabItem>
  </TabItems>
</SfTab>
  
```

```

</TabItem>
<TabItem Content="Xamarin is a San Francisco, California based software
company created in May 2011[3] by the engineers that created Mono,[4] Mono
for Android and MonoTouch that are cross-platform implementations of the
Common Language Infrastructure (CLI) and Common Language Specifications
(often called Microsoft .NET). With a C#-shared codebase, developers can use
Xamarin tools to write native Android, iOS, and Windows apps with native
user interfaces and share code across multiple platforms.[5] Xamarin has
over 1 million developers in more than 120 countries around the World as of
May 2015.">
<ChildContent>
<TabHeader Text="Xamarin"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="ASP.NET is an open-source server-side web application
framework designed for web development to produce dynamic web pages. It was
developed by Microsoft to allow programmers to build dynamic web sites, web
applications and web services. It was first released in January 2002 with
version 1.0 of the .NET Framework, and is the successor to Microsoft's
Active Server Pages (ASP) technology. ASP.NET is built on the Common
Language Runtime (CLR), allowing programmers to write ASP.NET code using any
supported .NET language. The ASP.NET SOAP extension framework allows ASP.NET
components to process SOAP messages.">
<ChildContent>
<TabHeader Text="ASP.NET"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="The ASP.NET MVC is a web application framework developed
by Microsoft, which implements the model-view-controller (MVC) pattern. It
is open-source software, apart from the ASP.NET Web Forms component which is
proprietary. In the later versions of ASP.NET, ASP.NET MVC, ASP.NET Web API,
and ASP.NET Web Pages (a platform using only Razor pages) will merge into a
unified MVC 6.The project is called ASP.NET vNext.">
<ChildContent>
<TabHeader Text="ASP.NET MVC"></TabHeader>
</ChildContent>
</TabItem>
<TabItem 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.[5] More recently, however, it has become common in both game
development and the creation of desktop applications.">
<ChildContent>
<TabHeader Text="JavaScript"></TabHeader>
</ChildContent>
</TabItem>
</TabItems>
</SfTab>

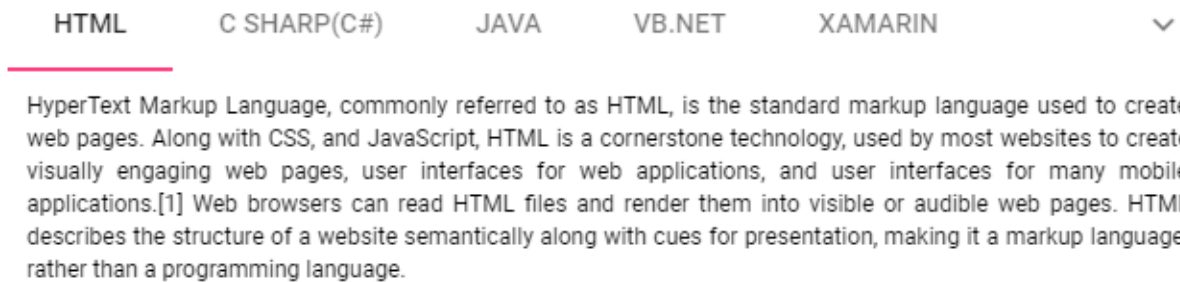
```



Popup

The **Popup** is the another type of [OverflowMode](#) in which the Tab container holds the items that can be placed within the available space. The rest of the overflowing items for which there is no space to fit within the viewing area are moved to overflow popup container.

- The items placed in popup can be viewed by opening the popup with the help of drop-down icon given at the end of the Tab header.
- If the popup height exceeds the height of the visible area, you can scroll through the popup items and select one.



ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<SfTab OverflowMode="OverflowMode.Popup" Width="500px">
  <TabItems>
    <TabItem Content="HyperText Markup Language, commonly referred to as HTML,
    is the standard markup language used to create web pages. Along with CSS,
    and JavaScript, HTML is a cornerstone technology, used by most websites to
    create visually engaging web pages, user interfaces for web applications,
    and user interfaces for many mobile applications.[1] Web browsers can read
    HTML files and render them into visible or audible web pages. HTML describes
    the structure of a website semantically along with cues for presentation,
    making it a markup language, rather than a programming language.">
      <ChildContent>
        <TabHeader Text="HTML"></TabHeader>
      </ChildContent>
    </TabItem>
    <TabItem Content="C# is intended to be a simple, modern, general-purpose,
    object-oriented programming language. Its development team is led by Anders
```

```
Hejlsberg. The most recent version is C# 5.0, which was released on August
15, 2012.">
<ChildContent>
<TabHeader Text="C Sharp (C#)"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="Java is a set of computer software and specifications
developed by Sun Microsystems, later acquired by Oracle Corporation, that
provides a system for developing application software and deploying it in a
cross-platform computing environment. Java is used in a wide variety of
computing platforms from embedded devices and mobile phones to enterprise
servers and supercomputers. While less common, Java applets run in secure,
sandboxed environments to provide many features of native applications and
can be embedded in HTML pages.">
<ChildContent>
<TabHeader Text="Java"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="The command-line compiler, VBC.EXE, is installed as part
of the freeware .NET Framework SDK. Mono also includes a command-line VB.NET
compiler. The most recent version is VB 2012, which was released on August
15, 2012.">
<ChildContent>
<TabHeader Text="VB.Net"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="Xamarin is a San Francisco, California based software
company created in May 2011[3] by the engineers that created Mono,[4] Mono
for Android and MonoTouch that are cross-platform implementations of the
Common Language Infrastructure (CLI) and Common Language Specifications
(often called Microsoft .NET). With a C#-shared codebase, developers can use
Xamarin tools to write native Android, iOS, and Windows apps with native
user interfaces and share code across multiple platforms.[5] Xamarin has
over 1 million developers in more than 120 countries around the World as of
May 2015.">
<ChildContent>
<TabHeader Text="Xamarin"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="ASP.NET is an open-source server-side web application
framework designed for web development to produce dynamic web pages. It was
developed by Microsoft to allow programmers to build dynamic web sites, web
applications and web services. It was first released in January 2002 with
version 1.0 of the .NET Framework, and is the successor to Microsoft's
Active Server Pages (ASP) technology. ASP.NET is built on the Common
Language Runtime (CLR), allowing programmers to write ASP.NET code using any
supported .NET language. The ASP.NET SOAP extension framework allows ASP.NET
components to process SOAP messages.">
<ChildContent>
<TabHeader Text="ASP.NET"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="The ASP.NET MVC is a web application framework developed
by Microsoft, which implements the model-view-controller (MVC) pattern. It
is open-source software, apart from the ASP.NET Web Forms component which is
proprietary. In the later versions of ASP.NET, ASP.NET MVC, ASP.NET Web API,
```

and ASP.NET Web Pages (a platform using only Razor pages) will merge into a unified MVC 6. The project is called ASP.NET vNext.">

```
<ChildContent>
<TabHeader Text="ASP.NET MVC"></TabHeader>
</ChildContent>
</TabItem>
<TabItem 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.[5] More recently, however, it has become common in both game
development and the creation of desktop applications.">
<ChildContent>
<TabHeader Text="JavaScript"></TabHeader>
</ChildContent>
</TabItem>
</TabItems>
</SfTab>
```

HTML C SHARP(C#) JAVA VB.NET XAMARIN ▼

HyperText Markup Language, commonly referred to as HTML, is the standard markup language used to create web pages. Along with CSS, and JavaScript, HTML is a cornerstone technology, used by most websites to create visually engaging web pages, user interfaces for web applications, and user interfaces for many mobile applications.[1] Web browsers can read HTML files and render them into visible or audible web pages. HTML describes the structure of a website semantically along with cues for presentation, making it a markup language, rather than a programming language.

Header in Blazor Tabs Component

This section explains about modifying the style of Tab header.

Styles

The header styles can be customized by adding predefined classes in the Tab root element. The predefined CSS class names are as follows:

- **e-fill:** The Selected Tab header background is set as solid fill.
- **e-background:** Tab header has a solid fill background, and the selected header has a highlighted border.
- **e-background e-accent:** Tab header has a solid fill background, and the selected header has a highlighted border with accent color.

If the above custom style classes are not included in the root element, the default style is applied to the Tab items.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
@using Syncfusion.Blazor.DropDowns
<div id="TabHeader">
<div class="row">
```

```

<div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
<label>Header Style</label>
</div>
<div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
<SfDropDownList TValue="string" DataSource="@HeaderStylesData" Width="30%"
TItem="DropdownFields" @bind-Index="index">
<DropDownListEvents TValue="string" ValueChange="OnHeaderStyleChange"
TItem="DropdownFields"></DropDownListEvents>
<DropDownListFieldSettings Value="Value"
Text="Text"></DropDownListFieldSettings>
</SfDropDownList>
</div>
</div>
</div>
<SfTab CssClass="@HeaderStyles">
<TabItems>
<TabItem Content="@Content0">
<ChildContent>
<TabHeader Text="Twitter"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="@Content1">
<ChildContent>
<TabHeader Text="Facebook"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="@Content2">
<ChildContent>
<TabHeader Text="WhatsApp"></TabHeader>
</ChildContent>
</TabItem>
</TabItems>
</SfTab>
@code{
private int? index { get; set; } = 0;
public string HeaderStyles { get; set; } = "";
public class DropdownFields
{
public string Value { get; set; }
public string Text { get; set; }
}
List<DropdownFields> HeaderStylesData = new List<DropdownFields>()
{
new DropdownFields() { Value= "", Text= "Default" },
new DropdownFields() { Value= "e-background e-accent", Text= "Accent" },
new DropdownFields() { Value= "e-fill", Text= "Fill" }
};
public string Content0 = "Twitter is an online social networking service
that enables users to send and read short 140-character " +
"messages called 'tweets'. Registered users can read and post tweets, but
those who are unregistered can only read " +
"them. Users access Twitter through the website interface, SMS or mobile
device app Twitter Inc. is based in San " +
"Francisco and has more than 25 offices around the world. Twitter was
created in March 2006 by Jack Dorsey, " +
"Evan Williams, Biz Stone, and Noah Glass and launched in July 2006. The
service rapidly gained worldwide popularity, " +

```

```

"with more than 100 million users posting 340 million tweets a day in
2012.The service also handled 1.6 billion " +
"search queries per day.";
public string Content1 = "Facebook is an online social networking service
headquartered in Menlo Park, California. Its website was " +
"launched on February 4, 2004, by Mark Zuckerberg with his Harvard College
roommates and fellow students Eduardo " +
"Saverin, Andrew McCollum, Dustin Moskovitz and Chris Hughes.The founders
had initially limited the website " +
"membership to Harvard students, but later expanded it to colleges in the
Boston area, the Ivy League, and Stanford " +
"University. It gradually added support for students at various other
universities and later to high-school students.";
public string Content2 = "WhatsApp Messenger is a proprietary cross-platform
instant messaging client for smartphones that operates " +
"under a subscription business model. It uses the Internet to send text
messages, images, video, user location and " +
"audio media messages to other users using standard cellular mobile numbers.
As of February 2016, WhatsApp had a user " +
"base of up to one billion,[10] making it the most globally popular
messaging application. WhatsApp Inc., based in " +
"Mountain View, California, was acquired by Facebook Inc. on February 19,
2014, for approximately US$19.3 billion.";
public void
OnHeaderStyleChange(Syncfusion.Blazor.DropDowns.ChangeEventArgs<string,
DropdownFields> args)
{
    HeaderStyles = args.Value;
}
}
<style>
.e-content .e-item {
font-size: 12px;
padding: 10px;
text-align: justify;
}
</style>

```

Header Style

fill

TWITTER

FACEBOOK

WHATSAPP

Twitter is an online social networking service that enables users to send and read short 140-character messages called 'tweets'. Registered users can read and post tweets, but those who are unregistered can only read them. Users access Twitter through the website interface, SMS or mobile device app. Twitter Inc. is based in San Francisco and has more than 25 offices around the world. Twitter was created in March 2006 by Jack Dorsey, Evan Williams, Biz Stone, and Noah Glass and launched in July 2006. The service rapidly gained worldwide popularity, with more than 100 million users posting 340 million tweets a day in 2012. The service also handled 1.6 billion search queries per day.

Icon positions

The position of the Tab header icons can be customized using the [IconPosition](#) property. This property depends on the header items [IconCSS](#) property. By default, Tab header icon is placed on left position. The position values are as follows:

- **Left:** Icon is placed on the left of the Tab header item.
- **Right:** Icon is placed on the right of the Tab header item.
- **Top:** Icon is placed on the top of the Tab header item.
- **Bottom:** Icon is placed on the bottom of the Tab header item.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
@using Syncfusion.Blazor.DropDowns
<div id="headerIconsTab">
<div class="row">
<div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
<label>Icon Position</label>
</div>
<div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
<SfDropDownList TValue="string" DataSource="@HeaderIconsData" Width="30%"
TItem="HeaderIcons" @bind-Index="index">
<DropDownListEvents ValueChange="ChangeHeaderIcon" TValue="string"
TItem="HeaderIcons"></DropDownListEvents>
<DropDownListFieldSettings Value="Value"
Text="Text"></DropDownListFieldSettings>
</SfDropDownList>
</div>
</div>
</div>
<SfTab>
<TabItems>
<TabItem Content="@Content0">
<ChildContent>
<TabHeader Text="Twitter" IconCss="e-twitter"
IconPosition="@PositionValue"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="@Content1">
<ChildContent>
<TabHeader Text="Facebook" IconCss="e-facebook"
IconPosition="@PositionValue"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="@Content2">
<ChildContent>
<TabHeader Text="WhatsApp" IconCss="e-whatsapp"
IconPosition="@PositionValue"></TabHeader>
</ChildContent>
</TabItem>
</TabItems>
</SfTab>
<style>
.e-content .e-item {
font-size: 12px;
```

```
padding: 10px;
text-align: justify;
}
@@font-face {
font-family: 'Socialicons';
src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgTlMvMvltCfsAAAEoAAAAVmNtYXCnKKeOAAABrAAAAEhnbHlml19
XagAAAgwAABhQaGVhZA8dCeEAAADQAAAAANmhoZWEIUQQMAAAArAAAACRobXR4LAAAAAAAYAAAA
sbG9jYR3AIwWAAAH0AAAAGG1heHABIAIAAAABCAAAACBuYW1l0X1q/wAAGlwAAAJVcG9zdGX5D00
AABY0AAAAkwABAAAEAAAAAFwEAAAAAAD9AABAAAAAIAAAAAAAAAAAACwABAAAAAQAA+iTiP18
PPPUACwQAAAAANYFYngAAAAAlgVieAAAAAAD9AP0AAAACAACAAAAAIAAAAAALafQACwAAAAA
AAgAAAAoACgAAAP8AAAAAQAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAUGZFZABApwCnCQQAAAAAXAQAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
EAAAABAAAAAQAAAAEAAAAAABAAAAAQAAAAAIAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAFAA
EADQAAAAEAAQAAQAPwn//wAApWd//wAAAAEABAAAAEAAgADAAQABQAGAAcACAAJAAoAAAAAAiQ
CzgOMBU4F/gZYB9QICAO+DCgABAAAAAAD0gPzAFUA4gF3AfMAAAEzHwYHFQ8EFR8IPwUfBRUPCCM
vFj0BPwoXNw8fHQEfDhUPAT8CHwkzPyA9Ai8iDwIFHwcPIysBLWYjDwI/AS8PNT8oHx4BDxAdAR8
PHQEHPwE7AR8EMz8dNS8kiW8FAYkFEgQDAyQDAQECAyIBAQMSEgkUCw4vBQQFChsGBQdqAgIBAwM
DCAoMDA0NBgYPEA8PFxYVFBQTEhITEREPDgwkCQQEBCICBAQFChMJBQUFBTURDxAPDw8ODg4NDQw
MDAsLCgkJCQgHBwCFBQUEAwICAQEDAgQEBgYHBwkJCgsOAgEmiwMEBAQUFRQVFRQVFRQVFRQVFRQV
VDw4ODg0NDAwLCwoKCQkICAcGBgUEBAQCAgICAgMEBAYGBgcICQkJCgsLCwwNDQ0ODg4PDw8QEBA
QEBEREREQEQHcBgUEBAICAQEBAQEDAwQFBQYHCAgICgoLCwwNDQ4ODxAREhISEhMTEExMUExQUFRQ
VGxsaGgcIBwfXNgeBAQ8KCgoIBwcGBQUdAwIBAQECAwMDBQUFBgcHCAgICgkKCwsMDAwNDg0ODw8
PEBAQEhISEhISEhIREREREREQERAQDw8PDw8ODQ0NDQwLDAoKCgkJCACh/aaQEB0cGhgWFBIRDgw
LCACeAwICAwMFBQYHBwgJCgoLAgE9+AYFBSMeHx4fIB8fFhQUFBQSExIRERAQE8ODhAQDQ0LCQg
HBgQCAgICBAQEBCgYGCAGJCgoLCwwMDQ4ODg8PEBAREBESEhISExITGBcZGRgZGBgXAU4CagMEXAk
FBAQFBCQCAwMGHcKfQkMIwIBAwYAwIBKQECBSgFBgULCgkHBgMBAQEDAwcICgWMDg8PEhITFBu
WGBgSEyYJCAgIBwcHDBEGAgEBAagEBAQGBgYHCAgJCgkLCwsMDAwNDQ4ODg8PDw8QEBAQERITEhE
SEREREBEPEA8QDhMEBASFNgeBAQEJCAcGBQMCAQEDAwUGCAgHCAgJCgoKCwsMDA0NDQ4ODg8ODxA
PEA8QEBAREBAQEBIQERAQDw8ODg0NDQwMCwoKCQkICAcGBgUFAwMDAQEBAQEC6RISExISExISEhM
SEhESEREREREQE8QDg8NDg0MDAwLCwoJCQcHBgUEAwICAgIEBggJAwECVnCFBAQVEBEREhESEhI
TExMTFBMUehEREBEQEBApDw8PDg4ODQ0MDAwLCwoKCgkICAcHBWUGBQQDAGIBAQEBAgIDBAQFBQc
GCAGICQoKCgsMDA0NDQ4PDw8QEBEBBwgIEhMUFhcYGRscHR8fIiIjFBMTEhMSEhIREhEREBEQEAM
EAwt1YQINCAyEAgIEBAUGBgICQoKDAwNDg8PERUVFhcWGBcYGBkZGRkaGxOTehISEhEREBAQDw8
ODg4MDQwLCgoKCQgHBWYFBQMDAwEBAGMFBQgIAAAAAEAAAAAAzoD9ACWAAATDwYVERUfHTsBPw4
9AS8OIy8PNSEzPw4vDyE9AS8ODwblCAYFBAQCAgECAwMEBQUGBwgICQoKCwwMDA0NDQ0ODg4PDw8
QEBDQCGsKCQkJCAgHBWUEBAICAIEBAUHBwgICQkJCgsK0QoKCgoJCAgIBwcFBAMDAQEBKQoJCgg
JCAgHBWUFBAQCAQEBAQIEBAUFBgHCAkJCQkK/tcCagMFBQYHCAkICQoJCwoLCwoJCQkIA9AJCgs
KDAwMDf4LExMTEhESEREQEBApDw4PDQ4MDA0KCQgIBgYEBAMCAgEBAwQFBwcJCQoKCwsMDA0NDAs
MCwoKCQkHBWUEAwEBAQEDBAUGCAgJCgoLCwwMDVgCAwMFBgcICQkJCgsLCwwLDAsKCgoJCAgHBgU
EAgIBtQ0MDAsLCgoJCQcHBQQDAQEBAQMEBQYIAAABAAAAAAP0A90AqAAAAT8DMx8MHQEPDSsBLxo
PCBc/Ah8LEx8PMz8dLw8jDw4CSAoTEhIRCAcGBWUFBBQDAwICAgEDBQoPExYWFAsLCgQFBAUFBAU
FBQUJCQkJCyQEBCUGBwcICAKKCgsLDQwODQ8RERQfI5IvNRMGBWYGBWYGBgYGDAsMWAcICAgICQk
JCQkKCgoLCgsJExITFBQVFRYWFIMkJTEWFRQREQ8NDAsJBWYFAwEBAQEDBAUHBwkJCwwNDhAQEGs
YGBYWFBBQSEhAQDg4MCwsC2wQGBQIBAgICAwQEBCUHBgcICBMSDxEdIiUoIXsOCgcCAQIEBAYICBU
bHyU37xQTERAPDQwKCQgGBQMCAQEDBgLDhofkUInCwIBAgQEBCwJCwscISf+pBYUEXIQDg4LCwk
IBgUDAgEDBACJDA0REhQXJysxRiEhIB4eHRwbGhkZFxcVFROXfHqTERAODQwKCAcGBAMBAQMFBwk
MDQ8REXUXGhsdAAUAAAAA/ED9ABCAKoA6wESAYQAAAEaQ8NKwEvDjU/EB8OJR0BHw8hPw8TLwM
hHwUVDxEvEzU/ASchDMFFR8PPw8vDw8OAR8HFQ8JIIy8GPQI/BjMlHQEPBC8DNS8DDwMVDwIjLwM
9Ai8BIw8EFQ8DIy8CNw8KFxUfASUzPwgZHWkhPwI1LxAlDwICkAMDBQcHCQkKDAwMDQ4ODwwMCws
LCgoJDwsJCAYFAGCAwQGBgcICQkKCwsMDA0LCw8ODg4MDAsKCQkHBgQEAv1/AQMFBGkJDAwODwg
RERITEwJpExMSEhEQDw4MDAUJBWYEAgEBAgEF/uYOCwkGBAICBAYHCQsMDg8QERETExQTFRQVFBQ
UFBMTEhAPDg0LCQgGBAMCAgEBAwMEBQYHCAkC/uoFAGEBASwBAWUGCQoLDQ0PERESEhQUFBQTEhE
QEA4MDAKJBgUDAQEDBQcICgsNDg8QERISFBQUFBMSERAQDgWMCggHBAMCPAYGBgQEAWEBQEBAM
EBAQFBmgHBgYEBAMCAwMEBQUFBjX90AECBAUUBQEBAQEBAgIRAgIBAQIFEAKDAwECBAQEBAQDwI
CAwUWAwIBAQQQDwwLCQgFAwEBAQEEATEEBBYUFYRYWFxcYFxcXfXyVFBgFBQYBjWYCAgICBAYHCQo
LDA4ODxAIERIR/d8FAQIB9wcHDg0NDAwLCgkIBWYFBAICAQgEBQYGDQwODg8REBINDQwMCwsKCgk
```

IBwcGBAQCAQEBAgQFBggICgoLDQ0NDg9929sUEXISERAPDgwMBQkHBgQCAQMFBgkJDAwODwgQERI
TEwHBBgMBARYXFxcXFxcWFhUUFMBREQ8ODAsJCAYEAwIBAgQFBwkKDA0PDxERExQPEA8PDw8PDw8
ODw4ODg4OAEBAQKPCgoUEhIREBANDQsKCAcFAwEBAwUHCAoLDQ4PEBESExQUFBMTEhEQDw4NCwo
IBwUDAQEDBQcICgsNDg8QEhITEwGSAQICBAUFBgdsBQQFBAQDAgIBAQECAGQFBQYHawcHBgUDAgE
BR2h1CAMCAQEBAgIF5wMCAQEBAQED6gUCAQEDAwbbBQICAQIDAwmG0ggEAQICAgTKAQ00EBASEhQ
VEiRdAgIBAQITDg0JCAYDAQFBwoMDhQCAQEBAQNuJBIRERAPDg4NCwoJCAMFBAEBAQIEAAAAAAM
AAAAA/QD3QADAFcAlwAANzMRIwUVIzclIxExET8OHw8RMxEvGw8MAR8PPw41Lw8PDhnW1gIjAQH
W1gIDBQgKCwCHBwgJCQoKCw4NDAsKCAgHBwUEBAICAQHWAQICAgQDBQUFBgYHBwcJCAkJCgoKCws
LDBgZGhQUEREPDg0MCwoJCQ79xAEBawMFBgYHCAkKCwsMDA4PDQwLCwoJCQcGBgUDAwIBAQMEBAY
GCAgJCgoLDQwODQ0MDA0KCQkHBwYEBAMBIGKFWwICW/17AXcUDA0ODgWGBQUBAMCAQEBAgMFBQc
ICgoLDA0NDw8Q/qcBhBIREBAPDw4NDQwMCwoKCQkICAcGBgUFBAWGAwEBAgMEBgYHCAgICQkSARI
MCwsKCgkICAgGBQUEAwEBAQEDBAUFBgICAKKCgsLDAsLCwsJCggIBWYGBAQDAQEBAQMEBAYGBwg
ICgkLCwsAAAAAABAAAAAPuA/QARgAAExEVHwYhESM1MzU/DzMVIw8GFTMVIxehPwYRLwYhDYSAGQ
FBwgKCgHPb24BAwMGBggJCgsMDQ0ODwgPlUcLCwkIBgQDe3sBBQoKCAcFBAICBAUHCAoK/IUKCgk
HBwQDA7v8igYLCgkIBgQDAZuFUBAQDw4ODQwLCgkIBwUEAgGFAwQHCAkKDDOF/mUDBAYICQoLA4I
LCgkIBgQDAQQFBwgKCwAAAAAGAAAAAP0A/QAOABEAIAABBQEqAUwAAAEPCR0Bhw07AT8NPQEvCCM
PASUVMxUjFSM1IzUzNSUPBRUfDTsBPww1Lw4jDwU3ByMfCA8PHw4dAQ8OLw8/DS8FPwIH1y8NPQE
/DwEVHw8hPw8RITChLw8hDw4BCgMTCwsFBAQEAgICAwQGBgCICgoLDAwODg8NDQwLCgkICAYGBQQ
DAwEBAQIDBAgMDiYRNw0B9nR0TXNz/kAFawMDAQIBAgMDBAQGBgCICQkKDAsIBwCHBwYFBQYFAwM
BAQECAwMEBQYGBwgJCQoLDAcIBwCHBwX+MTAQDggIAwICAQEBAQEDAwMICgsMDAsGAgEBAQECAwY
iGQoFCQcDAgIBAwQFCAGLDA0PERITFRYYFRISEA8NDAsKCAcGBAMCAQEBAwUHCQsOERMUFb0xCAC
DAwEBAQIFGQ4ODQ0LCgoICAcFBQQAQgMDBGICGwICBESEhESEBD+pwEDBQYJCgsNDg8IEBISExQ
CahQTEExIREA8ODQsGCQcGBAL8GAED5gIDBgCICgsNDg4QCBIRExP9lhMTExERE40ODQsKCAcGAwF
KAQkHCAYGBggICQkKCgkICAgHBgYFBQMDAgIBAgMDBAUFBgYHBwcICQgHBwYGBgYLCwwcBQPYck9
yck5zZwYGBwCHDxELCgWLCwsKCgkJBwUFAwECAwMDBAUHBwcIBw0QCwwLDAsMCgoKCAcGBAMBAgI
DAwQFLRkQDwwPCAgJCgoLCQkICAgNDAsKCQwJBQYGBQYEBACbFQsGDA4HCAGJCQ4NDQwNCwwKCgg
IBgYDAwEBAgMDBQYGBwgJCAoJCgsKCwUMDAwMDAsKCQYFBQUKDAYHCAgJBw0BAgQEBQcHCAkJCgo
KCwsLDQ4NDQ0MDAsGBgkIBQQCAQH+EAoKExMSERAQDQ0LBgkHBgQCAQMFBgkKCw0NEAgQEhITFAJ
HKwQSEhIQDw8NDAsJBQcFBAIBAwQHCAkLDA0PDxASEhIAAAAAAgAAAAAD7gP0AEAaHAAAAARUzFSM
RHws/BxUPAY8OESM1Pw81ER8OMyEzPw4RLw4jISMPDQIbysoDBgUICgYHCAgJCgsLDQ4PEBESE0Q
tICIiEREQDw8ODQwKCgCHBANuGBkVDw4ODgYFBgUEBAMCAv5fAQECAwQEBQUGBwCHCAgJCAM0CAk
ICACHWYFBQQAQwEBAQEBAgMEBAUFBgCHBwgICQj8zAgJCAgHBwcGBQUBAMCAQON0H/+9BIMCAk
HBAMDAgEBAQEBAgMDBQYHeA4GAwEBAgIDBAUFBgJCwsNDxABVGwKDXANDxEUCwwMDQ00DxAQEvz
CCQgICACHWYGBAUDAwICAgIDAUEBgYHBwcICAgJAz4JCAgIBwCHBwYEBQMDAgICAgMDBQQGBgC
HBwgICAAAAGAAAAAD7APzAPgBqAAAR8LFQ8MIY8QKwEPDh8bHQEPfi8WPQE/DTMfEjM/Di8ePQE
/Fh8CBR8HDwMfhjSBPwIfBzM/HTUvBz8CPQEvHiMPai8HIw8dAnALFhMSDw4LCQgFBAIBAgIDAww
FBgUGBgCGCAwLCQgHChQLCwsHBwkJCgsNDQwMCwsJCggIBWYFBAMDAQEBAgMEBQcHCRMTdxojFhQ
TEA8OCwUFAwQCAwEBAgIEBQUHCAgKCgWMDg4PEBERehMTFBUZGBYWFRMSEgsLCwoJCQgIBWYFBQM
CAgECAgMDBAUFBQYGBgYHCAcLCgkIBwCHBwCHBwKDAcPERMZDQ0MDAsKCQgHBgUEAwEBAQICAgM
EBAsMDQ8bTSIfGxkMCwsKCQgIBWYFBQMCAgICBAQGBgCICQoLDA0NDw8PERERExIUHxwb/bsBAgM
EBQcHCQUADAQEBAQMFBQYICAKLCwwNDg8QEBESEhMUFBWFRcWGBcYGBYWFRUPDxAQEBEREQ4ODg0
NDQ0MDAwMCwoLCgkJCQgHBwCGBgQFAwMDAgEBAQIDBAUGBgQEAgIDBAUHBwkJCgWMDQ4PDxERERM
TFBQVFRYWFxcYGBgUFRQTEBESEhITFBMODg4NDQ0NDA0LDAsKCwoJCQkIBwCHBwYEBQMDAwIBAzC
ECAoLDAwNDQ4NDg0NBgYGBQYKBQQDAwICAQECAUHDSEODQoEBAMCAgIBAQICAwMEBQUFBQYGBgY
GCAcHBgYFBQYIBx0GDAgJCgsNDg8JCAkKCgsLCwwPDg0ODQwMDAsLCgoICAgHBgUEBAMCAQEBAgI
EBQYICAYIBwkJCQoKCwsLCgsLCgoHBgYGBQUFBQQAQwMCAQEBAgUGCAkLGg0LCgkICAYDBAMCAQI
DBAQFBgYGBwCIBwkIDQcFBgUEBQgIBgYHEgkJCgoHBgCICAKJCgoLDAwMDg0NDQ0MDAsLCgoKCQg
IBWYGBQYGBAwMBAQEBawRbEhMSEREREBAXFxyGBgYFxcWFhUVFBQTEExEREQ8PDg0MDA0KCAcHBQ
DAGICAwCGBgUDAwEBAQIDAwmFBAYGBwCHCAkJCQoLDA0NDw8MDAwMDQ0NDQ4ODHAQE8PDw4OGB0ZGhg
YFxyGWFxUWFRQUEXISERAQDw4NDAsLCQkHBgUFAwEBAgIDCggHBgUDAgEBAgMDAwUEBgYGCACICQk
JCgoLCwwLDQwNDQ0NDg4AAAAACwAAAAAD8wOYABEAMwBbAKYAYwDTARcBOQFjAZGBoQAAAQ8DMzc
vBisBDwEnDwIdAh8FOWE/BjUvBisBDwEnFwcfBDM/Bic1MxUnNw8GIY8HNyclHwsVIxUfBjsBPwY
1MxcVDw0vCzU/CycVPwMfCR0CDwgjLwQPAREjFSMVIzUjNTcPCxUfDyE/DzUvDiMhIw8BJR8DFQ8
GKwEvBjU/Bx8DFR8KPwUHMzUjFQ8GKwEvBjUjDwcdAR8LowE/CTUvDg8DFTM1NyMHJyMDIqQDAgJ
CAgECAwQFBQYGCgUG2QQDAgIDBAUFBgYGBQYEBQICAECAgUEBgUGBgUF6wEBAwUCAwMEBwgEAgE
BAQFFOQEDBAYMDhAQDwgGBgYFAgIEAQECHQoLCgkJCAcFBAMCAxcBAQMDBQQFBhAGBQQEAgIBMwM
BAQMCBAQMBwCICAgIDxAPDg8HBgYEAwMBAQECAgQGBggJCQkJC+UQDg4NDQUGBwCFBAQCAgIDAuU

```

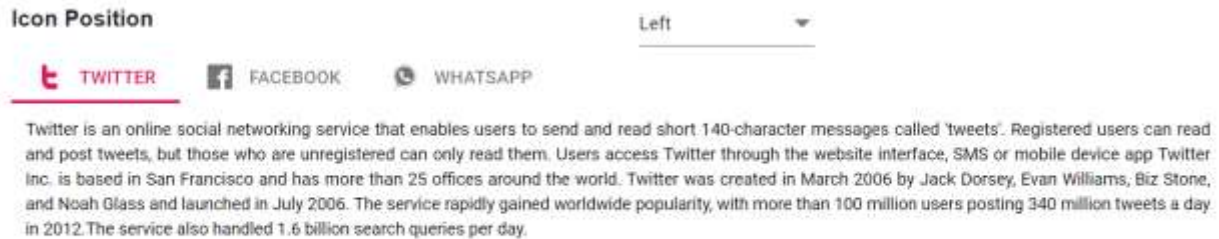
GBgcICQoLBQwNFAQ50VJFRyAPDQ0LCwkJBwMFAwIBAgQGBwkJCwsNDQ8PDxARAqAQEQ8PDw0NCws
KCAcDBQMQAQIEBgcICgsLDQ0PDxAQEP1gERAPAYoEAgIBAgICBAUGBQYGBgUFBAICAQECaWQFBQU
GBgUGdgEEAgQEBQQGBWgIBwcGBgoKAUw7AQEDAwQEBQUFBQQEAWIBAUC+CggGBgMCAgICBAMECgY
HCQsLCwwMCwoSBwcHCAUEAgEBAgIEAwQHBWgJCgsMDg8NDMPKv1AuLVEBbAMEBB8bBQMEAwICAQE
CCAMDawOAAwMDAgICAQECAGIDAwOAAwMDAwICAQECF4kgBQUCAQECBAQEAQIDCJrYARsEBAMHBQQ
CAQICAwQEBRoPmw0BAQIDAwQFBgYMDBgvMgYEAwIDAQEBAQMCaWQEGREGBgUFBBQQECQQEAWICAQE
BAwYHBQYGBWgJCgpDDwWLCggHBgYDAwMBQFQHBQQBAQICAwUFBQYHBwhwCgkJCAGGBQQDAQEBBAU
LEgEBIib+/yVJBQUGBwcICQkFCgsK9gsLCgoJCQgHBWYFBQMDAQEBAQMDBQUGBwcICQkFCgoL9wo
LCgoJCQgHBWYFBQMDAgID9wQEBAV3BAUDBAMCAQECaWQEBAR3BQQEBAMCAQEBaQJ3GAwRBAUEAwM
DAQEBAQEBaWcKEuCvAwMDAgMBAQEBaWIDAwoVAgUHCAoKDA0OQxgOBwcGCgQEAgMCAQICBQQFBws
KDxZTCgkHBgYGBQYFBQMDAgEBAQIEPayslG9vAAAAABIA3gABAAAAAAAAAAAAAABAAAAAABAAAs
AAQABAAAAAAAAACAADAABAAAAAADAAAsAEwABAAAAAAAEAAAsAHgABAAAAAAAFAAAsAKQABAAAAAA
GAAsANAABAAAAAAKACwAPwABAAAAAALABIAawADAAEECQAAAAIAfQADAAEECQABABYAfwADAAE
ECQACAA4AlQADAAEECQADABYAowADAAEECQAEABYAuwADAAEECQAFABYazwADAAEECQAGABYA5QA
DAAEECQAKAFgA+wADAAEECQALACQBvYBTb2NpYWxpY29uc1JlZ3VsYXJlZ3VsYXJlZ3VsYXJlZ3Vs
hbG1jb25zVmVyc2lvbiAxLjBTb2NpYWxpY29uc0Zvb2NpYWxpY29uc0Zvb2NpYWxpY29uc0Zvb2Np
pb24gTWV0cm8gU3R1ZGlvd3d3LnN5bmNmdXNpb24uY29tACAAUwBvAGMAaQBhAGwAaQBjAG8AbgB
zAFIAZQBnAHUAbABhAHIAUwBvAGMAaQBhAGwAaQBjAG8AbgBzAFMAbwBjAGkAYQBsAGkAYwBvAG4
AcwBWAGUAcgBzAGkAbwBuACAAMQAuADAAUwBvAGMAaQBhAGwAaQBjAG8AbgBzAEYAbwBuAHQAIAIB
nAGUAbgBIAHIAIYQB0AGUAZAAGAHUAcwBpAG4AZwAgAFMAeQBvAGMAZgBIAHMAaQBvAG4AIAIBNAGU
AdABYAG8AIAIBTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBvAGMAZgBIAHMAaQBvAG4ALgBjAG8AbQA
AAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAsBAgEDAQQBBQEGAQcBCAEJAQoBCwE
MAAh3aGf0c2FwcAd0d2l0dGVyBXZpbWVvCWluc3RhZ3JhbQhsaW5rZWRpbgYWNlYm9vawtnb29
nbGUtcGxlcwZ0dWl1bHIIc2t5cGUtMDEIeW9ldHViZTEAAAA=) format('trueType');
font-weight: normal;
font-style: normal;
}
.e-tab .e-tab-icon {
font-family: 'Socialicons' !important;
}
.e-tab .e-icons.e-tab-icon {
position: relative;
top: 1px;
}
.e-twitter:before {
content: '\a701';
}
.e-facebook:before {
content: '\a705';
}
.e-whatsapp:before {
content: '\a700';
}
.header {
width: 50%;
float: left;
min-height: 1px;
padding-right: 15px;
padding-left: 15px;
}
label {
display: inline-block;
max-width: 100%;
margin-bottom: 5px;
font-weight: 700;
}
</style>

```

```

@code{
private int? index { get; set; } = 0;
public string PositionValue { get; set; } = "left";
List<HeaderIcons> HeaderIconsData = new List<HeaderIcons>
{
new HeaderIcons() { Text= "Left", Value="left" },
new HeaderIcons() { Text= "Right" , Value="right"},
new HeaderIcons() { Text= "Top" , Value="top"},
new HeaderIcons() { Text= "Bottom", Value="bottom" }
};
public string Content0 = "Twitter is an online social networking service
that enables users to send and read short 140-character " +
"messages called 'tweets'. Registered users can read and post tweets, but
those who are unregistered can only read " +
"them. Users access Twitter through the website interface, SMS or mobile
device app Twitter Inc. is based in San " +
"Francisco and has more than 25 offices around the world. Twitter was
created in March 2006 by Jack Dorsey, " +
"Evan Williams, Biz Stone, and Noah Glass and launched in July 2006. The
service rapidly gained worldwide popularity, " +
"with more than 100 million users posting 340 million tweets a day in
2012.The service also handled 1.6 billion " +
"search queries per day.";
public string Content1 = "Facebook is an online social networking service
headquartered in Menlo Park, California. Its website was " +
"launched on February 4, 2004, by Mark Zuckerberg with his Harvard College
roommates and fellow students Eduardo " +
"Saverin, Andrew McCollum, Dustin Moskovitz and Chris Hughes.The founders
had initially limited the website " +
"membership to Harvard students, but later expanded it to colleges in the
Boston area, the Ivy League, and Stanford " +
"University. It gradually added support for students at various other
universities and later to high-school students.";
public string Content2 = "WhatsApp Messenger is a proprietary cross-platform
instant messaging client for smartphones that operates " +
"under a subscription business model. It uses the Internet to send text
messages, images, video, user location and " +
"audio media messages to other users using standard cellular mobile numbers.
As of February 2016, WhatsApp had a user " +
"base of up to one billion,[10] making it the most globally popular
messaging application. WhatsApp Inc., based in " +
"Mountain View, California, was acquired by Facebook Inc. on February 19,
2014, for approximately US$19.3 billion.";
public void
ChangeHeaderIcon(Syncfusion.Blazor.DropDowns.ChangeEventArgs<string,
HeaderIcons> args)
{
PositionValue = args.Value as String;
}
public class HeaderIcons
{
public string Text { get; set; }
public string Value { get; set; }
}
}

```



See Also

- [How to customize selected tab styles](#)

Content Render Modes in Blazor Tabs Component

In Blazor Tabs, the content of the Tabs can be rendered based on the scenario. The content rendering of tabs can be done by the following three different ways.

- [Dynamic rendering](#)
- [On Demand rendering or lazy loading](#)
- [On initial rendering](#)

Dynamic rendering

This mode is the default one in which the content of the selected tab alone will be loaded and available in DOM initially and it will be replaced with corresponding content if you select the tab dynamically. Since in this mode, the browser maintains the DOM with current active tab content alone, page loading performance is increased with rendering DOM. But the Tabs doesn't maintain its current state since every time tab loaded with fresh content.

In the following code example, there are two tabs. The first tab have a login page and second tab have Grid component. The second tab Grid component will be rendered in the DOM only when the login is completed. The second tab will be replaced the first tab in the DOM.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.Navigations
@using Syncfusion.Blazor.Grids
<SfTab LoadOn="ContentLoad.Dynamic" @bind-SelectedItem="@SelectedTab">
  <TabItems>
    <TabItem Disabled="@Disabled">
      <HeaderTemplate>Login</HeaderTemplate>
      <ContentTemplate>
        <div class="login-form">
          <div class='wrap'>
            <div id="heading">Sign in to view the Grid</div>
            <div id="input-container">
              <div class="e-float-input e-input-group">
                <input type="text" @bind-value="@UserName" required />
                <span class="e-float-line"></span>
              </div>
            </div>
          </div>
        </div>
      </ContentTemplate>
    </TabItem>
  </TabItems>
</SfTab>
```

```

<label class="e-float-text">Username</label>
</div>
<div class="e-float-input e-input-group">
<input type="password" @bind-value="@Password" required />
<span class="e-float-line"></span>
<label class="e-float-text">Password</label>
</div>
</div>
<div class="button-contain">
<SfButton @onclick="@OnClicked">Log in</SfButton>
</div>
</div>
</div>
</ContentTemplate>
</TabItem>
<TabItem Disabled="@DisableTab">
<HeaderTemplate>Grid</HeaderTemplate>
<ContentTemplate>
<SfGrid DataSource="@Orders">
<GridPageSettings></GridPageSettings>
<GridColumn>
<GridColumn Field=@nameof(Order.OrderID) HeaderText="Order ID"
TextAlign="TextAlign.Right" Width="120"></GridColumn>
<GridColumn Field=@nameof(Order.CustomerID) HeaderText="Customer Name"
Width="130"></GridColumn>
<GridColumn Field=@nameof(Order.OrderDate) HeaderText=" Order Date"
Format="yMd" Type="ColumnType.Date" TextAlign="TextAlign.Right"
Width="150"></GridColumn>
<GridColumn Field=@nameof(Order.Freight) HeaderText="Freight" Format="C2"
TextAlign="TextAlign.Right" Width="120"></GridColumn>
</GridColumn>
</SfGrid>
</ContentTemplate>
</TabItem>
</TabItems>
</SfTab>
@code {
private string UserName { get; set; } = "";
private string Password { get; set; } = "";
private Boolean DisableTab { get; set; } = true;
private Boolean Disabled { get; set; } = false;
private int SelectedTab { get; set; } = 0;
public List<Order> Orders { get; set; }
protected override void OnInitialized()
{
Orders = Enumerable.Range(1, 6).Select(x => new Order()
{
OrderID = 1000 + x,
CustomerID = (new string[] { "ALFKI", "ANANTR", "ANTON", "BLONP", "BOLID"
})[new Random().Next(5)],
Freight = 2.1 * x,
OrderDate = DateTime.Now.AddDays(-x),
}).ToList();
}
private void OnClicked()
{
if (this.UserName == "" && this.Password == "")

```

```
{
    Console.WriteLine("clicked");
}
else if (this.UserName == "")
{
    Console.WriteLine("Enter the username");
}
else if (this.Password == "")
{
    Console.WriteLine("Enter the password");
}
else if (this.UserName.Length < 4)
{
    Console.WriteLine("Username must be minimum 4 characters");
}
else
{
    this.UserName = "";
    this.Password = "";
    this.DisableTab = false;
    this.Disabled = true;
    this.SelectedTab = 1;
}
}

public class Order
{
    public int? OrderID { get; set; }
    public string CustomerID { get; set; }
    public DateTime? OrderDate { get; set; }
    public double? Freight { get; set; }
}
}
```

In this mode, if you want to maintain the state, you have to handle it in application end.

On Demand rendering or lazy loading

You can set this mode to our Tabs by setting `ContentLoad.Demand` to the property `LoadOn`. In this mode, the content of the selected tab alone will be loaded initially. The content of the selected tab will be rendered on selection. The content of the tabs which were loaded once will be maintained in the DOM. In this mode, since the selected tab content alone rendered on initial load and maintained the other tabs on selection in the DOM, state of the tabs like scroller position, form values etc., will be maintained.

In the following code example, Calendar and Scheduler have been rendered in first and second tab respectively. Initially, Scheduler is not available once the second tab is selected, scheduler will be rendered. Both the calendar and scheduler will be maintained in DOM, changing the date either in calendar or scheduler will change the date in other.

ASPX-CS

```
@using Syncfusion.Blazor.Calendars
@using Syncfusion.Blazor.Navigations
@using Syncfusion.Blazor.Schedule
<SfTab LoadOn="ContentLoad.Demand">
    <TabItems>
```



```
<TabItem>
<HeaderTemplate>Calendar</HeaderTemplate>
<ContentTemplate>
<SfCalendar TValue="DateTime" @bind-Value="@SelectedDate"></SfCalendar>
</ContentTemplate>
</TabItem>
<TabItem>
<HeaderTemplate>Scheduler</HeaderTemplate>
<ContentTemplate>
<SfSchedule TValue="AppointmentData" Height="450px" @bind-
SelectedDate="@SelectedDate">
<ScheduleViews>
<ScheduleView Option="View.Day"></ScheduleView>
</ScheduleViews>
</SfSchedule>
</ContentTemplate>
</TabItem>
</TabItems>
</SfTab>
@code {
private DateTime SelectedDate { get; set; } = new DateTime(2020, 1, 10);
public class AppointmentData
{
public int Id { get; set; }
public string Subject { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
}
}
```

On initial rendering

This mode can be set to the Tabs by setting `ContentLoad.Init` to the property `LoadOn`. In this mode, the content of all the tabs will be rendered on initial load and maintained in the DOM. You can use this mode, when you have less number of tabs and you need to maintain the state of tabs. In this mode, you can access the reference of components rendered in other tabs.

In the following example, all the three tabs are rendered in initial load itself and the data entered in the first tab will be maintained even when second or third tab is in active state.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
@using Syncfusion.Blazor.Inputs
@using Syncfusion.Blazor.Buttons
<SfTab LoadOn="ContentLoad.Init" @bind-SelectedItem="@SelectedTab">
<TabItems>
<TabItem>
<HeaderTemplate>Sign in</HeaderTemplate>
<ContentTemplate>
<div id="User details" style="padding:10px">
<form id="formId">
<div class="form-group">
<div class="e-float-input">
<SfTextBox Placeholder="Enter name" @bind-Value="UserName"></SfTextBox>
</div>
<div class="e-float-input">
```

```

<SfTextBox Placeholder="Email" @bind-Value="MailAddress"
Type="InputType.Email"></SfTextBox>
</div>
</div>
</form>
<div style="text-align: center">
<SfButton @onclick="@OnSignin">Sign in</SfButton>
<SfButton @onclick="@OnSkip">Skip</SfButton>
@if (EmptyField)
{
<div class="Error">* Please fill all fields</div>
}
</div>
</div>
</ContentTemplate>
</TabItem>
<TabItem>
<HeaderTemplate>Syncfusion Blazor</HeaderTemplate>
<ContentTemplate>
<p>You can check out our Syncfusion Blazor demo here -
https://blazor.syncfusion.com/demos/ </p>
<br />
<p>Also user guide will be avail here -
https://blazor.syncfusion.com/documentation/introduction/ </p>
</ContentTemplate>
</TabItem>
<TabItem>
<HeaderTemplate>Feedback</HeaderTemplate>
<ContentTemplate>
<div id="Feedback_Form" style="padding:10px">
<form id="formId">
<div class="form-group">
<div class="e-float-input">
<SfTextBox Placeholder="Enetr name" @bind-Value="UserName"></SfTextBox>
</div>
<div class="e-float-input">
<SfTextBox @bind-Value="MailAddress" Placeholder="Email"
Type="InputType.Email"></SfTextBox>
</div>
<div class="e-float-input">
<SfTextBox @bind-Value="Comments" Placeholder="Share your
comments"></SfTextBox>
</div>
</div>
</form>
<div style="text-align: center">
<SfButton @onclick="@OnSubmit">Submit</SfButton>
@if (EmptyField)
{
<div class="Error">* Please fill all fields</div>
}
</div>
</div>
</ContentTemplate>
</TabItem>
</TabItems>
</SfTab>

```

```
@code {
private string UserName;
private string MailAddress;
private string Comments;
private int SelectedTab = 0;
public Boolean EmptyField { get; set; } = false;
private void OnSignin()
{
if (this.UserName == null || this.MailAddress == null)
{
EmptyField = true;
}
else
{
EmptyField = false;
this.SelectedTab = 1;
}
}
private void OnSkip()
{
this.SelectedTab = 1;
}
public void OnSubmit()
{
if (this.UserName == null || this.MailAddress == null || this.Comments == null)
{
EmptyField = true;
}
else
{
EmptyField = false;
this.UserName = null;
this.MailAddress = null;
this.Comments = null;
}
}
}
```

Animations in Blazor Tabs Component

The tabs component supports custom animations for both previous and next actions from the provided animation option of **Animation** library. The animation property also allows to set **Easing**, **Duration** and various other effects.

Default animation is given as **SlideLeftIn** for previous tab animation and **SlideRightIn** for next tab animation. Disable the animation by setting the animation **Effect** as **None**. Also, please use the following CSS to disable indicator animation for animation **Effect** as **None**.

CSS

```
.e-tab .e-tab-header:not(.e-vertical) .e-indicator, .e-tab .e-tab-header.e-vertical .e-indicator {
transition: none;
}
```

The sample demonstrates some types of animation that suits our Tabs. Check all the animation effects [here](#).

ASPX-CS

```
@using Syncfusion.Blazor
@using Syncfusion.Blazor.Navigations
@using Syncfusion.Blazor.DropDowns
<div class="row">
<div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
<label> Previous Animation </label>
</div>
<div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
<SfDropDownList TValue="AnimationEffect" DataSource="@AnimationData"
TItem="Effect" @bind-Value="PreviousEffect">
<DropDownListEvents ValueChange="PreviousChange" TValue="AnimationEffect"
TItem="Effect"></DropDownListEvents>
<DropDownListFieldSettings Value="ID"
Text="Text"></DropDownListFieldSettings>
</SfDropDownList>
</div>
</div>
<div class="row">
<div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
<label> Next Animation </label>
</div>
<div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
<SfDropDownList TValue="AnimationEffect" DataSource="@AnimationData"
TItem="Effect" @bind-Value="NextEffect">
<DropDownListEvents ValueChange="NextChange" TValue="AnimationEffect"
TItem="Effect"></DropDownListEvents>
<DropDownListFieldSettings Value="ID"
Text="Text"></DropDownListFieldSettings>
</SfDropDownList>
</div>
</div>
<br />
<SfTab Width="600px">
<TabAnimationSettings>
<TabAnimationPrevious Effect="@PreviousEffect"></TabAnimationPrevious>
<TabAnimationNext Effect="@NextEffect"></TabAnimationNext>
</TabAnimationSettings>
<TabItems>
<TabItem Content="@Content1">
<ChildContent>
<TabHeader Text="Twitter"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="@Content2">
<ChildContent>
<TabHeader Text="Facebook"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="@Content3">
<ChildContent>
<TabHeader Text="WhatsApp"></TabHeader>
</ChildContent>
</TabItem>
</TabItems>
</SfTab>
```

```

</TabItem>
</TabItems>
</SfTab>
@code{
public AnimationEffect PreviousEffect = AnimationEffect.SlideLeftIn;
public AnimationEffect NextEffect = AnimationEffect.SlideRightIn;
public string Content1 = "Twitter is an online social networking service
that enables users to send and read short 140-character " +
"messages called 'tweets'. Registered users can read and post tweets, but
those who are unregistered can only read " +
"them. Users access Twitter through the website interface, SMS or mobile
device app Twitter Inc. is based in San " +
"Francisco and has more than 25 offices around the world. Twitter was
created in March 2006 by Jack Dorsey, " +
"Evan Williams, Biz Stone, and Noah Glass and launched in July 2006. The
service rapidly gained worldwide popularity, " +
"with more than 100 million users posting 340 million tweets a day in
2012.The service also handled 1.6 billion " +
"search queries per day.";
public string Content2 = "Facebook is an online social networking service
headquartered in Menlo Park, California. Its website was " +
"launched on February 4, 2004, by Mark Zuckerberg with his Harvard College
roommates and fellow students Eduardo " +
"Saverin, Andrew McCollum, Dustin Moskovitz and Chris Hughes.The founders
had initially limited the website " +
"membership to Harvard students, but later expanded it to colleges in the
Boston area, the Ivy League, and Stanford " +
"University. It gradually added support for students at various other
universities and later to high-school students.";
public string Content3 = "WhatsApp Messenger is a proprietary cross-platform
instant messaging client for smartphones that operates " +
"under a subscription business model. It uses the Internet to send Text
messages, images, vIDeo, user location and " +
"audio media messages to other users using standard cellular mobile numbers.
As of February 2016, WhatsApp had a user " +
"base of up to one billion,[10] making it the most globally popular
messaging application. WhatsApp Inc., based in " +
"Mountain View, California, was acquired by Facebook Inc. on February 19,
2014, for approximately US$19.3 billion.";
List<Effect> AnimationData = new List<Effect> {
new Effect() { ID= AnimationEffect.SlideLeftIn, Text= "SlidLeftIn" },
new Effect() { ID= AnimationEffect.SlideRightIn, Text= "SlideRightIn" },
new Effect() { ID= AnimationEffect.FadeIn, Text= "FadeIn" },
new Effect() { ID= AnimationEffect.FadeOut, Text= "FadeOut" },
new Effect() { ID= AnimationEffect.FadeZoomIn, Text= "FadeZoomIn" },
new Effect() { ID= AnimationEffect.FadeZoomOut, Text= "FadeZoomOut" },
new Effect() { ID= AnimationEffect.ZoomIn, Text= "ZoomIn" },
new Effect() { ID= AnimationEffect.ZoomOut, Text= "ZoomOut" },
new Effect() { ID= AnimationEffect.None, Text= "None" }
};
public void
PreviousChange(Syncfusion.Blazor.DropDowns.ChangeEventArgs<AnimationEffect,
Effect> args)
{
this.PreviousEffect = args.Value;
}
}

```

```
public void
NextChange(Syncfusion.Blazor.DropDowns.ChangeEventArgs<AnimationEffect,
Effect> args)
{
    this.NextEffect = args.Value;
}
public class Effect
{
    public AnimationEffect ID { get; set; }
    public string Text { get; set; }
}
}
```



Localization in Blazor Tabs Component

Localization library allows to localize the default text content of Tab. In Tab, the close button's tooltip text alone will be localize based on culture.

| Locale key | en-US (default) |

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

| closeButtonTitle | Close |

Blazor Server Side

The static local texts in the Tabs component can be changed to other culture by referring the Resource file. Refer more details about localization [here](#). By default, Tabs is set to follow the English culture ('en-US'). The following steps explain how to render the Tabs in German culture ('de-DE').

- Open the **Startup.cs** file and add the below configuration in the **ConfigureServices** function as follows.

CSHARP

```
using Syncfusion.Blazor;
using System.Globalization;
using Microsoft.AspNetCore.Localization;
namespace TabLocalization
{
```

```

public class Startup
{
    ....
    ....
    public void ConfigureServices(IServiceCollection services)
    {
        ....
        ....
        services.AddSyncfusionBlazor();
        services.AddLocalization(options => options.ResourcesPath = "Resources");
        services.Configure<RequestLocalizationOptions>(options =>
        {
            // define the list of cultures your app will support
            var supportedCultures = new List<CultureInfo>()
            {
                new CultureInfo("de-DE")
            };
            // set the default culture
            options.DefaultRequestCulture = new RequestCulture("de-DE");
            options.SupportedCultures = supportedCultures;
            options.SupportedUICultures = supportedCultures;
            options.RequestCultureProviders = new List<IRequestCultureProvider>() {
                new QueryStringRequestCultureProvider() // Here, You can also use other
                localization provider
            };
        });
        services.AddSingleton(typeof(ISyncfusionStringLocalizer),
            typeof(SampleLocalizer));
    }
}

```

Add [UseRequestLocalization\(\)](#) middle-ware in Configure method in **Startup.cs** file to get browser Culture Information.

- Then, write a **class** by inheriting **ISyncfusionStringLocalizer** interface and override the **ResourceManager** property to get the resource file details from the application end.

CSHARP

```

using Syncfusion.Blazor;
namespace TabLocalization
{
    public class SampleLocalizer : ISyncfusionStringLocalizer
    {
        public string GetText(string key)
        {
            return this.ResourceManager.GetString(key);
        }
        public System.Resources.ResourceManager ResourceManager
        {
            get
            {
                return TabLocalization.Resources.SyncfusionBlazorLocale.ResourceManager;
            }
        }
    }
}

```

```
}
}
}
```

- Add **.resx** file to [Resource](#) folder and that file contains the key value pair of locale content in the following format.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<SfTab ShowCloseButton="true">
<TabItems>
<TabItem Content="Twitter is an online social networking service that
enables users to send and read short 140-character messages called
tweets. Registered users can read and post tweets, but those who are
unregistered can only read them. Users access Twitter through the website
interface, SMS or mobile device app. Twitter Inc. is based in San Francisco
and has more than 25 offices around the world. Twitter was created in March
2006 by Jack Dorsey, Evan Williams, Biz Stone, and Noah Glass and launched in
July 2006. The service rapidly gained worldwide popularity, with more than
100 million users posting 340 million tweets a day in 2012. The service also
handled 1.6 billion search queries per day.">
<ChildContent>
<TabHeader Text="Twitter">
</TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="Facebook is an online social networking service
headquartered in Menlo Park, California. Its website was launched on February
4, 2004, by Mark Zuckerberg with his Harvard College roommates and fellow
students Eduardo Saverin, Andrew McCollum, Dustin Moskovitz and Chris
Hughes.">
<ChildContent>
<TabHeader Text="Facebook">
</TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="WhatsApp Messenger is a proprietary cross-platform instant
messaging client for smartphones that operates under a subscription business
model. It uses the Internet to send text messages, images, video, user
location and audio media messages to other users using standard cellular
mobile numbers. As of February 2016, WhatsApp had a userbase of up to one
billion, [10] making it the most globally popular messaging
application. WhatsApp Inc., based in Mountain View, California, was acquired
by Facebook Inc. on February 19, 2014, for approximately US$19.3 billion.">
<ChildContent>
<TabHeader Text="Whatsapp">
</TabHeader>
</ChildContent>
</TabItem>
</TabItems>
</SfTab>
```


Blazor Web Assembly

The following steps explain you to set **de-DE** culture for Tabs in web assembly application.

1. Add the Localization service configuration in the **~/Program.cs** file.

CSHARP

```
using Syncfusion.Blazor;
using Microsoft.JSInterop;
using System.Globalization;
using TabLocalization.Shared;
public class Program
{
    public static async Task Main(string[] args)
    {
        var builder = WebAssemblyHostBuilder.CreateDefault(args);
        .....
        .....
        builder.Services.AddSyncfusionBlazor();
        #region Localization
        // Register the Syncfusion locale service to customize the SyncfusionBlazor
        // component locale culture
        builder.Services.AddSingleton(typeof(ISyncfusionStringLocalizer),
        typeof(SyncfusionLocalizer));
        // Set the default culture of the application
        CultureInfo.DefaultThreadCurrentCulture = new CultureInfo("de-DE");
        CultureInfo.DefaultThreadCurrentUICulture = new CultureInfo("de-DE");
        #endregion
        await builder.Build().RunAsync();
    }
}
```

2. Create **~/Shared/SyncfusionLocalizer.cs** file and implement **ISyncfusionStringLocalizer** to the class. This acts as a middleware to connect the Syncfusion Blazor UI components and resource files.

Map the **SfResources.ResourceManager** to this interface **ResourceManager**.

CSHARP

```
using Syncfusion.Blazor;
public class SyncfusionLocalizer : ISyncfusionStringLocalizer
{
    // To get the locale key from mapped resources file
    public string GetText(string key)
    {
        return this.ResourceManager.GetString(key);
    }
    // To access the resource file and get the exact value for locale key
    public System.Resources.ResourceManager ResourceManager {
        get
        {
            return TabLocalization.Resources.SfResources.ResourceManager;
        }
    }
}
```

```
}
}
```

3. Add the resource files in the `~/Resources` folder. The locale resource files for different cultures are available in this [GitHub](#) repository. You can get any culture resource file from there and utilize it in your application.

After adding the resource file in the application we need to generate the designer class for the resources. To generate the designer class, open the default `resx` file in Visual Studio, and set its `Access Modifier` to `Public`. This will create an entry in your `.csproj` file similar to the following.

XML

```
<ItemGroup>
  <Compile Update="Resources\SfResources.Designer.cs">
    <DesignTime>True</DesignTime>
    <AutoGen>True</AutoGen>
    <DependentUpon>SfResources.resx</DependentUpon>
  </Compile>
</ItemGroup>
<ItemGroup>
  <EmbeddedResource Update="Resources\SfResources.resx">
    <Generator>PublicResXFileCodeGenerator</Generator>
    <LastGenOutput>SfResources.Designer.cs</LastGenOutput>
  </EmbeddedResource>
</ItemGroup>
```

4. Add the custom JavaScript interop function to get or set the culture in `~/wwwroot/index.html`.

HTML

```
<body>
  ....
  ....
  <script src="_framework/blazor.webassembly.js"></script>
  <script>
    window.cultureInfo = {
      get: () => window.localStorage['BlazorCulture'],
      set: (value) => window.localStorage['BlazorCulture'] = value
    };
  </script>
</body>
```

5. Finally, add the Tabs in razor page as in the following code example.

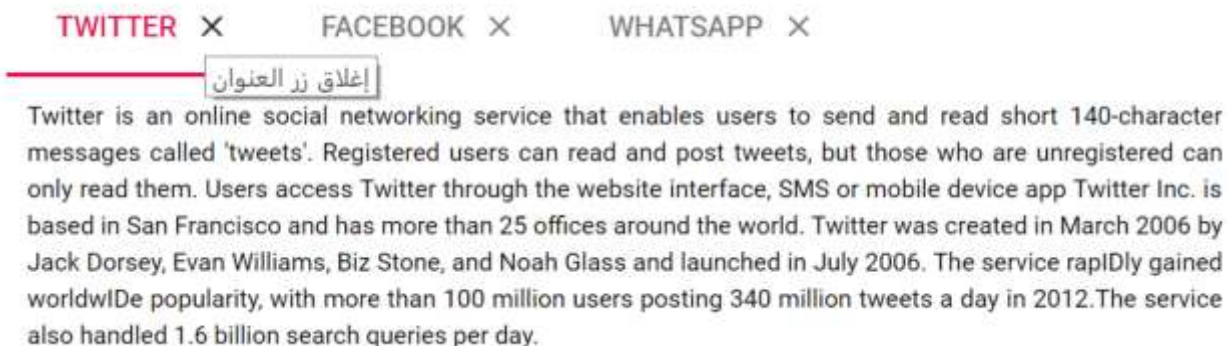
ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<SfTab ShowCloseButton="true">
```

```

<TabItems>
<TabItem Content="Twitter is an online social networking service that
enables users to send and read short 140-character messages called
tweets. Registered users can read and post tweets, but those who are
unregistered can only read them. Users access Twitter through the website
interface, SMS or mobile device app Twitter Inc. is based in San Francisco
and has more than 25 offices around the world. Twitter was created in March
2006 by Jack Dorsey, Evan Williams, Biz Stone, and Noah Glass and launched in
July 2006. The service rapidly gained worldwide popularity, with more than
100 million users posting 340 million tweets a day in 2012. The service also
handled 1.6 billion search queries per day.">
<ChildContent>
<TabHeader Text="Twitter">
</TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="Facebook is an online social networking service
headquartered in Menlo Park, California. Its website was launched on February
4, 2004, by Mark Zuckerberg with his Harvard College roommates and fellow
students Eduardo Saverin, Andrew McCollum, Dustin Moskovitz and Chris
Hughes.">
<ChildContent>
<TabHeader Text="Facebook">
</TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="WhatsApp Messenger is a proprietary cross-platform instant
messaging client for smartphones that operates under a subscription business
model. It uses the Internet to send text messages, images, video, user
location and audio media messages to other users using standard cellular
mobile numbers. As of February 2016, WhatsApp had a userbase of up to one
billion, [10] making it the most globally popular messaging
application. WhatsApp Inc., based in Mountain View, California, was acquired
by Facebook Inc. on February 19, 2014, for approximately US$19.3 billion.">
<ChildContent>
<TabHeader Text="Whatsapp">
</TabHeader>
</ChildContent>
</TabItem>
</TabItems>
</SfTab>

```



Orientation in Blazor Tabs Component

This section explains about modifying the position and modes of Tab header.

It allows placing the header section inside the Tabs component at different positions by using the [HeaderPlacement](#) property. The available positions are as follows:

- **Top:** Tab header items can be arranged horizontally, and their content can be placed after the header.
- **Bottom:** Tab header items can be arranged horizontally, and their content can be placed before the header.
- **Left:** Tab header items can be arranged vertically, and their content can be placed after the header.
- **Right:** Tab header items can be arranged vertically, and their content can be placed before the header.

It is also adaptable to the available space when the tab items exceed the view space. The modes can be customized by using [OverflowMode](#) property. The available modes are as follows:

- Scrollable
- Popup

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
@using Syncfusion.Blazor.DropDowns
<SfTab ShowCloseButton="true" HeaderPlacement="@Header" OverflowMode="@Mode"
Width="500px" Height="250px">
  <TabItems>
    <TabItem Content="@Content0">
      <ChildContent>
        <TabHeader Text="HTML"></TabHeader>
      </ChildContent>
    </TabItem>
    <TabItem Content="@Content1">
      <ChildContent>
        <TabHeader Text="C Sharp (C#)"></TabHeader>
      </ChildContent>
    </TabItem>
    <TabItem Content="@Content2">
      <ChildContent>
        <TabHeader Text="Java"></TabHeader>
      </ChildContent>
    </TabItem>
    <TabItem Content="@Content3">
      <ChildContent>
        <TabHeader Text="VB.Net"></TabHeader>
      </ChildContent>
    </TabItem>
    <TabItem Content="@Content4">
      <ChildContent>
        <TabHeader Text="Xamarin"></TabHeader>
      </ChildContent>
    </TabItem>
  </TabItems>
</SfTab>
```

```

<TabItem Content="@Content5">
  <ChildContent>
    <TabHeader Text="ASP.NET"></TabHeader>
  </ChildContent>
</TabItem>
<TabItem Content="@Content6">
  <ChildContent>
    <TabHeader Text="ASP.NET MVC"></TabHeader>
  </ChildContent>
</TabItem>
<TabItem Content="@Content7">
  <ChildContent>
    <TabHeader Text="JavaScript"></TabHeader>
  </ChildContent>
</TabItem>
</TabItems>
</SfTab>
<div class="col-md-4 property-section">
  <div class="property-panel-section">
    <div class="property-panel-header">Properties</div>
    <div class="property-panel-content">
      <table id="property" title="Properties" style="width: 100%" class="property-
panel-table">
        <tbody>
          <tr>
            <td style="width: 50%;">
              <div>Header Placement</div>
            </td>
            <td style="width: 50%;">
              <div>
                <SfDropDownList DataSource="@OrientationData" @bind-Value="@HeaderValue"
TValue="string" TItem="DropdownFields">
                  <DropDownListEvents ValueChange="OnHeaderPositionChange" TValue="string"
TItem="DropdownFields"></DropDownListEvents>
                  <DropDownListFieldSettings Value="Value"
Text="Text"></DropDownListFieldSettings>
                </SfDropDownList>
              </div>
            </td>
          </tr>
          <tr>
            <td style="width: 50%;">
              <div>Header Styles</div>
            </td>
            <td style="width: 50%;">
              <div>
                <SfDropDownList TValue="string" TItem="DropdownFields"
DataSource="@TabModeData" @bind-Value="@ModeValue">
                  <DropDownListEvents ValueChange="OnChangeMode" TValue="string"
TItem="DropdownFields"></DropDownListEvents>
                  <DropDownListFieldSettings Value="Value"
Text="Text"></DropDownListFieldSettings>
                </SfDropDownList>
              </div>
            </td>
          </tr>
        </tbody>
      </table>
    </div>
  </div>

```

```

</table>
</div>
</div>
</div>
@code {
public string HeaderValue { get; set; } = "Top";
public HeaderPosition Header { get; set; } = HeaderPosition.Top;
public OverflowMode Mode { get; set; } = OverflowMode.Scrollable;
public string ModeValue { get; set; } = "Scrollable";
public string Content0 = "HyperText Markup Language, commonly referred to as HTML, is the standard markup language used to create webpages. Along with CSS, and JavaScript, HTML is a cornerstone technology, used by most websites to create visuallyengaging web pages, user interfaces for web applications, and user interfaces for many mobile applications.[1] Webbrowsers can read HTML files and render them into visible or audible web pages. HTML describes the structure of awebsite semantically along with cues for presentation, making it a markup language, rather than a programming language.";
public string Content1 = "C# is intended to be a simple, modern, general-purpose, object-oriented programming language. Its developmentteam is led by Anders Hejlsberg. The most recent version is C# 5.0, which was released on August 15, 2012.";
public string Content2 = "Java is a set of computer software and specifications developed by Sun Microsystems, later acquired by OracleCorporation, that provides a system for developing application software and deploying it in a cross - platform computingenvironment. Java is used in a wide variety of computing platforms from embedded devices and mobile phones toenterprise servers and supercomputers. While less common, Java applets run in secure, sandboxed environments toprovide many features of native applications and can be embedded in HTML pages.";
public string Content3 = "The command-line compiler, VBC.EXE, is installed as part of the freeware .NET Framework SDK. Mono alsoincludes a command - line VB.NET compiler. The most recent version is VB 2012, which was released on August 15, 2012.";
public string Content4 = "Xamarin is a San Francisco, California based software company created in May 2011[3] by the engineers that created Mono,[4] Mono for Android and MonoTouch that are cross-platform implementations of the Common Language Infrastructure (CLI) and Common LanguageSpecifications (often called Microsoft .NET). With a C#-shared codebase, developers can use Xamarin tools to write native Android,iOS, and Windows apps with native user interfaces and share code across multiple platforms.[5] Xamarin has over 1 million developersin more than 120 countries around the World as of May 2015.";
public string Content5 = "ASP.NET is an open-source server-side web application framework designed for web development to producedynamic web pages. It was developed by Microsoft to allow programmers to build dynamic web sites, web applicationsand web services. It was first released in January 2002 with version 1.0 of the .NET Framework, and is the successor to Microsoft Active Server Pages (ASP) technology. ASP.NET is built on the Common Language Runtime (CLR), allowingprogrammers to write ASP.NET code using any supported .NET language. The ASP.NET SOAP extension framework allowsASP.NET components to process SOAP messages.";
public string Content6 = "The ASP.NET MVC is a web application framework developed by Microsoft, which implements the model-view-controller (MVC) pattern. It is open - source software, apart from the ASP.NET Web Forms component which isproprietary. In the later versions of ASP.NET, ASP.NET MVC, ASP.NET Web API, and ASP.NET Web Pages (a platform usingonly Razor pages) will merge into a unified MVC 6. The project is called ASP.NET Next.";

```

```

public string Content7 = "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.[5] More recently, however, it has become common in both game development and the creation of desktop applications.";
List<DropdownFields> OrientationData = new List<DropdownFields>()
{
    new DropdownFields() { Value= "Top", Text= "Top" },
    new DropdownFields() { Value= "Bottom", Text= "Bottom" },
    new DropdownFields() { Value= "Left", Text= "Left" },
    new DropdownFields() { Value= "Right", Text= "Right" }
};
List<DropdownFields> TabModeData = new List<DropdownFields>()
{
    new DropdownFields() { Value= "Scrollable", Text= "Scrollable" },
    new DropdownFields() { Value= "Popup", Text= "Popup" }
};
public void
OnHeaderPositionChange(Syncfusion.Blazor.DropDowns.ChangeEventArgs<string,
DropdownFields> args)
{
    Header = (HeaderPosition)Enum.Parse(typeof(HeaderPosition), (args.Value as string));
}
public void OnChangeMode(Syncfusion.Blazor.DropDowns.ChangeEventArgs<string,
DropdownFields> args)
{
    Mode = (OverflowMode)Enum.Parse(typeof(OverflowMode), (args.Value as string));
}
public class DropdownFields
{
    public string Value { get; set; }
    public string Text { get; set; }
}
<style>
.e-content .e-item {
font-size: 12px;
padding: 10px;
text-align: justify;
}
</style>

```

< **HTML** X C SHARP(C#) X JAVA X VB.NET X >

HyperText Markup Language, commonly referred to as HTML, is the standard markup language used to create webpages. Along with CSS, and JavaScript, HTML is a cornerstone technology, used by most websites to create visually engaging web pages, user interfaces for web applications, and user interfaces for many mobile applications.[1] Web browsers can read HTML files and render them into visible or audible web pages. HTML describes the structure of a website semantically along with cues for presentation, making it a markup language, rather than a programming language.

Properties

Header Placement	Top	▼
Header Styles	Scrollable	▼

Drag and Drop in Blazor Tabs Component

The Tab component allows to drag and drop any item by setting `AllowDragAndDrop` to `true`. Items can be reordered to any place by dragging and dropping them onto the desired location.

- If you need to prevent dragging action for a particular item, the `OnDragStart` event can be used which will trigger when the item drag is started. If you need to prevent dropping action for a particular item, the `Dragged` event can be used which will trigger when the drag action is stopped.
- The `DragArea` defines the area in which the draggable element movement will be occurring. Outside that area will be restricted for the draggable element movement.
- The `OnDragStart` event will be triggered before dragging the item from Tab.
- The `Dragged` event will be triggered when the Tab item is dropped on the target element successfully.

In the following sample, the `AllowDragAndDrop` property is enabled.

External drag and drop is not possible in blazor Tabs.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<SfTab CssClass="drag-drop-tab" AllowDragAndDrop="true">
  <TabItems>
    <TabItem Content="India officially the Republic of India, is a country in
    South Asia. It is the seventh-largest country by area, the second-most
    populous country with over 1.2 billion people, and the most populous
```



```

democracy in the world. Bounded by the Indian Ocean on the south, the
Arabian Sea on the south-west, and the Bay of Bengal on the south-east, it
shares land borders with Pakistan to the west;China, Nepal, and Bhutan to
the north-east; and Burma and Bangladesh to the east. In the Indian Ocean,
India is in the vicinity of Sri Lanka and the Maldives; in addition, India
Andaman and Nicobar Islands share a maritime border with Thailand and
Indonesia.">
<ChildContent>
<TabHeader Text="India"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="Australia, officially the Commonwealth of Australia, is a
country comprising the mainland of the Australian continent, the island of
Tasmania and numerous smaller islands. It is the world sixth-largest country
by total area. Neighboring countries include Indonesia, East Timor and Papua
New Guinea to the north; the Solomon Islands, Vanuatu and New Caledonia to
the north-east; and New Zealand to the south-east.">
<ChildContent>
<TabHeader Text="Australia"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="The United States of America (USA or U.S.A.), commonly
called the United States (US or U.S.) and America, is a federal republic
consisting of fifty states and a federal district. The 48 contiguous states
and the federal district of Washington, D.C. are in central North America
between Canada and Mexico. The state of Alaska is west of Canada and east of
Russia across the Bering Strait, and the state of Hawaii is in the mid-North
Pacific. The country also has five populated and nine unpopulated
territories in the Pacific and the Caribbean.">
<ChildContent>
<TabHeader Text="USA"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="France, officially the French Republic is a sovereign
state comprising territory in western Europe and several overseas regions
and territories. The European part of France, called Metropolitan France,
extends from the Mediterranean Sea to the English Channel and the North Sea,
and from the Rhine to the Atlantic Ocean; France covers 640,679 square kilo
metres and as of August 2015 has a population of 67 million, counting all
the overseas departments and territories.">
<ChildContent>
<TabHeader Text="France"></TabHeader>
</ChildContent>
</TabItem>
</TabItems>
</SfTab>
<style>
.drag-drop-tab .e-content .e-item {
font-size: 12px;
padding: 10px;
text-align: justify;
}
</style>

```



Accessibility in Blazor Tabs Component

ARIA attributes

Tabs component is designed by considering [WAI-ARIA](#) standard. Tab is supported with ARIA Accessibility which is accessible by on-screen readers, and other assistive technology devices.

The following list of attributes are added in the Tab.

| Roles and Attributes | Functionalities |

| --- | --- |

| tablist | This is set to role attribute in the Tab element that describes actual role of the element. |

|tabpanel | This is set to role attribute for the Tab content that describes the role for viewing the active content. |

| presentation | This is set to role attribute for nested elements in the Tab. |

| aria-orientation | It indicates the Tab header orientation. Default value of this attribute is horizontal. |

| aria-activedescendant | It indicates the current active child of the Tab component. |

| aria-haspopup | It indicates the popup mode in the Tab. The default value of this attribute is false. If popup mode is enabled, the attribute value is set to true. |

| aria-disabled | It indicates the disabled state of the Tab. |

| aria-selected | It indicates the selection state for Tab items. Active Tab is set to true for this attribute. |

| aria-hidden | It indicates the hidden element of the Tab. |

| aria-controls | It indicates the associated tabpanel for the header. |

| aria-labelledby | It indicates the associated Tab header for the content. |

Keyboard interaction

By default, keyboard navigation is enabled. This component implements keyboard navigation support by following the WAI-ARIA practices. Once focused on the active Tab element, use the following key combination for interacting with the Tab.

Key	Description
Left	Moves focus to the previous Tab. If focus is on the first Tab, the focus will not move to any Tab.

Key	Description
Left	Moves focus to the previous Tab. If focus is on the first Tab, the focus will not move to any Tab.

Key	Description
Left	Moves focus to the previous Tab. If focus is on the first Tab, the focus will not move to any Tab.

| Right | Moves focus to the next Tab. If focus is on the last Tab element, the focus will not move to any Tab. |

| Enter or Space | Selects the Tab if it is not selected. Opens the popup dropdown icon if it is focused. Select the Tab item as active when popup item is focused. |

| Esc(Escape) | Closes the popup if popup is in opened state. |

| Down or Up | When the popup is open and focused, it will move to previous or next Tab items of the popup in the vertical direction. |

| Home | Moves focus to the first Tab. |

| End | Moves focus to the last Tab. |

| Shift + F10 | If popup mode is enabled, it opens the popup when the Tab is focused. |

| Delete | Deletes the Tab, if close button is enabled in Tab header. |

Style and Appearance in Blazor Tabs Component

The following content provides the exact CSS structure that can be used to modify the control's appearance based on user preference.

Customizing Tab

Use the following CSS to customize the Tab.

CSS

```
.e-tab {  
border: 5px solid rgb(173, 255, 47);  
}
```

Customizing the Tab items

Use the following CSS to customize the header items of Tab.

CSS

```
.e-tab .e-tab-header .e-toolbar-items {  
background: #9faed8;  
border: 2px solid blue;  
}
```

Use the following CSS to customize the content items of Tab.

CSS

```
.e-tab .e-content .e-item {  
color: #a78515;  
font-size: 14px;  
}
```

Customizing Tab's header

Use the following CSS to customize the header of Tab control.

CSS

```
.e-tab .e-tab-header {  
background: #badfba !important;  
}
```

Customizing Tab's header icon

Use the following CSS to customize the header item icon of Tab control.

CSS

```
.e-tab .e-tab-header .e-toolbar-item .e-tab-icon {  
color: #badfba !important;  
}
```

Customizing Tab's content

Use the following CSS to customize the content of Tab control.

CSS

```
.e-tab .e-content {  
background: #d1f6d1 !important;  
}
```

Customizing the hover state of Tab control

Use the following CSS to customize the tab item when hovering.

CSS

```
.e-tab .e-tab-header .e-toolbar-item .e-tab-wrap:hover {  
background: #d1f6d1 !important;  
}
```

Use the following CSS to customize the tab item popup icon when hovering.

CSS

```
.e-tab .e-tab-header .e-hor-nav .e-popup-up-icon:hover,  
.e-tab .e-tab-header .e-hor-nav .e-popup-down-icon:hover {  
background: #d1f6d1 !important;  
}
```

Customizing selected item of Tab control

Use the following CSS to customize the selected tab item.

CSS

```
.e-tab .e-tab-header .e-toolbar-item.e-active {  
background: #d1f4d1;  
}
```

Use the following CSS to customize the selected tab item text and icon.

CSS

```
.e-tab .e-tab-header .e-toolbar-item.e-active .e-tab-text,
```

```
.e-tab .e-tab-header .e-toolbar-item.e-active .e-tab-icon {
color: green !important;
}
```

How To

Add/Remove Tab items in Blazor Tabs Component

Tabs provides a support to add or remove the specified tab item using following ways.

- Using conditional rendering.
- Using public methods.

Using conditional rendering

Tab items can be added or removed dynamically by iteration of tab items using conditional **foreach** loop.

In the following demo, initially there are three tabs as the **TabItems** has three items. On **Add Item** button click, new item is added to the **TabItems** results in adding fourth tab to the Tabs component. On clicking the **Remove Item**, the first item of **TabItems** has been removed which results in removing first tab of our Tabs component.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
@using Syncfusion.Blazor.Buttons
<SfButton @onclick="AddItemClick" Content="Add Item"></SfButton>
<SfButton @onclick="RemoveItemClick" Content="Remove Item"></SfButton>
<br />
<br />
<SfTab>
<TabItems>
@foreach (TabData Item in TabItems)
{
<TabItem>
<HeaderTemplate>
<div>@(Item.Header)</div>
</HeaderTemplate>
<ContentTemplate>
<div>@(Item.Content)</div>
</ContentTemplate>
</TabItem>
}
</TabItems>
</SfTab>
@code {
List<TabData> TabItems = new List<TabData>()
{
new TabData
{
Header = "ASP.NET",
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 TabData
{
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 TabData
{
Header = "ASP.NET Razor",
Content = "Razor is an ASP.NET programming syntax used to create dynamic web
pages with the C# or Visual Basic .NET programming languages. Razor was in
development in June 2010 and was released for Microsoft Visual Studio 2010
in January 2011. Razor is a simple-syntax view engine and was released as
part of MVC 3 and the WebMatrix tool set. Side Code content"
}
};
public class TabData
{
public string Header { get; set; }
public string Content { get; set; }
}
void AddItemClick()
{
TabItems.Add(new TabData
{
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."
});
}
void RemoveItemClick()
{
if (TabItems.Count > 0)
{
TabItems.RemoveAt(0);
}
}
}

```

ADD ITEM REMOVE ITEM

ASP.NET ASP.NET MVC ASP.NET RAZOR

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.

Using public methods

Tabs can be added dynamically by passing list of items and index value to the `AddTab` method. The tab items can be removed by passing the specified item through `RemoveTab` method. Tab items can also be removed by clicking the close icon which appears on the tab header on setting true to `ShowCloseButton` property.

In the following demo, a tab item can be added as first tab and removed as the last tab item by clicking the **Add Tab** and **Remove Last Tab** buttons respectively.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
@using Syncfusion.Blazor.Buttons
<SfButton OnClick="AddItemClick" Content="Add Tab"></SfButton>
<SfButton OnClick="RemoveItemClick" Content="Remove Last Tab"></SfButton>
<br/>
<SfTab @ref="Tab" ShowCloseButton="true">
  <TabItems>
    <TabItem Content="New York City comprises 5 boroughs sitting where the
    Hudson River meets the Atlantic Ocean. At its core is Manhattan, a densely
    populated borough that's among the world's major commercial, financial and
    cultural centers. Its iconic sites include skyscrapers such as the Empire
    State Building and sprawling Central Park. Broadway theater is staged in
    neon-lit Times Square.">
      <ChildContent>
        <TabHeader Text="New York"></TabHeader>
      </ChildContent>
    </TabItem>
    <TabItem Content="Los Angeles is a sprawling Southern California city and
    the center of the nation's film and television industry. Near its iconic
    Hollywood sign, studios such as Paramount Pictures, Universal and Warner
    Brothers offer behind-the-scenes tours. On Hollywood Boulevard, TCL Chinese
    Theatre displays celebrities' hand- and footprints, the Walk of Fame honors
    thousands of luminaries and vendors sell maps to stars' homes.">
      <ChildContent>
        <TabHeader Text="Los Angeles"></TabHeader>
      </ChildContent>
    </TabItem>
    <TabItem Content="Chicago, on Lake Michigan in Illinois, is among the
    largest cities in the U.S. Famed for its bold architecture, it has a skyline
    punctuated by skyscrapers such as the iconic John Hancock Center, 1,451-ft.
    Willis Tower (formerly the Sears Tower) and the neo-Gothic Tribune Tower.
    The city is also renowned for its museums, including the Art Institute of
    Chicago with its noted Impressionist and Post-Impressionist works.">
      <ChildContent>
        <TabHeader Text="Chicago"></TabHeader>
      </ChildContent>
    </TabItem>
  </TabItems>
</SfTab>
@code{
  SfTab Tab;
  List<TabItem> TabData;
  public void AddItemClick()
  {
    TabData = new List<TabItem>()
```

```
{
new TabItem() { Header = new TabHeader() { Text = "Sydney" }, Content =
"Sydney, capital of New South Wales and one of Australia largest cities, is
best known for its harbourfront Sydney Opera House, with a distinctive sail-
like design. Massive Darling Harbour and the smaller Circular Quay port are
hubs of waterside life, with the arched Harbour Bridge and esteemed Royal
Botanic Garden nearby. Sydney Tower's outdoor platform, the Skywalk, offers
360-degree views of the city and suburbs." }
};
Tab.AddTab(TabData, 0);
}
public void RemoveItemClick()
{
Tab.RemoveTab(Tab.Items.Count - 1);
}
}
<style>
.e-content .e-item {
font-size: 12px;
margin: 10px;
text-align: justify;
}
</style>
```



Show/Hide Tab item in Blazor Tabs Component

The [Visible](#) property of the Tab item is used to show or hide the item by setting true or false value to the property. In the following demo, specified tab item is show or hide dynamically when the **Show/Hide Item** button is clicked.

ASPX-CS

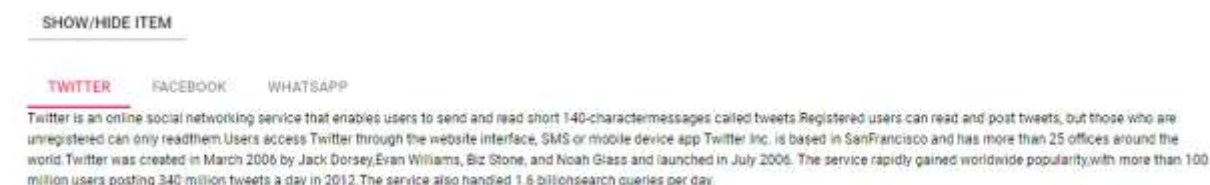
```
@using Syncfusion.Blazor.Navigations
@using Syncfusion.Blazor.Buttons
<SfButton @onclick="OnItemClick" Content="Show/Hide item"></SfButton>
<br />
<br />
<SfTab>
<TabItems>
<TabItem Visible="@ShowItem" Content="Twitter is an online social networking
service that enables users to send and read short 140-character messages
called tweets. Registered users can read and post tweets, but those who are
unregistered can only read them. Users access Twitter through the website
interface, SMS or mobile device app. Twitter Inc. is based in San Francisco
and has more than 25 offices around the world. Twitter was created in March
2006 by Jack Dorsey, Evan Williams, Biz Stone, and Noah Glass and launched in
July 2006. The service rapidly gained worldwide popularity, with more than
100 million users posting 340 million tweets a day in 2012. The service also
handled 1.6 billion search queries per day.">
<ChildContent>
<TabHeader Text="Twitter"></TabHeader>
```



```

</ChildContent>
</TabItem>
<TabItem Content="Facebook is an online social networking service
headquartered in Menlo Park, California. Its website was launched on February
4, 2004, by Mark Zuckerberg with his Harvard College roommates and fellow
students Eduardo Saverin, Andrew McCollum, Dustin Moskovitz and Chris
Hughes.">
<ChildContent>
<TabHeader Text="Facebook"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="WhatsApp Messenger is a proprietary cross-platform instant
messaging client for smartphones that operates under a subscription business
model. It uses the Internet to send text messages, images, video, user
location and audio media messages to other users using standard cellular
mobile numbers. As of February 2016, WhatsApp had a userbase of up to one
billion, [10] making it the most globally popular messaging
application. WhatsApp Inc., based in Mountain View, California, was acquired
by Facebook Inc. on February 19, 2014, for approximately US$19.3 billion.">
<ChildContent>
<TabHeader Text="Whatsapp"></TabHeader>
</ChildContent>
</TabItem>
</TabItems>
</SfTab>
@code {
public bool ShowItem = true;
void OnItemClick()
{
ShowItem = !ShowItem;
}
}

```



Enable/Disable Tab item in Blazor Tabs Component

The [Disabled](#) property of the Tab item is used to enable or disable the item by setting false or true value to the property. In the following demo, specified tab item is enabled and disabled dynamically when you click **Enable/Disable First Item** button.

ASPX-CS

```

@using Syncfusion.Blazor.Navigations
@using Syncfusion.Blazor.Buttons
<SfButton IsPrimary="true" @onclick="EnableItemClick"
Content="Enable/Disable First Item"></SfButton>
<br />
<br />
<SfTab>
<TabItems>

```

```

<TabItem Disabled=@IsEnable Content="Twitter is an online social networking
service that enables users to send and read short 140-character messages
called tweets. Registered users can read and post tweets, but those who are
unregistered can only read them. Users access Twitter through the website
interface, SMS or mobile device app Twitter Inc. is based in San Francisco
and has more than 25 offices around the world. Twitter was created in March
2006 by Jack Dorsey, Evan Williams, Biz Stone, and Noah Glass and launched in
July 2006. The service rapidly gained worldwide popularity, with more than
100 million users posting 340 million tweets a day in 2012. The service also
handled 1.6 billion search queries per day.">
<ChildContent>
<TabHeader Text="Twitter"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="Facebook is an online social networking service
headquartered in Menlo Park, California. Its website was launched on February
4, 2004, by Mark Zuckerberg with his Harvard College roommates and fellow
students Eduardo Saverin, Andrew McCollum, Dustin Moskovitz and Chris
Hughes.">
<ChildContent>
<TabHeader Text="Facebook"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="WhatsApp Messenger is a proprietary cross-platform instant
messaging client for smartphones that operates under a subscription business
model. It uses the Internet to send text messages, images, video, user
location and audio media messages to other users using standard cellular
mobile numbers. As of February 2016, WhatsApp had a userbase of up to one
billion, [10] making it the most globally popular messaging
application. WhatsApp Inc., based in Mountain View, California, was acquired
by Facebook Inc. on February 19, 2014, for approximately US$19.3 billion.">
<ChildContent>
<TabHeader Text="Whatsapp"></TabHeader>
</ChildContent>
</TabItem>
</TabItems>
</SfTab>
@code {
public bool IsEnable { get; set; } = false;
public void EnableItemClick()
{
IsEnable = !IsEnable;
}
}

```

ENABLE/DISABLE FIRST ITEM

TWITTER FACEBOOK WHATSAPP

Twitter is an online social networking service that enables users to send and read short 140-character messages called tweets. Registered users can read and post tweets, but those who are unregistered can only read them. Users access Twitter through the website interface, SMS or mobile device app Twitter Inc. is based in San Francisco and has more than 25 offices around the world. Twitter was created in March 2006 by Jack Dorsey, Evan Williams, Biz Stone, and Noah Glass and launched in July 2006. The service rapidly gained worldwide popularity, with more than 100 million users posting 340 million tweets a day in 2012. The service also handled 1.6 billion search queries per day.

Add nested tabs in Blazor Tabs Component

The tabs control supports to render the nested level of tabs by using the `ContentTemplate` property.

To render the nested tabs, define the nested tab elements within the `ContentTemplate` property of the parent tab.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<SfTab>
<TabItems>
<TabItem>
<ContentTemplate>
<SfTab>
<TabItems>
<TabItem Content="New York City comprises 5 boroughs sitting where the
Hudson River meets the Atlantic Ocean. At its core is Manhattan, a densely
populated borough that's among the world's major commercial, financial and
cultural centers. Its iconic sites include skyscrapers such as the Empire
State Building and sprawling Central Park. Broadway theater is staged in
neon-lit Times Square.">
<ChildContent>
<TabHeader Text="New York"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="Los Angeles is a sprawling Southern California city and
the center of the nation's film and television industry. Near its iconic
Hollywood sign, studios such as Paramount Pictures, Universal and Warner
Brothers offer behind-the-scenes tours. On Hollywood Boulevard, TCL Chinese
Theatre displays celebrities' hand- and footprints, the Walk of Fame honors
thousands of luminaries and vendors sell maps to stars' homes.">
<ChildContent>
<TabHeader Text="Los Angeles"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="Chicago, on Lake Michigan in Illinois, is among the
largest cities in the U.S. Famed for its bold architecture, it has a skyline
punctuated by skyscrapers such as the iconic John Hancock Center, 1,451-ft.
Willis Tower (formerly the Sears Tower) and the neo-Gothic Tribune Tower.
The city is also renowned for its museums, including the Art Institute of
Chicago with its noted Impressionist and Post-Impressionist works.">
<ChildContent>
<TabHeader Text="Chicago"></TabHeader>
</ChildContent>
</TabItem>
</TabItems>
</SfTab>
</ContentTemplate>
<ChildContent>
<TabHeader Text="USA"></TabHeader>
</ChildContent>
</TabItem>
<TabItem>
<ContentTemplate>
<SfTab>
<TabItems>
```

```

<TabItem Content="Paris, France capital, is a major European city and a
global center for art, fashion, gastronomy and culture. Its 19th-century
cityscape is crisscrossed by wide boulevards and the River Seine. Beyond
such landmarks as the Eiffel Tower and the 12th-century, Gothic Notre-Dame
cathedral, the city is known for its cafe culture and designer boutiques
along the Rue du Faubourg Saint-Honoré.">
<ChildContent>
<TabHeader Text="Paris"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="Marseille, a port city in southern France, has been a
crossroads of immigration and trade since its founding by the Greeks circa
600 B.C. At its heart is the Vieux-Port (Old Port), where fishmongers sell
their catch along the boat-lined quay. Basilique Notre-Dame-de-la-Garde is a
Romanesque-Byzantine church. Modern landmarks include Le Corbusier's
influential Cité Radieuse complex and Zaha Hadid's CMA CGM Tower.">
<ChildContent>
<TabHeader Text="Marseille"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="Lyon, the capital city in France's Auvergne-Rhône-Alpes
region, sits at the junction of the Rhône and Saône rivers. Its center
reflects 2,000 years of history from the Roman Amphithéâtre des Trois
Gaules, medieval and Renaissance architecture in Vieux (Old) Lyon, to the
modern Confluence district on Presquîle peninsula. Traboules, covered
passageways between buildings, connect Vieux Lyon and La Croix-Rousse
hill.">
<ChildContent>
<TabHeader Text="Lyon"></TabHeader>
</ChildContent>
</TabItem>
</TabItems>
</SfTab>
</ContentTemplate>
<ChildContent>
<TabHeader Text="France"></TabHeader>
</ChildContent>
</TabItem>
<TabItem>
<ContentTemplate>
<SfTab>
<TabItems>
<TabItem Content="Sydney, capital of New South Wales and one of Australia
largest cities, is best known for its harbourfront Sydney Opera House, with
a distinctive sail-like design. Massive Darling Harbour and the smaller
Circular Quay port are hubs of waterside life, with the arched Harbour
Bridge and esteemed Royal Botanic Garden nearby. Sydney Tower's outdoor
platform, the Skywalk, offers 360-degree views of the city and suburbs.">
<ChildContent>
<TabHeader Text="Sydney"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="Melbourne is the coastal capital of the southeastern
Australian state of Victoria. At the city centre is the modern Federation
Square development, with plazas, bars, and restaurants by the Yarra River.
In the Southbank area, the Melbourne Arts Precinct is the site of Arts

```

```

Centre Melbourne - a performing arts complex - and the National Gallery of
Victoria, with Australian and indigenous art">
<ChildContent>
<TabHeader Text="Melbourne"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="Brisbane, capital of Queensland, is a large city on the
Brisbane River. Clustered in its South Bank cultural precinct are the
Queensland Museum and Sciencentre, with noted interactive exhibitions.
Another South Bank cultural institution is Queensland Gallery of Modern Art,
among Australia major contemporary art museums. Looming over the city is Mt.
Coot-tha, site of Brisbane Botanic Gardens.">
<ChildContent>
<TabHeader Text="Brisbane"></TabHeader>
</ChildContent>
</TabItem>
</TabItems>
</SfTab>
</ContentTemplate>
<ChildContent>
<TabHeader Text="Australia"></TabHeader>
</ChildContent>
</TabItem>
</TabItems>
</SfTab>

```



Prevent content swipe selection in Blazor Tabs Component

The tab selection can be prevented on touch swipe action by using the Tab [Selecting](#) event.

ASPX-CS

```

@using Syncfusion.Blazor.Navigations
<SfTab>
<TabEvents Selecting="Select">
</TabEvents>
<TabItems>
<TabItem Content="Twitter is an online social networking service that
enables users to send and read short 140-character messages called
tweets. Registered users can read and post tweets, but those who are
unregistered can only read them. Users access Twitter through the website
interface, SMS or mobile device app. Twitter Inc. is based in San Francisco
and has more than 25 offices around the world. Twitter was created in March
2006 by Jack Dorsey, Evan Williams, Biz Stone, and Noah Glass and launched in
July 2006. The service rapidly gained worldwide popularity, with more than
100 million users posting 340 million tweets a day in 2012. The service also
handled 1.6 billion search queries per day.">

```

```

<ChildContent>
<TabHeader Text="Twitter">
</TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="Facebook is an online social networking service
headquartered in Menlo Park, California. Its website was launched on February
4, 2004, by Mark Zuckerberg with his Harvard College roommates and fellow
students Eduardo Saverin, Andrew McCollum, Dustin Moskovitz and Chris
Hughes.">
<ChildContent>
<TabHeader Text="Facebook">
</TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="WhatsApp Messenger is a proprietary cross-platform instant
messaging client for smartphones that operates under a subscription business
model. It uses the Internet to send text messages, images, video, user
location and audio media messages to other users using standard cellular
mobile numbers. As of February 2016, WhatsApp had a userbase of up to one
billion, [10] making it the most globally popular messaging
application. WhatsApp Inc., based in Mountain View, California, was acquired
by Facebook Inc. on February 19, 2014, for approximately US$19.3 billion.">
<ChildContent>
<TabHeader Text="WhatsApp">
</TabHeader>
</ChildContent>
</TabItem>
</TabItems>
</SfTab>
@code {
public void Select(SelectingEventArgs args)
{
if (args.IsSwiped)
{
args.Cancel = true;
}
}
}
}

```

Create wizard in Blazor Tabs Component

Tab items can be disabled during initial control rendering by passing the boolean value and tab index to the [EnableTab](#) public method.

In the below Wizard sample, each tab is integrated with required components to complete the reservation. Each field is provided with validation for all mandatory option to proceed to next tabs. Using tab item's template property the components are added into content.

Create the following contents for each tab in the wizard.

1. Search tab:

Created with [DropDownList](#) to select the source, destination and type of ticket. A [DatePicker](#) for choosing the date of journey.

2. Train tab:

Based on the selected start and end point, populate the Grid with random list of available seats and train list. Initially define the columns and row selected event for validating, after the source and destination chosen update the `DataSource` for the Grid.

3. Passenger tab:

A table with Textbox, Numeric, DropDownList for adding passenger name, age, gender and preferred berth/seat. Add validation on entering passenger details to proceed.

4. Payment tab:

Calculate the ticket cost and generate data for Grid with passenger details, train number and ticket cost summary.

You can go back on each tab using buttons available in it and tabs are disabled to navigate through tab header click actions.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
@using Syncfusion.Blazor.DropDowns
@using Syncfusion.Blazor.Calendars
@using Syncfusion.Blazor.Grids
@using Syncfusion.Blazor.Inputs
@using Syncfusion.Blazor.Popups
@using Syncfusion.Blazor.Buttons
<div class="control-section e-tab-section">
<SfTab @ref="Tab" ID="BlazorTab" Height="390">
<TabEvents Created="TabCreate"></TabEvents>
<TabItems>
<TabItem>
<ChildContent>
<TabHeader Text="New Booking"></TabHeader>
</ChildContent>
<ContentTemplate>
<div id="booking">
<div class="wizard-title">Plan your journey</div>
<div class="responsive-align">
<div class="row">
<div class="col-xs-6 col-sm-6 col-lg-6 col-md-6 search-item">
<SfDropDownList TValue="string" TItem="CitiesFields" @ref="StartPoint"
Width="100%" DataSource="@CitiesData" Placeholder="From">
<DropDownListFieldSettings Text="Name"
Value="Name"></DropDownListFieldSettings>
</SfDropDownList>
</div>
<div class="col-xs-6 col-sm-6 col-lg-6 col-md-6 search-item">
<SfDropDownList @ref="EndPoint" TValue="string" TItem="CitiesFields"
Width="100%" DataSource="@CitiesData" Placeholder="To">
<DropDownListFieldSettings Text="Name"
Value="Name"></DropDownListFieldSettings>
</SfDropDownList>
</div>
</div>
</div>
</div>
</div>
```

```

</div>
</div>
<div class="row">
<div class="col-xs-6 col-sm-6 col-lg-6 col-md-6 search-item">
<SfDatePicker @ref="Date" Width="100%" Placeholder="Journey Date"
FloatLabelType="FloatLabelType.Auto" TValue="DateTime?"
Value="@ (DateTime.Now) " Min="@ (DateTime.Now) "
Max="@ (DateTime.Now.AddMonths (3) ) "></SfDatePicker>
</div>
<div class="col-xs-6 col-sm-6 col-lg-6 col-md-6 search-item">
<SfDropDownList @ref="Quota" TValue="string" TItem="DropDownFields"
FloatLabelType="FloatLabelType.Auto" DataSource="@QuotaList"
Placeholder="Ticket type">
<DropDownListFieldSettings Text="Text"
Value="Text"></DropDownListFieldSettings>
</SfDropDownList>
</div>
</div>
<div class="btn-container">
<SfButton @onclick="SearchButtonClick">Search Train</SfButton>
</div>
@if (EmptyField)
{
<span class="error">* Please fill all the details before proceeding</span>
}
@if (SameField)
{
<span class="error">* Arrival point can't be same as Departure</span>
}
</div>
</div>
</ContentTemplate>
</TabItem>
<TabItem>
<ChildContent>
<TabHeader Text="Train List"></TabHeader>
</ChildContent>
<ContentTemplate>
<div id="selectTrain">
<div class="wizard-title">Select the train from the list </div>
<SfGrid @ref="AvailableTrain" Width="100%" TValue="GridFields"
DataSource="@GridData">
<GridEvents RowSelected="SelectedTrainDetails"
TValue="GridFields"></GridEvents>
<GridColumn>
<GridColumn Field=@nameof(GridFields.TrainNo) HeaderText="Train No"
Width="120" Type="ColumnType.Number"></GridColumn>
<GridColumn Field=@nameof(GridFields.Name) Width="140"></GridColumn>
<GridColumn Field=@nameof(GridFields.Departure) HeaderText="Departure"
Width="120"></GridColumn>
<GridColumn Field=@nameof(GridFields.Arrival) HeaderText="Arrival"
Width="140"></GridColumn>
<GridColumn Field=@nameof(GridFields.Availability) HeaderText="Availability"
Width="140" Type="ColumnType.Number"></GridColumn>
</GridColumn>
</SfGrid>
<br />

```



```

<div class="btn-container">
<SfButton @onclick="@SelectTrainBack">Back</SfButton>
<SfButton @onclick="@SelectTrainContinue">Continue</SfButton>
</div>
@if (EmptyField)
{
<span class="error">* Select your convenient train</span>
}
</div>
</ContentTemplate>
</TabItem>
<TabItem>
<ChildContent>
<TabHeader Text="Add Passenger"></TabHeader>
</ChildContent>
<ContentTemplate>
<div id="details">
<div class="details-page wizard-title">Enter the passenger details</div>
<div id="PassengersList">
<table id="passenger-table">
<colgroup>
<col />
<col />
<col />
<col />
<col />
<col />
</colgroup>
<thead>
<tr>
<th class="name-header">Name</th>
<th class="age-header">Age</th>
<th class="gender-header">Gender</th>
<th class="type-header">Berth Preference</th>
</tr>
</thead>
<tbody>
<tr>
<td>
<SfTextBox @ref="@FirstPassengerName" Placeholder="Passenger
Name"></SfTextBox>
</td>
<td>
<SfNumericTextBox @ref="FirstPassengerAge" Min="1" Max="100" Value="18"
Format=n0 ShowSpinButton="false"></SfNumericTextBox>
</td>
<td>
<SfDropDownList @ref="FirstPassengerGender" TValue="string"
TItem="DropDownFields" Text="Male" DataSource="@Gender">
<DropDownListFieldSettings Text="Text"
Value="Text"></DropDownListFieldSettings>
</SfDropDownList>
</td>
<td>
<SfDropDownList @ref="FirstPassengerBerth" TValue="string"
TItem="DropDownFields" Placeholder="Optional" DataSource="@Berth">

```

```

<DropDownListFieldSettings Text="Text"
Value="Text"></DropDownListFieldSettings>
</SfDropDownList>
</td>
</tr>
<tr>
<td>
<SfTextBox @ref="@SecondPassengerName" Placeholder="Passenger
Name"></SfTextBox>
</td>
<td>
<SfNumericTextBox @ref="SecondPassengerAge" Min="1" Max="100" Value="18"
Format=n0 ShowSpinButton="false"></SfNumericTextBox>
</td>
<td>
<SfDropDownList @ref="SecondPassengerGender" TValue="string"
TItem="DropdownFields" Text="Male" DataSource="@Gender">
<DropDownListFieldSettings Text="Text"
Value="Text"></DropDownListFieldSettings>
</SfDropDownList>
</td>
<td>
<SfDropDownList @ref="SecondPassengerBerth" TValue="string"
TItem="DropdownFields" Placeholder="Optional" DataSource="@Berth">
<DropDownListFieldSettings Text="Text"
Value="Text"></DropDownListFieldSettings>
</SfDropDownList>
</td>
</tr>
<tr>
<td>
<SfTextBox @ref="@ThirdPassengerName" Placeholder="Passenger
Name"></SfTextBox>
</td>
<td>
<SfNumericTextBox @ref="ThirdPassengerAge" Min="1" Max="100" Value="18"
Format=n0 ShowSpinButton="false"></SfNumericTextBox>
</td>
<td>
<SfDropDownList @ref="ThirdPassengerGender" TValue="string"
TItem="DropdownFields" Text="Male" DataSource="@Gender">
<DropDownListFieldSettings Text="Text"
Value="Text"></DropDownListFieldSettings>
</SfDropDownList>
</td>
<td>
<SfDropDownList @ref="ThirdPassengerBerth" TValue="string"
TItem="DropdownFields" Placeholder="Optional" DataSource="@Berth">
<DropDownListFieldSettings Text="Text"
Value="Text"></DropDownListFieldSettings>
</SfDropDownList>
</td>
</tr>
</tbody>
</table>
</div>
<br />

```

```

<div class="btn-container">
<SfButton @onclick="@PassengerListBack">Back</SfButton>
<SfButton @onclick="@PassengerListContinue">Continue</SfButton>
</div>
@if (EmptyField)
{
<span class="error">* Please enter passenger details</span>
}
</div>
</ContentTemplate>
</TabItem>
<TabItem>
<ChildContent>
<TabHeader Text="Make Payment"></TabHeader>
</ChildContent>
<ContentTemplate>
<div id="confirm">
<div class="tab-title1 wizard-title">Confirm the details and proceed</div>
<SfGrid @ref="TicketDetailGrid" Width="100%" TValue="PassengerFields"
DataSource="@PassengerData">
<GridColumn>
<GridColumn Field=@nameof(PassengerFields.TrainNo) HeaderText="Train No"
Width="120" Type="ColumnType.Number"></GridColumn>
<GridColumn Field=@nameof(PassengerFields.Name) HeaderText="Name"
Width="140"></GridColumn>
<GridColumn Field=@nameof(PassengerFields.Gender) HeaderText="Gender"
Width="120"></GridColumn>
<GridColumn Field=@nameof(PassengerFields.Berth) Width="140"
Type="ColumnType.Number"></GridColumn>
</GridColumn>
</SfGrid>
<br />
<div id="amount">Total payable amount: $1700</div>
<br />
<div class="btn-container">
<SfButton @onclick="@ConfirmBack">Back</SfButton>
<SfButton @onclick="@ConfirmPayment">Pay</SfButton>
</div>
</div>
</ContentTemplate>
</TabItem>
</TabItems>
</SfTab>
<div>
<SfDialog @ref="AlertDialog" Width=250 Target="#BlazorTab" IsModal=true
Visible=false ShowCloseIcon="true">
<DialogEvents Created="DialogCreate"></DialogEvents>
<DialogTemplates>
<Header><div>Success</div></Header>
<Content><div>Your payment successfully processed</div></Content>
</DialogTemplates>
<DialogButtons>
<DialogButton OnClick="@OnSubmit" Content="OK" IsPrimary="true">
</DialogButton>
</DialogButtons>
</SfDialog>
</div>

```

```

</div>
@code{
SfTab Tab;
public SfDropDownList<string, CitiesFields> StartPoint;
public SfDropDownList<string, CitiesFields> EndPoint;
public SfDropDownList<string, DropdownFields> Quota;
public SfGrid<GridFields> AvailableTrain;
public SfGrid<PassengerFields> TicketDetailGrid;
public int SelectedTrain { get; set; }
public string SelectedTrainName { get; set; }
public Boolean IsSelectedTrain { get; set; } = false;
SfDialog AlertDialog;
SfDatePicker<DateTime?> Date;
SfTextBox FirstPassengerName;
SfTextBox SecondPassengerName;
SfTextBox ThirdPassengerName;
public SfNumericTextBox<int> FirstPassengerAge;
public SfNumericTextBox<int> SecondPassengerAge;
public SfNumericTextBox<int> ThirdPassengerAge;
public SfDropDownList<string, DropdownFields> FirstPassengerGender;
public SfDropDownList<string, DropdownFields> SecondPassengerGender;
public SfDropDownList<string, DropdownFields> ThirdPassengerGender;
public SfDropDownList<string, DropdownFields> FirstPassengerBerth;
public SfDropDownList<string, DropdownFields> SecondPassengerBerth;
public SfDropDownList<string, DropdownFields> ThirdPassengerBerth;
public bool EmptyField { get; set; } = false;
public bool SameField { get; set; } = false;
public List<PassengerFields> PassengerData;
public class CitiesFields
{
public string Name { get; set; }
public int Fare { get; set; }
}
public List<CitiesFields> CitiesData = new List<CitiesFields>()
{
new CitiesFields() { Name = "Chicago", Fare = 300 },
new CitiesFields() { Name = "San Francisco", Fare = 125 },
new CitiesFields() { Name = "Los Angeles", Fare = 175 },
new CitiesFields() { Name = "Seattle", Fare = 250 },
new CitiesFields() { Name = "Florida", Fare = 150 }
};
public class DropdownFields
{
public string ID { get; set; }
public string Text { get; set; }
}
public List<DropdownFields> QuotaList = new List<DropdownFields>()
{
new DropdownFields { ID = "1", Text = "Business Class" },
new DropdownFields { ID = "2", Text = "Economy Class" },
new DropdownFields { ID = "3", Text = "Common Class" }
};
public async Task SearchButtonClick()
{
if ((StartPoint.Value == null) || (EndPoint.Value == null) || (Quota.Value
== null))
{

```

```

EmptyField = true;
}
else if (StartPoint.Value == EndPoint.Value)
{
    SameField = true;
}
else
{
    EmptyField = false;
    SameField = false;
    GridData = new List<GridFields>() {
        new GridFields
        {
            TrainNo = 1000,
            Name = "Train" + "1",
            Departure = StartPoint.Value,
            Arrival = EndPoint.Value,
            Availability = 30
        },
        new GridFields
        {
            TrainNo = 1002,
            Name = "Train" + "2",
            Departure = StartPoint.Value,
            Arrival = EndPoint.Value,
            Availability = 28
        }
    };
    await Tab.EnableTabAsync(1, true);
    await Tab.SelectAsync(1);
}
}

public void SelectedTrainDetails(RowSelectEventArgs<GridFields> args)
{
    IsSelectedTrain = true;
    if (IsSelectedTrain)
    {
        EmptyField = false;
        SelectedTrain = args.Data.TrainNo;
    }
    else
    {
        EmptyField = true;
    }
}

public class GridFields
{
    public int TrainNo { get; set; }
    public string Name { get; set; }
    public string Departure { get; set; }
    public string Arrival { get; set; }
    public int Availability { get; set; }
}

public class PassengerFields
{
    public int TrainNo { get; set; }
    public string Name { get; set; }
}

```

```

public string Gender { get; set; }
public string Berth { get; set; }
}
public List<GridFields> GridData;
public List<DropdownFields> Gender = new List<DropdownFields>() {
    new DropdownFields { ID = "1", Text = "Male" },
    new DropdownFields { ID = "2", Text = "Female" }
};
public List<DropdownFields> Berth = new List<DropdownFields>()
{
    new DropdownFields { ID = "1", Text = "Upper" },
    new DropdownFields { ID = "2", Text = "Lower" },
    new DropdownFields { ID = "2", Text = "Middle" },
    new DropdownFields { ID = "2", Text = "Window" },
    new DropdownFields { ID = "2", Text = "Aisle" }
};
public async Task TabCreate()
{
    await Tab.EnableTabAsync(1, false);
    await Tab.EnableTabAsync(2, false);
    await Tab.EnableTabAsync(3, false);
}
public async Task SelectTrainBack()
{
    await Tab.SelectAsync(0);
}
public async Task SelectTrainContinue()
{
    if (IsSelectedTrain)
    {
        EmptyField = false;
        await Tab.EnableTabAsync(2, true);
        await Tab.SelectAsync(2);
    }
    else
    {
        await Tab.EnableTabAsync(2, false);
        EmptyField = true;
    }
}
public async Task PassengerListBack()
{
    await Tab.SelectAsync(1);
}
public async Task PassengerListContinue()
{
    if (FirstPassengerName.Value == null)
    {
        EmptyField = true;
    }
    else
    {
        EmptyField = false;
        await Tab.EnableTabAsync(3, true);
        await Tab.SelectAsync(3);
        PassengerData = new List<PassengerFields>();
        if (FirstPassengerName.Value != null)

```

```

{
    PassengerData.Add(new PassengerFields
    {
        TrainNo = SelectedTrain,
        Name = FirstPassengerName.Value,
        Gender = FirstPassengerGender.Value,
        Berth = FirstPassengerBerth.Value
    });
}
if (SecondPassengerName.Value != null)
{
    PassengerData.Add(new PassengerFields
    {
        TrainNo = SelectedTrain,
        Name = SecondPassengerName.Value,
        Gender = SecondPassengerGender.Value,
        Berth = SecondPassengerBerth.Value
    });
}
if (ThirdPassengerName.Value != null)
{
    PassengerData.Add(new PassengerFields
    {
        TrainNo = SelectedTrain,
        Name = ThirdPassengerName.Value,
        Gender = ThirdPassengerGender.Value,
        Berth = ThirdPassengerBerth.Value
    });
}
}
}
public async Task ConfirmBack()
{
    await Tab.SelectAsync(2);
}
public async Task ConfirmPayment()
{
    await AlertDialog.ShowAsync();
}
public async Task DialogCreate()
{
    await AlertDialog.HideAsync();
}
public async Task OnSubmit(Object args)
{
    await AlertDialog.HideAsync();
    await Tab.EnableTabAsync(0, true);
    await Tab.EnableTabAsync(1, false);
    await Tab.EnableTabAsync(2, false);
    await Tab.EnableTabAsync(3, false);
    await Tab.SelectAsync(0);
}
}
<style>
.wizard-title {
font-size: 15px;
padding: 7px;

```

```

}
.responsive-align {
width: 75%;
margin: 0 auto;
}
.error {
color: red;
}
.search-item {
padding-right: 50px;
padding-bottom: 20px;
}
#amount {
text-align: right;
font-size: 15px;
padding: 15px 0px;
}
#passenger-table th {
text-align: center;
font-size: 14px;
font-weight: 400;
border: 1px solid gainsboro;
}
#passenger-table td, th {
padding: 10px;
}
#passenger-table td {
border: 1px solid gainsboro;
}
</style>

```

The screenshot shows a web interface for train booking. At the top, there are four tabs: 'NEW BOOKING' (highlighted with a red underline), 'TRAIN LIST', 'ADD PASSENGER', and 'MAKE PAYMENT'. Below the tabs is a section titled 'Plan your journey'. It contains four input fields: 'From' and 'To' (both with dropdown arrows), 'Journey Date' (with a calendar icon and the date '1/8/2020'), and 'Ticket type' (with a dropdown arrow). At the bottom of this section is a button labeled 'SEARCH TRAIN'.

Style Customization for active Item in Blazor Tabs Component

The style of tabs can be customized by overriding its header and active tab CSS classes. Define HTML string for adding animation and customizing the tab header and pass it to [Text](#) property. Now you can override the style using custom CSS classes added to the tab elements.

ASPX-CS

```

@using Syncfusion.Blazor.Navigations
<SfTab CssClass="e-tab-custom-class">

```



```

<TabItems>
<TabItem Content="Andrew received his BTS commercial in 1974 and a Ph.D. in
international marketing from the University of Dallas in 1981.He is fluent
in French and Italian and reads German.He joined the company as a sales
representative, was promoted to sales manager in January 1992 and to vice
president of sales in March 1993.Andrew is a member of the Sales Management
Roundtable, the Seattle Chamber of Commerce, and the Pacific Rim Importers
Association.">
<HeaderTemplate>

<div class="e-title fade-in">Andrew</div>
</HeaderTemplate>
</TabItem>
<TabItem Content="Margaret holds a BA in English literature from Concordia
College (1958) and an MA from the American Institute of Culinary Arts
(1966).She was assigned to the London office temporarily from July through
November 1992.">
<HeaderTemplate>

<div class="e-title fade-in">Margaret</div>
</HeaderTemplate>
</TabItem>
<TabItem Content="Janet has a BS degree in chemistry from Boston College
(1984).She has also completed a certificate program in food retailing
management.Janet was hired as a sales associate in 1991 and promoted to
sales representative in February 1992.">
<HeaderTemplate>

<div class="e-title fade-in">Janet</div>
</HeaderTemplate>
</TabItem>
</TabItems>
</SfTab>
<style>
.e-toolbar-item.e-active .e-tab-wrap .emp {
display: none;
}
.e-content .e-item {
font-size: 12px;
margin: 10px;
text-align: justify;
}
.e-image {
background-size: 33px;
width: 33px;
height: 30px;
margin: 0 auto;
}
.e-tab .e-toolbar-item .e-title {
display: none;
}
.e-toolbar .e-toolbar-items .e-toolbar-item:not(.e-separator),
.e-toolbar .e-toolbar-items .e-toolbar-item .e-tab-wrap {
width: 105px;
height: 50px;
}
.e-tab .e-tab-header .e-toolbar-items .e-active .e-tab-wrap {

```

```

background-color: #08c;
}
.e-tab .e-tab-header {
background-color: #e6e6e6;
}
.e-tab .e-tab-header .e-toolbar-item:not(.e-active) .e-tab-wrap:hover {
background-color: #f2f2f2;
color:#000;
}
.e-tab .e-toolbar .e-toolbar-items .e-toolbar-item .e-text-wrap {
height: 50px;
}
.e-tab .e-toolbar-items .e-active .e-image {
display: none;
}
.e-tab .e-toolbar-items .e-active .e-title {
display: block;
color: white;
}
.e-tab .e-toolbar-item .e-text-wrap,
.e-tab .e-toolbar-item .e-tab-text {
width: inherit;
text-align: center;
}
.fade-in {
opacity: 1;
animation-name: fadeInOpacity;
animation-iteration-count: 1;
animation-timing-function: ease-in;
animation-duration: 0.5s;
}
@@keyframes fadeInOpacity {
0% {
opacity: 0;
}
100% {
opacity: 1;
}
}
</style>

```



ANDREW

Andrew received his BTS commercial in 1974 and a Ph.D. in international marketing from the University of Dallas in 1981. He is fluent in French and Italian and reads German. He joined the company as a sales representative, was promoted to sales manager in January 1992 and to vice president of sales in March 1993. Andrew is a member of the Sales Management Roundtable, the Seattle Chamber of Commerce, and the Pacific Rim Importers Association.

Set state persistence in Blazor Tabs Component

State persistence allows the Tab to retain the current [SelectedItem](#) property value in the browser cookies for state maintenance. This action is handled through the [EnablePersistence](#) property which is

set to false by default. When it is set to true, [SelectedItem](#) property value of the Tab component will be retained even after refreshing the page.

Tab ID is essential to set state persistence.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<SfTab ID="Tab" Width="600px" EnablePersistence="true" >
  <TabItems>
    <TabItem Content="@Content1">
      <ChildContent>
        <TabHeader Text="Twitter"></TabHeader>
      </ChildContent>
    </TabItem>
    <TabItem Content="@Content2">
      <ChildContent>
        <TabHeader Text="Facebook"></TabHeader>
      </ChildContent>
    </TabItem>
    <TabItem Content="@Content3">
      <ChildContent>
        <TabHeader Text="WhatsApp"></TabHeader>
      </ChildContent>
    </TabItem>
  </TabItems>
</SfTab>

@code {
public string Content1 = "Twitter is an online social networking service
that enables users to send and read short 140-character " +
"messages called 'tweets'. Registered users can read and post tweets, but
those who are unregistered can only read " +
"them. Users access Twitter through the website interface, SMS or mobile
device app Twitter Inc. is based in San " +
"Francisco and has more than 25 offices around the world. Twitter was
created in March 2006 by Jack Dorsey, " +
"Evan Williams, Biz Stone, and Noah Glass and launched in July 2006. The
service rapidly gained worldwide popularity, " +
"with more than 100 million users posting 340 million tweets a day in
2012.The service also handled 1.6 billion " +
"search queries per day.";
public string Content2 = "Facebook is an online social networking service
headquartered in Menlo Park, California. Its website was " +
"launched on February 4, 2004, by Mark Zuckerberg with his Harvard College
roommates and fellow students Eduardo " +
"Saverin, Andrew McCollum, Dustin Moskovitz and Chris Hughes.The founders
had initially limited the website " +
"membership to Harvard students, but later expanded it to colleges in the
Boston area, the Ivy League, and Stanford " +
"University. It gradually added support for students at various other
universities and later to high-school students.";
public string Content3 = "WhatsApp Messenger is a proprietary cross-platform
instant messaging client for smartphones that operates " +
"under a subscription business model. It uses the Internet to send text
messages, images, video, user location and " +
"audio media messages to other users using standard cellular mobile numbers.
As of February 2016, WhatsApp had a user " +
```

```
"base of up to one billion,[10] making it the most globally popular
messaging application. WhatsApp Inc., based in " +
"Mountain View, California, was acquired by Facebook Inc. on February 19,
2014, for approximately US$19.3 billion.";
}
```

Customize the Scrolling distance in Blazor Tabs Component

The [ScrollStep](#) property supports to customize the scrolling distance when you click the left and right side navigation icons. A required value can be passed through `ScrollStep` property to customize tab scrolling distance.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<SfTab Width="400px" OverflowMode="OverflowMode.Scrollable"
ScrollStep="150">
<TabItems>
<TabItem Content="@Content0">
<ChildContent>
<TabHeader Text="HTML"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="@Content1">
<ChildContent>
<TabHeader Text="C Sharp (C#)"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="@Content2">
<ChildContent>
<TabHeader Text="Java"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="@Content3">
<ChildContent>
<TabHeader Text="VB.Net"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="@Content4">
<ChildContent>
<TabHeader Text="Xamarin"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="@Content5">
<ChildContent>
<TabHeader Text="ASP.NET"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="@Content6">
<ChildContent>
<TabHeader Text="ASP.NET MVC"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="@Content7">
<ChildContent>
<TabHeader Text="JavaScript"></TabHeader>
</ChildContent>
</TabItem>
```

```

</TabItem>
</TabItems>
</SfTab>
@code{
public string Content0 = "HyperText Markup Language, commonly referred to as
HTML, is the standard markup language used to create webpages.Along with
CSS, and JavaScript, HTML is a cornerstone technology, used by most websites
to create visuallyengaging web pages, user interfaces for web applications,
and user interfaces for many mobile applications.[1] Webrowsers can read
HTML files and render them into visible or audible web pages.HTML describes
the structure of awebsite semantically along with cues for presentation,
making it a markup language, rather than a programming language.";
public string Content1 = "C# is intended to be a simple, modern, general-
purpose, object-oriented programming language. Its developmentteam is led by
Anders Hejlsberg.The most recent version is C# 5.0, which was released on
August 15, 2012.";
public string Content2 = "Java is a set of computer software and
specifications developed by Sun Microsystems, later acquired by
OracleCorporation, that provides a system for developing application
software and deploying it in a cross - platform computingenvironment.Java is
used in a wide variety of computing platforms from embedded devices and
mobile phones toenterprise servers and supercomputers.While less common,
Java applets run in secure, sandboxed environments toprovide many features
of native applications and can be embedded in HTML pages.";
public string Content3 = "The command-line compiler, VBC.EXE, is installed
as part of the freeware .NET Framework SDK. Mono alsoincludes a command -
line VB.NET compiler.The most recent version is VB 2012, which was released
on August 15, 2012.";
public string Content4 = "Xamarin is a San Francisco, California based
software company created in May 2011[3] by the engineers that created
Mono,[4] Mono for Android and MonoTouch that are cross-platform
implementations of the Common Language Infrastructure (CLI) and Common
Language Specifications (often called Microsoft .NET). With a C#-shared
codebase, developers can use Xamarin tools to write native Android,iOS, and
Windows apps with native user interfaces and share code across multiple
platforms.[5] Xamarin has over 1 million developersin more than 120
countries around the World as of May 2015.";
public string Content5 = "ASP.NET is an open-source server-side web
application framework designed for web development to producedynamic web
pages.It was developed by Microsoft to allow programmers to build dynamic
web sites, web applicationsand web services.It was first released in January
2002 with version 1.0 of the.NET Framework, and is the successor to Microsoft
Active Server Pages(ASP) technology.ASP.NET is built on the Common Language
Runtime(CLR), allowingprogrammers to write ASP.NET code using any supported
.NET language. The ASP.NET SOAP extension framework allowsASP.NET components
to process SOAP messages.";
public string Content6 = "The ASP.NET MVC is a web application framework
developed by Microsoft, which implements themodel-view-controller(MVC)
pattern.It is open - source software, apart from the ASP.NET Web Forms
component which isproprietary.In the later versions of ASP.NET, ASP.NET MVC,
ASP.NET Web API, and ASP.NET Web Pages(a platform usingonly Razor pages)
will merge into a unified MVC 6.The project is called ASP.NET Next.";
public string Content7 = "JavaScript (JS) is an interpreted computer
programming language. It was originally implemented as part ofweb browsers
so that client - side scripts could interact with the user, control the
browser, communicateasynchronously, and alter the document content that was

```

```
displayed.[5] More recently, however, it has become common in both game
development and the creation of desktop applications.";
}
```

< **HTML** C SHARP(C#) JAVA VB.NET >

HyperText Markup Language, commonly referred to as HTML, is the standard markup language used to create web pages. Along with CSS, and JavaScript, HTML is a cornerstone technology, used by most websites to create visually engaging web pages, user interfaces for web applications, and user interfaces for many mobile applications.[1] Web browsers can read HTML files and render them into visible or audible web pages. HTML describes the structure of a website semantically along with cues for presentation, making it a markup language, rather than a programming language.

Prevent reorder active tab in Blazor Tabs Component

By default, the active tab will reorder when you click the tab items from the popup or resize the browser on popup mode. If we set `false` to `ReorderActiveTab` property, the active tab item from the popup will not be reordered and an active item is highlighted inside the popup itself and the content of the selected tab will display. The following code example depicts how to prevent the reorder active tab item inside the popup.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<SfTab OverflowMode="OverflowMode.Popup" Width="500px"
ReorderActiveTab="false">
  <TabItems>
    <TabItem Content="HyperText Markup Language, commonly referred to as HTML,
is the standard markup language used to create web pages. Along with CSS,
and JavaScript, HTML is a cornerstone technology, used by most websites to
create visually engaging web pages, user interfaces for web applications,
and user interfaces for many mobile applications.[1] Web browsers can read
HTML files and render them into visible or audible web pages. HTML describes
the structure of a website semantically along with cues for presentation,
making it a markup language, rather than a programming language.">
      <ChildContent>
        <TabHeader Text="HTML"></TabHeader>
      </ChildContent>
    </TabItem>
    <TabItem Content="C# is intended to be a simple, modern, general-purpose,
object-oriented programming language. Its development team is led by Anders
Hejlsberg. The most recent version is C# 5.0, which was released on August
15, 2012.">
      <ChildContent>
        <TabHeader Text="C Sharp (C#)"></TabHeader>
      </ChildContent>
    </TabItem>
  </TabItems>
</SfTab>
```

```

</TabItem>
<TabItem Content="Java is a set of computer software and specifications
developed by Sun Microsystems, later acquired by Oracle Corporation, that
provides a system for developing application software and deploying it in a
cross-platform computing environment. Java is used in a wide variety of
computing platforms from embedded devices and mobile phones to enterprise
servers and supercomputers. While less common, Java applets run in secure,
sandboxed environments to provide many features of native applications and
can be embedded in HTML pages.">
<ChildContent>
<TabHeader Text="Java"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="The command-line compiler, VBC.EXE, is installed as part
of the freeware .NET Framework SDK. Mono also includes a command-line VB.NET
compiler. The most recent version is VB 2012, which was released on August
15, 2012.">
<ChildContent>
<TabHeader Text="VB.Net"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="Xamarin is a San Francisco, California based software
company created in May 2011[3] by the engineers that created Mono,[4] Mono
for Android and MonoTouch that are cross-platform implementations of the
Common Language Infrastructure (CLI) and Common Language Specifications
(often called Microsoft .NET). With a C#-shared codebase, developers can use
Xamarin tools to write native Android, iOS, and Windows apps with native
user interfaces and share code across multiple platforms.[5] Xamarin has
over 1 million developers in more than 120 countries around the World as of
May 2015.">
<ChildContent>
<TabHeader Text="Xamarin"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="ASP.NET is an open-source server-side web application
framework designed for web development to produce dynamic web pages. It was
developed by Microsoft to allow programmers to build dynamic web sites, web
applications and web services. It was first released in January 2002 with
version 1.0 of the .NET Framework, and is the successor to Microsoft's
Active Server Pages (ASP) technology. ASP.NET is built on the Common
Language Runtime (CLR), allowing programmers to write ASP.NET code using any
supported .NET language. The ASP.NET SOAP extension framework allows ASP.NET
components to process SOAP messages.">
<ChildContent>
<TabHeader Text="ASP.NET"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="The ASP.NET MVC is a web application framework developed
by Microsoft, which implements the model-view-controller (MVC) pattern. It
is open-source software, apart from the ASP.NET Web Forms component which is
proprietary. In the later versions of ASP.NET, ASP.NET MVC, ASP.NET Web API,
and ASP.NET Web Pages (a platform using only Razor pages) will merge into a
unified MVC 6.The project is called ASP.NET vNext.">
<ChildContent>
<TabHeader Text="ASP.NET MVC"></TabHeader>
</ChildContent>
</TabItem>

```

```

<TabItem 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.[5] More recently, however, it has become common in both game
development and the creation of desktop applications.">
<ChildContent>
<TabHeader Text="JavaScript"></TabHeader>
</ChildContent>
</TabItem>
</TabItems>
</SfTab>

```

Find interaction in Blazor Tabs Component

You can find whether the tab is selected by user interactions or programmatically in the [Selecting](#) and [Selected](#) event with the field `IsInteracted`. When the user changes the tab by clicking on the tab header, it will return true. Otherwise, it will return false. The following code example depicts how to find the tab selecting state in Selected event.

ASPX-CS

```

@using Syncfusion.Blazor.Navigations
@using Syncfusion.Blazor.DropDowns
<div class="row">
<div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
<SfDropDownList TValue="int" TItem="DropDownData"
FloatLabelType="Syncfusion.Blazor.Inputs.FloatLabelType.Always" @bind-
Index="@workIndex" Placeholder="Select Tab Item using dropdown"
DataSource="@DropDownDatasource">
<DropDownListFieldSettings Text="Text"
Value="ID"></DropDownListFieldSettings>
<DropDownListEvents TValue="int" TItem="DropDownData"
ValueChange="OnChange"></DropDownListEvents>
</SfDropDownList>
</div>
</div>
<br />
<div class="eventlog" style="word-break: normal;">
<span><b>@EventLog</b></span>
</div>
<SfTab @ref="Tab" Width="600px">
<TabEvents Selected="Selected">
</TabEvents>
<TabItems>
<TabItem Content="@Content1">
<ChildContent>
<TabHeader Text="Twitter"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="@Content2">
<ChildContent>
<TabHeader Text="Facebook"></TabHeader>
</ChildContent>
</TabItem>
<TabItem Content="@Content3">

```



```

<ChildContent>
<TabHeader Text="WhatsApp"></TabHeader>
</ChildContent>
</TabItem>
</TabItems>
</SfTab>
@code{
SfTab Tab;
private int? workIndex { get; set; } = 0;
String EventLog = null;
public string Content1 = "Twitter is an online social networking service
that enables users to send and read short 140-character " +
"messages called 'tweets'. Registered users can read and post tweets, but
those who are unregistered can only read " +
"them. Users access Twitter through the website interface, SMS or mobile
device app Twitter Inc. is based in San " +
"Francisco and has more than 25 offices around the world. Twitter was
created in March 2006 by Jack Dorsey, " +
"Evan Williams, Biz Stone, and Noah Glass and launched in July 2006. The
service rapidly gained worldwide popularity, " +
"with more than 100 million users posting 340 million tweets a day in
2012.The service also handled 1.6 billion " +
"search queries per day.";
public string Content2 = "Facebook is an online social networking service
headquartered in Menlo Park, California. Its website was " +
"launched on February 4, 2004, by Mark Zuckerberg with his Harvard College
roommates and fellow students Eduardo " +
"Saverin, Andrew McCollum, Dustin Moskovitz and Chris Hughes.The founders
had initially limited the website " +
"membership to Harvard students, but later expanded it to colleges in the
Boston area, the Ivy League, and Stanford " +
"University. It gradually added support for students at various other
universities and later to high-school students.";
public string Content3 = "WhatsApp Messenger is a proprietary cross-platform
instant messaging client for smartphones that operates " +
"under a subscription business model. It uses the Internet to send Text
messages, images, vIDeo, user location and " +
"audio media messages to other users using standard cellular mobile numbers.
As of February 2016, WhatsApp had a user " +
"base of up to one billion,[10] making it the most globally popular
messaging application. WhatsApp Inc., based in " +
"Mountain View, California, was acquired by Facebook Inc. on February 19,
2014, for approximately US$19.3 billion.";
public void OnChange(ChangeEventArgs<int, DropdownData> args)
{
Tab.Select(args.Value);
}
public void Selected(Syncfusion.Blazor.Navigations.SelectEventArgs args)
{
this.EventLog = args.IsInteracted ? "Tab Item selected by user interaction"
: "Tab Item selected by programmatically";
}
List<DropdownData> DropdownDatasource = new List<DropdownData> {
new DropdownData() { ID= 0, Text= "Twitter" },
new DropdownData() { ID= 1, Text= "Facebook" },
new DropdownData() { ID= 2, Text= "WhatsApp" },
};
};

```

```
public class DropdownData
{
    public int ID { get; set; }
    public string Text { get; set; }
}
<style>
.eventlog b {
color: #388e3c;
}
</style>
```

TextBox

Getting Started with Blazor TextBox Component

This section briefly explains about how to include a [Blazor TextBox](#) Component in the Blazor Server-Side and Client-Side application. Refer to Getting Started with [Blazor Server-Side TextBox](#) and [Blazor WebAssembly TextBox](#) documentation pages for configuration specifications.

To get start quickly with Blazor TextBox component, check on this video.

{% youtube

"youtube:https://www.youtube.com/watch?v=vzBKF4Gs-Mc"%}

Importing Syncfusion Blazor component in the application

- Install Syncfusion.Blazor.Inputs NuGet package to the application by using the NuGet Package Manager.

Please ensure to check the Include prerelease option for the Beta release.

- The client-side resources can be added through [CDN](#) or from [NuGet](#) package in the **HEAD** element of the ~/wwwroot/index.html page.

HTML

```
<head>
<link href="_content/Syncfusion.Blazor.Themes/bootstrap4.css"
rel="stylesheet" />
<!-- <link
href="https://cdn.syncfusion.com/blazor/{{version}}/styles/{{theme}}.css"
rel="stylesheet" /> -->
</head>
```

For Internet Explorer 11 kindly refer the polyfills. Refer the [documentation](#) for more information.

HTML

```
<head>
<link href="_content/Syncfusion.Blazor.Themes/bootstrap4.css"
rel="stylesheet" />
```

```
<script  
src="https://github.com/Daddoon/Blazor.Polyfill/releases/download/3.0.1/blaz  
or.polyfill.min.js"></script>  
</head>
```

Adding component package to the application

Open `~/_Imports.razor` file and import the `Syncfusion.Blazor.Inputs` packages.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
```

Add `SyncfusionBlazor` service in `Program.cs`

Open the `Program.cs` file and add services required by Syncfusion components using `builder.Services.AddSyncfusionBlazor()` method.

CSHARP

```
using Syncfusion.Blazor;  
namespace BlazorApplication  
{  
    public class Program  
    {  
        public static async Task Main(string[] args)  
        {  
            ....  
            ....  
            builder.Services.AddSyncfusionBlazor();  
            await builder.Build().RunAsync();  
        }  
    }  
}
```

To enable custom client side resource loading from CRG or CDN. You need to disable resource loading by `AddSyncfusionBlazor(true)` and load the scripts in the **HEAD** element of the `~/wwwroot/index.html` page.

HTML

```
<head>  
<script src="https://cdn.syncfusion.com/blazor/{{ site.blazorversion  
}}/syncfusion-blazor.min.js"></script>  
</head>
```

Adding TextBox component to the application

To initialize the TextBox component add the below code to the `Index.razor` view page which is present under `~/Pages` folder.

ASPX-CS

```
<SfTextBox Placeholder='First Name'></SfTextBox>
```

Run the application

After successful compilation of the application, press **F5** to run the application.

First Name

Adding icons to the TextBox

A TextBox can be created with icon as a group by creating the parent `div` element with the class `e-input-group` and add the icon element as `span` with the class `e-input-group-icon`.

ASPX-CS

```
<div class='e-input-group'>
<input class='e-input' Placeholder='Date of Birth' type='text'>
<span class='e-input-group-icon e-input-calendar'></span>
</div>
<style>
.e-input-group-icon:before {
font-family: e-icons;
}
.e-input-group .e-input-group-icon.e-input-calendar {
font-size: 16px;
}
.e-input-group-icon.e-input-calendar:before {
content: "□";
}
</style>
```

Date of Birth



Floating label

The floating label TextBox floats the label above the TextBox after focusing, or filled with value in the TextBox. The floating label TextBox can be created by using the `FloatLabelType` API.

ASPX-CS

```
<SfTextBox Placeholder='First Name'
FloatLabelType='@FloatLabelType.Auto'></SfTextBox>
```

First Name

You can also explore our [Blazor TextBox example](#) that shows how to configure the textbox in Blazor.

See Also

- [Getting Started with Syncfusion Blazor for Client-Side in .NET Core CLI](#)
- [Getting Started with Syncfusion Blazor for Server-side in Visual Studio](#)
- [Getting Started with Syncfusion Blazor for Server-Side in .NET Core CLI](#)

Data Binding in Blazor TextBox Component

This section briefly explains how to bind the value to the TextBox component in the following different ways.

- One-way data binding
- Two-way data binding
- Dynamic value binding
- Complex data binding

One-way binding

The value can be bound to the TextBox component directly for **Value** property as mentioned in the following code example. In one-way binding, pass the property or variable name along with **@** (For Ex: "@Name").

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfTextBox Value="@Name"></SfTextBox>
<button @onclick="@UpdateValue">Update Value</button>
@code {
public string Name { get; set; } = "Hello, World!";
public void UpdateValue()
{
Name = "Hello, Blazor!";
}
}
```

Two-way data binding

Two-way binding can be achieved by using **bind-Value** attribute and its support string, int, Enum, DateTime, and bool types. If the component value has been changed, it will affect all places where you bind the variable for the **bind-value** attribute.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<p>TextBox value is: @Name</p>
<SfTextBox @bind-Value="@Name"></SfTextBox>
@code {
public string Name { get; set; } = "Syncfusion";
}
```

Dynamic value binding

The property value can be changed dynamically by manually calling the **StateHasChanged()** method inside public event of **Blazor TextBox component** only. This method notifies the component that its state has changed and queues a re-render.

There is no need to call this method for native events since it's called after any life cycle method and can also be invoked manually to trigger a re-render.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
```

```
<SfTextBox Placeholder="Enter a Numeric Values" Input="OnInput"
CssClass="@CssClass"></SfTextBox>
@code {
public string CssClass { get; set; }
public void OnInput(InputEventArgs args)
{
if (!System.Text.RegularExpressions.Regex.IsMatch(args.Value, "[0-9]*$")){
CssClass = "e-error";
}
else {
CssClass = "e-success";
}
this.StateHasChanged();
}
}
```

Complex data binding

The complex data values can be bound to the TextBox component. The following code demonstrates how to bind complex data values to the TextBox component.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs;
<SfTextBox Value="@country.data.countryname"></SfTextBox>
@code {
Country country = new Country();
protected override void OnInitialized()
{
country.data = new Data();
}
public class Country
{
public string id { get; set; }
public Data data;
}
public class Data
{
public string countryname { get; set; } = "India";
}
}
```

Multiline in Blazor TextBox Component

This feature allows the textbox to accept one or more lines of text like address, description, comments, and more.

Create multiline textbox

The default textbox can be converted into the multiline textbox by setting the [Multiline](#) API value as true or pass HTML5 textarea as element to the textbox.

The multiline text box allows to resize it in vertical direction alone.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<div class="multiline">
```

```
<SfTextBox Multiline=true Placeholder="Enter your address" Value="Mr.
Dodsworth Dodsworth, System Analyst, Studio 103"></SfTextBox>
</div>
<style>
.multiline{
margin-top: 60px;
width: 20%;
}
</style>
```

Mr. Dodsworth Dodsworth, System
Analyst, Studio 103

Implementing floating label

The floating label behavior can be achieved in the multiline text box by setting [FloatLabelType](#) to 'Auto'. The Placeholder text act as floating label to the multiline textbox. The Placeholder text can be provided to the multiline textbox either by using the [Placeholder](#) property or Placeholder attribute.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<label>Float label type as Auto</label>
<div class="multiline">
<SfTextBox Multiline=true FloatLabelType="@FloatLabelType.Auto"
Placeholder="Enter your address"></SfTextBox>
</div>
<label>Float label type as Always</label>
<div class="multiline">
<SfTextBox Multiline=true FloatLabelType="@FloatLabelType.Always"
Placeholder="Enter your address"></SfTextBox>
</div>
<label>Float label type as Never</label>
<div class="multiline">
<SfTextBox Multiline=true FloatLabelType="@FloatLabelType.Never"
Placeholder="Enter your address"></SfTextBox>
</div>
<style>
.multiline{
margin-top: 60px;
width: 20%;
}
</style>
```

Float label type as auto

Enter your address

Float label type as always

Enter your address

Float label type as never

Enter your address

Disable resizing

By default, the multiline text box is rendered with resizable. The resize of the multiline text box can be disabled by applying the following CSS styles.

CSS

```
textarea.e-input,
.e-float-input textarea,
.e-float-input.e-control-wrapper textarea,
.e-input-group textarea,
.e-input-group.e-control-wrapper textarea {
  resize: none;
}
```

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfTextBox Multiline=true FloatLabelType="@FloatLabelType.Auto"
Placeholder="Enter your address"></SfTextBox>
```

Enter your address

Sizing in Blazor TextBox Component

The TextBox can be rendered in two different sizes:

Property | Description

Normal | By default, the TextBox is rendered with normal size.

Small | Add `e-small` class to the input element, or else add to the input container.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<label>Normal</label>
<SfTextBox Placeholder="Enter text"></SfTextBox>
<label>Float input</label>
<SfTextBox Placeholder="Focus the input"
FloatLabelType="@FloatLabelType.Auto"></SfTextBox>
<label>Small</label>
<SfTextBox Placeholder="Enter text" CssClass="e-small"></SfTextBox>
<label>Float input</label>
<SfTextBox Placeholder="Focus the input" CssClass="e-small"
FloatLabelType="@FloatLabelType.Auto"></SfTextBox>
```

Normal

Enter text

Float input

Focus the input

Small

Enter text

Float input

Focus the input

Groups in Blazor TextBox Component

The following section explains the steps required to create TextBox with icon and floating label.

TextBox:

Create a TextBox component.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfTextBox Placeholder="Enter your name"></SfTextBox>
```

Floating label:

Create a Floating label TextBox using [FloatLabelType](#) API.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfTextBox Placeholder="Enter your name"
FloatLabelType="@FloatLabelType.Auto"></SfTextBox>
```

Refer to the following sections to add the icons to the TextBox.

With icon and floating label

Create a TextBox with icon and the users can place the icon in either side of the TextBox by using **AddIcon** method append the icon before or after the input. Based on the argument prepend or append, it will act as prefix or suffix icon.

ASPX-CS


```
@using Syncfusion.Blazor.Inputs
<div id="wrapper">
<label>TextBox with icon</label>
<SfTextBox @ref="TextBoxDateObj" Placeholder="Enter date"
Created="@OnCreateDate"></SfTextBox>
<SfTextBox @ref="TextBoxSearchObj" Placeholder="Search here"
Created="@OnCreateSearch"></SfTextBox>
<label>Floating TextBox with icon</label>
<SfTextBox @ref="FloatTextBoxDateObj" Placeholder="Enter date"
FloatLabelType="@FloatLabelType.Auto"
Created="@OnFloatCreateDate"></SfTextBox>
<SfTextBox @ref="FloatTextBoxSearchObj" Placeholder="Search here"
FloatLabelType="@FloatLabelType.Auto"
Created="@OnFloatCreateSearch"></SfTextBox>
</div>
@code{
SfTextBox TextBoxDateObj;
SfTextBox TextBoxSearchObj;
SfTextBox FloatTextBoxDateObj;
SfTextBox FloatTextBoxSearchObj;
public void OnCreateDate()
{
this.TextBoxDateObj.AddIconAsync("append", "e-date-icon");
}
public void OnCreateSearch()
{
this.TextBoxSearchObj.AddIconAsync("prepend", "e-search");
}
public void OnFloatCreateDate()
{
this.FloatTextBoxDateObj.AddIconAsync("append", "e-date-icon");
}
public void OnFloatCreateSearch()
{
this.FloatTextBoxSearchObj.AddIconAsync("prepend", "e-search");
}
}
<style>
.e-search::before {
content: '\e993';
font-family: e-icons;
}
#wrapper {
width: 30%;
}
</style>
```


TextBox with icon

Enter date 

 Search here

Floating TextBox with icon

Enter date
01/01/2020 

Search here
 Dodsworth

Binding events to icons

You can bind the event to the icons by passing events as a parameter to the `AddIcon` method. You can bind the single or multiple events to the icons.

The following sample demonstrates binding events to the icons.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<label>Single Event</label>
<SfTextBox @ref="@TextBoxSearchObj" Created="@OnCreateSearch"></SfTextBox>
<label>Multiple Events</label>
<SfTextBox @ref="@TextBoxDateObj" Created="@OnCreateDate"></SfTextBox>
@code {
    SfTextBox TextBoxSearchObj;
    SfTextBox TextBoxDateObj;
    public async Task OnCreateSearch()
    {
        // Event creation with event handler
        var Click = EventCallback.Factory.Create<MouseEventArgs>(this, SearchClick);
        await TextBoxSearchObj.AddIcon("append", "e-search-icon", new
        Dictionary<string, object>() { { "onclick", Click } });
    }
    public void SearchClick()
    {
        // Icon Click event triggered
    }
    public async Task OnCreateDate()
    {
        // Event creation with event handler
        var MouseDown = EventCallback.Factory.Create<MouseEventArgs>(this,
        DateMouseDown);
        var MouseUp = EventCallback.Factory.Create<MouseEventArgs>(this,
        DateMouseUp);
        await TextBoxDateObj.AddIcon("prepend", "e-date-icon", new
        Dictionary<string, object>() { { "onmouseup", MouseUp }, { "onmousedown",
        MouseDown } });
    }
    public void DateMouseDown()
    {
        // Icon mouse down event triggered
    }
    public void DateMouseUp()
    {

```

```
// Icon mouse up event triggered
}
}
<style>
.e-search-icon::before {
content: '\e724';
font-family: e-icons;
}
.e-date-icon::before {
content: '\e901';
font-family: e-icons;
}
</style>
```

With clear button and floating label

The clear button is added to the input for clearing the value given in the TextBox. It is shown only when the input field has a value, otherwise not shown.

The clear button can be added to the TextBox by enabling the [ShowClearButton](#) API.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<label> TextBox with clear button </label>
<SfTextBox Placeholder="FirstName" ShowClearButton=true></SfTextBox>
<label> Floating TextBox with clear button </label>
<SfTextBox Placeholder="FirstName" ShowClearButton=true
FloatLabelType="@FloatLabelType.Auto"></SfTextBox>
```

TextBox with clear button

Andrew X

Floating TextBox with clear button

FirstName
Andrew X

Multi-line input with floating label

The following example demonstrates how to set [Multiline](#) in the `TextBox` component with the float label structure.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfTextBox Placeholder="Enter text" Multiline=true
FloatLabelType="@FloatLabelType.Auto"></SfTextBox>
```

Enter your address
Mr. Dodsworth Dodsworth, System
Analyst, Studio 103, The Business
Center

Validation in Blazor TextBox Component

The TextBox supports three types of validation styles namely error, warning, and success. These states are enabled by adding corresponding classes .e-error, .e-warning, or .e-success to the input parent element.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<label>Success</label>
<SfTextBox Placeholder="Enter your address" CssClass="e-success"
></SfTextBox>
<label>Error</label>
<SfTextBox Placeholder="Enter your address" CssClass="e-error"></SfTextBox>
<label>Warning</label>
<SfTextBox Placeholder="Enter your address" CssClass="e-
warning"></SfTextBox>
```



Native Events in Blazor TextBox Component

The following section explains the steps to include native events and pass data to event handler in textbox component.

Bind native events to textbox

Any native event can be accessed by using on <event> attribute with a component. The attribute's value is treated as an event handler.

In the following example, the KeyPressed method is called every time the key is pressed on textbox.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfTextBox Placeholder='First Name' @onkeypress='@KeyPressed'></SfTextBox>
@code {
public void KeyPressed() {
Console.WriteLine("Key Pressed!");
}
}
```

Also, the above example code can be rewritten as follows using Lambda expressions.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfTextBox Placeholder='First Name' @onkeypress="@(() =>
Console.WriteLine("Key Pressed!"))">
```

```
</SfTextBox>
```

Pass event data to event handler

Blazor provides set of argument types for map to native events. The list of event types and event arguments are:

- Focus Events - FocusEventArgs
- Mouse Events - MouseEventArgs
- Keyboard Events - KeyboardEventArgs
- Input Events - ChangeEventArgs/EventArgs
- Touch Events – TouchEventArgs
- Pointer Events – PointerEventArgs

In the following example, the KeyPressed method is called every time any key is pressed inside textbox. But the message will be printed when the "s" key is pressed.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfTextBox Placeholder='First Name' @onkeypress='@(e => KeyPressed(e))'
></SfTextBox>
@code {
public void KeyPressed(KeyboardEventArgs args) {
if (args.Key == "s") {
Console.WriteLine("S was pressed");
}
}
}
```

Using Lambda expression also, the event data can be passed to the event handler.

List of Native events supported

| List of Native events | | |

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

| onclick | onblur | onfocus | onfocusout |

| onmousemove | onmouseover | onmouseout | onmousedown | onmouseup |

| ondblclick | onkeydown | onkeyup | onkeypress |

| ontouchend | onfocusin | onmouseup | ontouchstart |

Events in Blazor TextBox Component

This section explains the list of events of the TextBox component which will be triggered for appropriate TextBox actions.

Blur

Blur event triggers when the TextBox has focus-out.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfTextBox Blur="@BlurHandler"></SfTextBox>
```

```
@code {  
private void BlurHandler(FocusEventArgs args)  
{  
    // Here you can customize your code  
}  
}
```

Created

Created event triggers when the TextBox component is created.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs  
<SfTextBox Created="@CreatedHandler"></SfTextBox>  
@code {  
private void CreatedHandler(Object args)  
{  
    // Here you can customize your code  
}  
}
```

Destroyed

Destroyed event triggers when the TextBox component is destroyed.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs  
<SfTextBox Destroyed="@DestroyedHandler"></SfTextBox>  
@code {  
private void DestroyedHandler(Object args)  
{  
    // Here you can customize your code  
}  
}
```

Focus

Focus event triggers when the TextBox gets focus.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs  
<SfTextBox Focus="@FocusHandler"></SfTextBox>  
@code {  
private void FocusHandler(FocusEventArgs args)  
{  
    // Here you can customize your code  
}  
}
```

Input

Input event triggers each time when the value of TextBox has changed.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfTextBox Input="@InputHandler"></SfTextBox>
@code {
private void InputHandler(InputEventArgs args)
{
// Here you can customize your code
}
}
```

ValueChange

ValueChange event triggers when the content of TextBox has changed and gets focus-out.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfTextBox ValueChange="@ValueChangeHandler"></SfTextBox>
@code {
private void ValueChangeHandler(ChangedEventArgs args)
{
// Here you can customize your code
}
}
```

ValueChanged

ValueChanged event specifies the callback to trigger when the value changes.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfTextBox ValueChanged="@ValueChangedHandler"></SfTextBox>
@code {
private void ValueChangedHandler(String args)
{
// Here you can customize your code
}
}
```

TextBox is limited with these events and new events will be added in the future based on the user requests. If the event you are looking for is not on the list, then please request [here](#).

How To

Set the Rounded Corner in Blazor TextBox Component

Render the TextBox with **rounded corner** by adding the **e-corner** class to the input parent element.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfTextBox Placeholder='First Name' CssClass="e-corner"></SfTextBox>
<style>
.e-input-group.e-corner {
border-radius: 4px;
}
</style>
```


Set the Disabled State in Blazor TextBox Component

To disable the TextBox, use its [Enabled](#) property.

The following example demonstrates the TextBox in a disabled state.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfTextBox Placeholder='First Name' Enabled=false></SfTextBox>
```

First Name

Set the Read-only TextBox in Blazor TextBox Component

The following example demonstrates how to set [Readonly](#) in TextBox Component. This can be achieved by using [Readonly](#) property.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<SfTextBox Placeholder='First Name' Readonly=true></SfTextBox>
```

First Name

Change the Floating Label Color of the Blazor TextBox Component

The floating label color of the TextBox can be changed for both [success](#) and [warning](#) validation states by applying the following CSS styles.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
<div class="e-float-input @(TextClass) e-success">
  <input type="text" @onfocus="@Focus" @onblur="@Blur" required />
  <span class="e-float-line"></span>
  <label class="e-float-text">Success</label>
</div>
<div class="e-float-input @(FloatTextClass) e-warning">
  <input type="text" @onfocus="@FlaotFocus" @onblur="@FloatBlur" required />
  <span class="e-float-line"></span>
  <label class="e-float-text">Warning</label>
</div>
@code {
  public string FocusClass { get; set; } = " e-input-focus";
  public string TextClass { get; set; } = "e-input-group";
  public string FloatTextClass { get; set; } = "e-input-group";
  public void Focus(FocusEventArgs args)
  {
    this.TextClass = this.TextClass + FocusClass;
    StateHasChanged();
  }
  public void FlaotFocus(FocusEventArgs args)
  {
    this.FloatTextClass = this.FloatTextClass + FocusClass;
```

```

StateHasChanged();
}
public void Blur(FocusEventArgs args)
{
    if (this.TextClass.Contains(FocusClass))
    {
        this.TextClass = this.TextClass.Replace(FocusClass, "");
    }
    StateHasChanged();
}
public void FloatBlur(FocusEventArgs args)
{
    if (this.FloatTextClass.Contains(FocusClass))
    {
        this.FloatTextClass = this.FloatTextClass.Replace(FocusClass, "");
    }
    StateHasChanged();
}
}
<style>
.e-float-input.e-success label.e-float-text,
.e-float-input.e-success input:focus ~ label.e-float-text,
.e-float-input.e-success input:valid ~ label.e-float-text {
    color: #22b24b;
}
/* For Warning state */
.e-float-input.e-warning label.e-float-text,
.e-float-input.e-warning input:focus ~ label.e-float-text,
.e-float-input.e-warning input:valid ~ label.e-float-text {
    color: #ffca00;
}
</style>

```

Success

Warning

Change the Color of the Text Based on its Value in Blazor TextBox

The color of the TextBox can be changed by validating its value using regular expression in the **Input** event for predicting the numeric values as demonstrated in the following code sample.

ASPX-CS

```

@using Syncfusion.Blazor.Inputs
<SfTextBox Placeholder="Enter a Numeric Values"
FloatLabelType="@FloatLabelType.Auto" Input="OnInput"
CssClass="@CssClass"></SfTextBox>
@code {
    public string CssClass { get; set; }
    public void OnInput(InputEventArgs args)
    {
        if (!System.Text.RegularExpressions.Regex.IsMatch(args.Value, "^[0-9]*$")) {
            CssClass = "e-error";
        }
    }
}

```

```

}
else {
  CssClass = "e-success";
}
this.StateHasChanged();
}
}

```

Enter a Numeric Values

a

Customize Background and Text Color in Blazor TextBox Component

The text box styles can be customized such as background-color, text-color and border-color by overriding its default styles.

ASPX-CS

```

@using Syncfusion.Blazor.Inputs
<div class="@ (TextClass)">
  <div class="e-input-in-wrap">
    <input class="e-input" type="text" Placeholder="Enter Date" Value="John"
    @onfocus="@Focus" @onblur="@Blur" />
    <span class="e-input-group-icon e-input-date"></span>
  </div>
</div>
@code {
  public string FocusClass { get; set; } = " e-input-focus";
  public string TextClass { get; set; } = "e-input-group";
  public void Focus(FocusEventArgs args)
  {
    this.TextClass = this.TextClass + FocusClass;
    StateHasChanged();
  }
  public void Blur(FocusEventArgs args)
  {
    if (this.TextClass.Contains(FocusClass))
    {
      this.TextClass = this.TextClass.Replace(FocusClass, "");
    }
    StateHasChanged();
  }
}
<style>
.e-input-group {
  background: yellow;
  color: red;
}
</style>

```

John

Create TextBox component dynamically in Blazor TextBox Component

The TextBox component can be rendered at runtime in the following ways:

1. Using RenderTreeBuilder
2. Using RenderFragment

Dynamic rendering using RenderTreeBuilder

The RenderTreeBuilder class will let to create required content or component in dynamic manner at runtime. In the following code example, the TextBox Component has been created at runtime through button click.

ASPX-CS

```
@using Microsoft.AspNetCore.Components;
@using Syncfusion.Blazor.Buttons;
@using Syncfusion.Blazor.Inputs;
<div id="component-container">
@DynamicRender
</div>
<SfButton ID="dynamic-button" Content="Render TextBox"
@onclick="RenderComponent"></SfButton>
@code {
private RenderFragment DynamicRender { get; set; }
private RenderFragment CreateComponent() => builder =>
{
builder.OpenComponent(0, typeof(SfTextBox));
builder.AddAttribute(1, "Placeholder", "Enter your text");
builder.CloseComponent();
};
private void RenderComponent()
{
DynamicRender = CreateComponent();
}
}
```

Render TextBox

Enter your text

Dynamic rendering using RenderFragment

By using RenderFragment, the templated component can be reused in more than one place. The specific fragment value alone can be changed without re-rendering the entire component. In the following example, a single text box has been created and updated the placeholder value at runtime.

CSHARP

```
@using Microsoft.AspNetCore.Components;
@using Syncfusion.Blazor.Buttons;
@using Syncfusion.Blazor.Inputs;
```

```

<SfButton ID="dynamic-button" Content="Render TextBox"
@onclick="ChangeAttribute"></SfButton>
<div id="component-container">
@DynamicFragment
</div>
@code {
private string dynamicContent = "Type here";
protected override void OnInitialized()
{
DynamicFragment = builder =>
{
builder.OpenComponent(0, typeof(SfTextBox));
builder.AddAttribute(1, "Placeholder", dynamicContent);
builder.CloseComponent();
};
}
private void ChangeAttribute()
{
dynamicContent = "Enter your text";
}
private Microsoft.AspNetCore.Components.RenderFragment DynamicFragment;
}

```

Render TextBox

Enter your text

TimePicker

Getting Started with Blazor TimePicker Component

This section briefly explains about how to include a [Blazor TimePicker](#) component in the Blazor Server-Side and Client-Side application. Refer to the Getting Started with [Blazor Server-Side TimePicker](#) and [Blazor WebAssembly TimePicker](#) documentation pages for configuration specifications.

To get start quickly with Blazor TimePicker component, check on this video.

{% youtube

"youtube:https://www.youtube.com/watch?v=X6RsJJLIY"%}

Importing Syncfusion Blazor component in the application

- Install **Syncfusion.Blazor.Calendars** NuGet package to the application by using the **NuGet Package Manager**.

Please ensure to check the **Include prerelease** option for the Beta release.

- The client-side resources can be added through [CDN](#) or from [NuGet](#) package in the **HEAD** element of the `~/wwwroot/index.html` page.

HTML

```
<head>
<link href="_content/Syncfusion.Blazor.Themes/bootstrap4.css"
rel="stylesheet" />
<!-- <link
href="https://cdn.syncfusion.com/blazor/{{version}}/styles/{{theme}}.css"
rel="stylesheet" /> -->
</head>
```

For Internet Explorer 11 kindly refer the polyfills. Refer the [documentation](#) for more information.

HTML

```
<head>
<link href="_content/Syncfusion.Blazor.Themes/bootstrap4.css"
rel="stylesheet" />
<script
src="https://github.com/Daddoon/Blazor.Polyfill/releases/download/3.0.1/blaz
or.polyfill.min.js"></script>
</head>
```

Adding component package to the application

Open `~/_Imports.razor` file and import the `Syncfusion.Blazor.Calendars` package.

ASPX-CS

```
@using Syncfusion.Blazor.Calendars
```

Add SyncfusionBlazor service in Program.cs

Open the **Program.cs** file and add services required by Syncfusion components using **builder.Services.AddSyncfusionBlazor()** method.

C#

```
using Syncfusion.Blazor;
namespace BlazorApplication
{
    public class Program
    {
        public static async Task Main(string[] args)
        {
            ....
            ....
            builder.Services.AddSyncfusionBlazor();
            await builder.Build().RunAsync();
        }
    }
}
```

To enable custom client side resource loading from CRG or CDN. You need to disable resource loading by `AddSyncfusionBlazor(true)` and load the scripts in the **HEAD** element of the `~/wwwroot/index.html` page.

HTML

```
<head>
<script src="https://cdn.syncfusion.com/blazor/{ site.blazorversion
  }/syncfusion-blazor.min.js"></script>
</head>
```

Adding TimePicker component to the application

To initialize the TimePicker component add the below code to the `Index.razor` view page which is present under `~/Pages` folder.

The following code shows a basic TimePicker component.

ASPX-CS

```
<SfTimePicker TValue="DateTime?" Placeholder="Select a time"></SfTimePicker>
```

Run the application

After successful compilation of the application, press **F5** to run the application.



Setting the time format

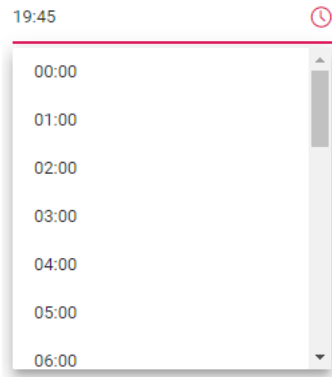
Time format is a way of representing the time value in different string format in textbox and popup list. By default, the TimePicker's Format is based on the culture.

But the Format of the TimePicker can be customized using the [Format](#) property.

The below code demonstrates how to render TimePicker component in 24 hours Format with 60 minutes interval. The time interval is set to 60 minutes by using the [Step](#) property.

ASPX-CS

```
<SfTimePicker TValue="DateTime?" Value="@TimeValue" Step=60
Format="HH:mm"></SfTimePicker>
@code {
public DateTime TimeValue { get; set; } = DateTime.Now;
}
```



See Also

- [Getting Started with Syncfusion Blazor for Client-Side in .NET Core CLI](#)
- [Getting Started with Syncfusion Blazor for Server-side in Visual Studio](#)
- [Getting Started with Syncfusion Blazor for Server-Side in .NET Core CLI](#)

Data Binding in Blazor TimePicker Component

This section briefly explains how to bind the value to the TimePicker component in the below different ways.

- One-Way Data Binding
- Two-Way Data Binding
- Dynamic Value Binding

One-Way Binding

We can bind the value to the TimePicker component directly for **Value** property as mentioned in the following code example. In one-way binding, we need to pass property or variable name along with **@** (For Ex: "@DateValue").

ASPX-CS

```
@using Syncfusion.Blazor.Calendars
<SfTimePicker TValue="DateTime?" Value="@DateValue"></SfTimePicker>
<button @onclick="@UpdateValue">Update Value</button>
@code {
    public DateTime DateValue { get; set; } = new DateTime(DateTime.Now.Year,
    DateTime.Now.Month, 28, 12, 30, 00);
    public void UpdateValue()
    {
        DateValue = DateTime.Now;
    }
}
```

Two-Way Data Binding

Two-way binding can be achieved by using **bind-Value** attribute and its support string, int, Enum, DateTime, bool types. If component value has been changed, it will affect all the places where we bind the variable for the **bind-value** attribute.

ASPX-CS


```
@using Syncfusion.Blazor.Calendars
<p>TimePicker value is: @DateValue</p>
<SfTimePicker TValue="DateTime?" @bind-Value="@DateValue"></SfTimePicker>
@code {
public DateTime? DateValue { get; set; } = DateTime.Now;
}
```

Dynamic Value Binding

The property value can be changed dynamically by manually calling the `StateHasChanged()` method inside public event of **Blazor TimePicker component** only. This method notifies the component that its state has changed and queues a re-render.

There is no need to call this method for native events since it's called after any lifecycle method has been called and can also be invoked manually to trigger a re-render.

ASPX-CS

```
@using Syncfusion.Blazor.Calendars
<p>TimePicker value is: @TimeValue</p>
<SfTimePicker TValue="DateTime?" Value="@TimeValue">
<TimePickerEvents TValue="DateTime?"
ValueChange="@onChange"></TimePickerEvents>
</SfTimePicker>
@code {
public DateTime? TimeValue { get; set; } = DateTime.Now;
private void onChange(Syncfusion.Blazor.Calendars.ChangeEventArgs<DateTime?>
args)
{
TimeValue = args.Value;
StateHasChanged();
}
}
```

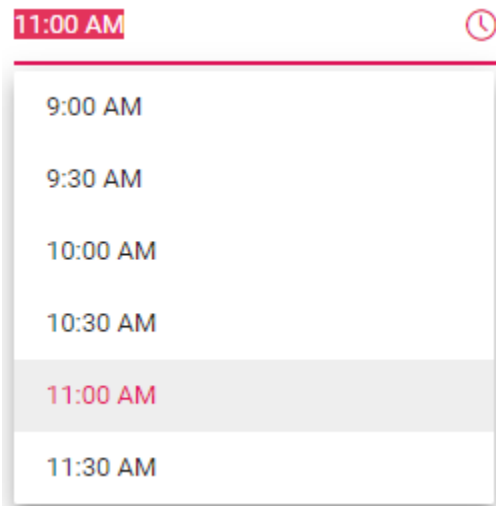
Time Range in Blazor TimePicker Component

TimePicker provides an option to select a time value within a specified range by using the [Min](#) and [Max](#) properties. The Min value should always be lesser than the Max value. The `Value` property depends on the Min/Max with respect to [StrictMode](#) property.

The following code allows to select a time value within a range of **9:00 AM** to **11:30 AM**. For more information about StrictMode, refer to the [Strict Mode](#) section from the documentation.

ASPX-CS

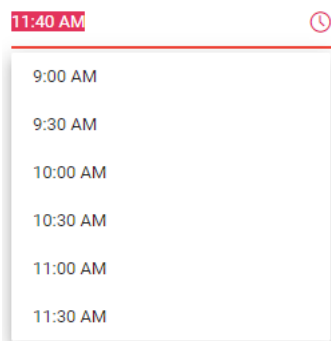
```
@using Syncfusion.Blazor.Calendars
<SfTimePicker TValue="DateTime?" Value='@TimeValue' Min='@MinVal'
Max='@MaxVal'></SfTimePicker>
@code{
public DateTime MinVal { get; set; } = new DateTime(DateTime.Now.Year,
DateTime.Now.Month, 15, 09, 00, 00);
public DateTime MaxVal { get; set; } = new DateTime(DateTime.Now.Year,
DateTime.Now.Month, 15, 11, 30, 00);
public DateTime? TimeValue { get; set; } = new DateTime(DateTime.Now.Year,
DateTime.Now.Month, 15, 11, 00, 00);
}
```



When the **Min** and **Max** properties are configured and the selected time value is out-of-range or invalid, then the model value will be set to **out of range** time value or **null** respectively with highlighted **error** class to indicate that the time is out of range or invalid.

ASPX-CS

```
@using Syncfusion.Blazor.Calendars
<SfTimePicker TValue="DateTime?" Value="@TimeValue" Min="@MinVal"
Max="@MaxVal"></SfTimePicker>
@code{
    public DateTime MinVal { get; set; } = new DateTime(DateTime.Now.Year,
    DateTime.Now.Month, 15, 09, 00, 00);
    public DateTime MaxVal { get; set; } = new DateTime(DateTime.Now.Year,
    DateTime.Now.Month, 15, 11, 30, 00);
    public DateTime? TimeValue { get; set; } = new DateTime(DateTime.Now.Year,
    DateTime.Now.Month, 15, 11, 40, 00);
}
```



If the value of **Min** or **Max** property is changed through code behind, you have to update the **Value** property to set within the range.

Globalization in Blazor TimePicker Component

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**.

Blazor server side

Add **UseRequestLocalization** middle-ware in Configure method in **Startup.cs** file to get browser Culture Info.

Refer the following code to add configuration in Startup.cs file

CSHARP

```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Localization;
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            app.UseRequestLocalization();
            ....
            ....
        }
    }
}
```

The **Localization** library allows to localize default text content. The TimePicker component has static text that can be changed to other cultures (Arabic, Deutsch, French, etc.).

In the following examples, demonstrate how to enable **Localization** for TimePicker in server side Blazor samples. Here, Resource file is used to translate the static text.

The Resource file is an XML file which contains the strings(key and value pairs) that has to be translated into different language. Refer Localization [link](#) to know more about how to configure and use localization in the ASP.NET Core application framework.

- Open the **Startup.cs** file and add the below configuration in the **ConfigureServices** function as follows.

CSHARP

```
using Syncfusion.Blazor;
using System.Globalization;
using Microsoft.AspNetCore.Localization;
namespace BlazorApplication
{
    public class Startup
    {
        {
            ....
            ....
        }
    }
}
```

```

public void ConfigureServices(IServiceCollection services)
{
    ....
    ....
    services.AddSyncfusionBlazor();
    services.AddLocalization(options => options.ResourcesPath = "Resources");
    services.Configure<RequestLocalizationOptions>(options =>
    {
        // define the list of cultures your app will support
        var supportedCultures = new List<CultureInfo>()
        {
            new CultureInfo("de")
        };
        // set the default culture
        options.DefaultRequestCulture = new RequestCulture("de");
        options.SupportedCultures = supportedCultures;
        options.SupportedUICultures = supportedCultures;
        options.RequestCultureProviders = new List<IRequestCultureProvider>() {
            new QueryStringRequestCultureProvider() // Here, you can also use other
            localization provider
        };
    });
    services.AddSingleton(typeof(ISyncfusionStringLocalizer),
        typeof(SampleLocalizer));
}
}
}

```

- Then, write a **class** by inheriting **ISyncfusionStringLocalizer** interface and override the **Manager** property to get the resource file details from the application end.

C#

```

using Syncfusion.Blazor;
namespace blazorCalendars
{
    public class SampleLocalizer : ISyncfusionStringLocalizer
    {
        public string Get(string key)
        {
            return this.Manager.GetString(key);
        }
        public System.Resources.ResourceManager Manager
        {
            get
            {
                return blazorCalendars.Resources.SyncfusionBlazorLocale.ResourceManager;
            }
        }
    }
}

```

- Add **.resx** file to Resource folder and enter the key value (Locale Keywords) in the **Name** column and the translated string in the Value column as follows.

Name	Value (in Deutsch culture)
---	---
TimePicker_Placeholder	Wähle eine Zeit

- Finally, specify the culture for TimePicker using **locale** property.

ASPX-CS

```
@using Syncfusion.Blazor.Calendars
<SfTimePicker TValue="DateTime?" Locale="de"></SfTimePicker>
```

Blazor WebAssembly

By default, the TimePicker week and month names are specific to the **American English** culture. It utilizes the **Blazor Internationalization** package to parse and format the date object based on the culture by using the official [UNICODE CLDR](#) JSON data.

The following steps explain how to render the TimePicker in German culture ('de-DE') in Blazor Web Assembly application.

- Open the **program.cs** file and add the below configuration in the **Builder ConfigureServices** function as follows.

CSHARP

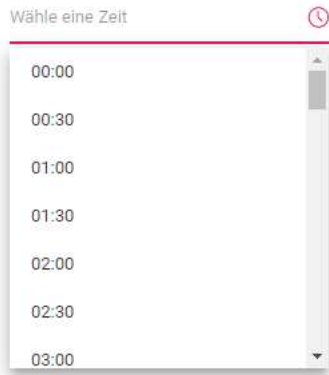
```
using Syncfusion.Blazor;
using Microsoft.AspNetCore.Builder;
namespace WebAssemblyLocale
{
    public class Program
    {
        public static async Task Main(string[] args)
        {
            ....
            ....
            builder.Services.Configure<RequestLocalizationOptions>(options =>
            {
                // Define the list of cultures your app will support
                var supportedCultures = new List<System.Globalization.CultureInfo>()
                {
                    new System.Globalization.CultureInfo("en-US"),
                    new System.Globalization.CultureInfo("de"),
                };
                // Set the default culture
                options.DefaultRequestCulture = new
                Microsoft.AspNetCore.Localization.RequestCulture("de");
                options.SupportedCultures = supportedCultures;
                options.SupportedUICultures = supportedCultures;
                options.RequestCultureProviders = new
                List<Microsoft.AspNetCore.Localization.IRequestCultureProvider>() {
```

```
new Microsoft.AspNetCore.Localization.QueryStringRequestCultureProvider()
};
});
....
....
}
}
}
```

- Download the required locale packages to render the Blazor TimePicker component with specified locale.
- To download the locale definition of Blazor components, use this [link](#).
- After downloading the blazor-locale package, copy the blazor-locale folder with required local definition file into wwwroot folder.
- By default, the blazor-locale package contains the localized text for static text present in components like button text, placeholder, tooltip, and more.
- Set the culture by using the SetCulture method.

ASPX-CS

```
@using Syncfusion.Blazor.Calendars
@inject HttpClient Http;
<SfTimePicker TValue="DateTime?" Locale="de"></SfTimePicker>
@code {
    [Inject]
    protected IJSRuntime JsRuntime { get; set; }
    protected override async Task OnInitializedAsync()
    {
        this.JsRuntime.Sf().LoadLocaleData(await Http.GetJsonAsync<object>("blazor-locale/src/de.json")).SetCulture("de");
    }
}
```



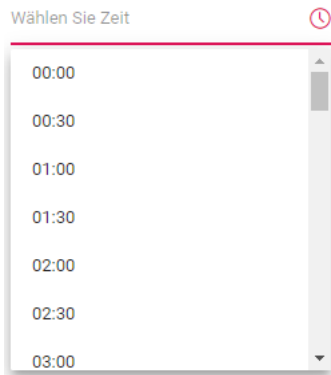
Customize the localized text

- The localized text of particular component can be changed by editing the wwwroot/blazor-locale/src/{{locale name}}.json file.
- In the following code, modified the localized text of placeholder in de culture.

[wwwroot/blazor-locale/src/de.json]

CSHARP

```
{
  "de": {
    "timepicker": {
      "placeholder": "Wählen Sie Zeit"
    }
  }
}
```



Right-To-Left

The TimePicker supports RTL (right-to-left) functionality for languages like Arabic and Hebrew to display the text in the right-to-left direction. Use the [EnableRtl](#) property to set the RTL direction.

The following code example initializes the TimePicker component in Arabic culture.

ASPX-CS

```
@using Syncfusion.Blazor.Calendars
@inject HttpClient Http;
<SfTimePicker TValue="DateTime?" Locale="ar" EnableRtl=true></SfTimePicker>
@code {
    [Inject]
    protected IJSRuntime JsRuntime { get; set; }
    protected override async Task OnInitializedAsync()
    {
        this.JsRuntime.Sf().LoadLocaleData(await Http.GetJsonAsync<object>("blazor-locale/src/ar.json")).SetCulture("ar");
    }
}
```



Native Events in Blazor TimePicker Component

The following section explains steps to include native events and pass data to event handler in TimePicker component.

Bind native events to TimePicker

Any native event can be accessed by using on `<event>` attribute with a component. The attribute's value is treated as an event handler.

In the following example, the KeyPressed method is called every time the key is pressed on input.

ASPX-CS

```
@using Syncfusion.Blazor.Calendars
<SfTimePicker TValue="DateTime?" @onkeypress="@KeyPressed"></SfTimePicker>
@code {
    public void KeyPressed() {
        Console.WriteLine("Key Pressed!");
    }
}
```

Also, the above example code can be rewritten as follows using Lambda expressions.

ASPX-CS

```
@using Syncfusion.Blazor.Calendars
<SfTimePicker TValue="DateTime?" @onkeypress="@(() => Console.WriteLine("Key Pressed!"))"></SfTimePicker>
```

Pass event data to event handler

Blazor provides set of argument types for map to native events. The list of event types and event arguments are:

- Focus Events - FocusEventArgs
- Mouse Events - MouseEventArgs
- Keyboard Events - KeyboardEventArgs
- Input Events - ChangeEventArgs/EventArgs
- Touch Events – TouchEventArgs

- Pointer Events – PointerEventArgs

In the following example, the KeyPressed method is called every time any key is pressed inside input. But the message will be printed when you press "5" key.

ASPX-CS

```
@using Syncfusion.Blazor.Calendars
<SfTimePicker TValue="DateTime?" @onkeypress='@(e =>
KeyPressed(e))'></SfTimePicker>
@code {
public void KeyPressed(KeyboardEventArgs args)
{
if (args.Key == "5")
{
Console.WriteLine("5 was pressed");
}
}
}
```

Using Lambda expression also, the event data can be passed to the event handler.

List of Native events supported

| List of Native events | | |

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

| onclick | onblur | onfocus | onfocusout |

| onmousemove | onmouseover | onmouseout | onmousedown | onmouseup |

| ondblclick | onkeydown | onkeyup | onkeypress |

| ontouchend | onfocusin | onmouseup | ontouchstart |

Strict Mode in Blazor TimePicker Component

The [StrictMode](#) is an act that allows to enter only valid time value within the specified Min/Max range in the textbox. If the value is invalid, the component value sets to the previous value. If the value is out of range, the component sets the time value to Min/Max value.

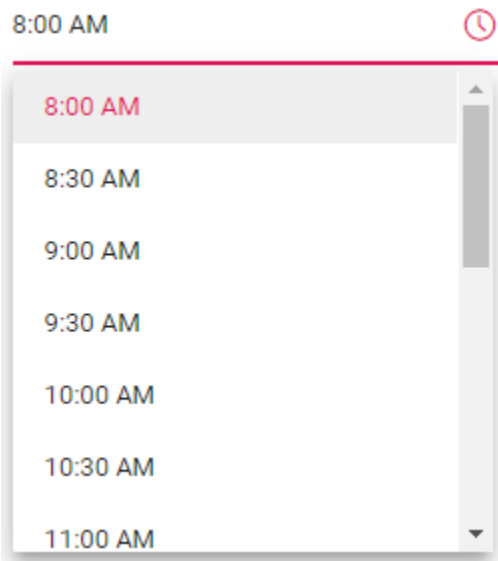
The following example demonstrates the TimePicker in **StrictMode** with **Min/Max** range of **10:00 AM** to **4:00 PM** . It allows to enter only valid time within the specified range.

- If you enter the out-of-range value like **8:00 PM**, the value sets to the Max time **4:00 PM** as the value **8:00 PM** is greater than Max value of **4:00 PM**.
- If you enter invalid time value like **9:00 tt**, the value sets to the previous value.

ASPX-CS

```
@using Syncfusion.Blazor.Calendars
<SfTimePicker TValue="DateTime?" Value='@TimeValue' StrictMode=true
Min='@MinVal' Max='@MaxVal'></SfTimePicker>
@code {
public DateTime MinVal { get; set; } = new DateTime(DateTime.Now.Year,
DateTime.Now.Month, 15, 08, 00, 00);
```

```
public DateTime MaxVal { get; set; } = new DateTime(DateTime.Now.Year,
DateTime.Now.Month, 15, 16, 00, 00);
public DateTime? TimeValue { get; set; } = new DateTime(DateTime.Now.Year,
DateTime.Now.Month, 15, 3, 00, 00);
}
```



By default, the TimePicker act in `StrictMode` as `false` state, that allows to enter the invalid or out-of-range time in textbox.

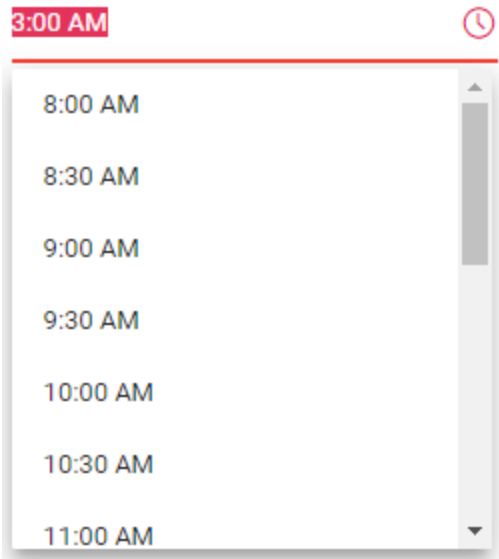
If the time is out-of-range or invalid, then the model value will be set to `out of range` time value or `null` respectively with highlighted `error` class to indicates the time is out of range or invalid.

The following example demonstrates the `StrictMode` as `false`. Here, it allows to enter the valid or invalid value in text box.

- If you are entering the out-of-range or invalid time value, then the model value will be set to `out of range` time value or `null` respectively with highlighted `error` class to indicate that the time is out of range or invalid.

ASPX-CS

```
@using Syncfusion.Blazor.Calendars
<SfTimePicker TValue="DateTime?" Value="@TimeValue" StrictMode=false
Min="@MinVal" Max="@MaxVal"></SfTimePicker>
@code {
public DateTime MinVal { get; set; } = new DateTime(DateTime.Now.Year,
DateTime.Now.Month, 15, 08, 00, 00);
public DateTime MaxVal { get; set; } = new DateTime(DateTime.Now.Year,
DateTime.Now.Month, 15, 16, 00, 00);
public DateTime? TimeValue { get; set; } = new DateTime(DateTime.Now.Year,
DateTime.Now.Month, 15, 3, 00, 00);
}
```



If the value of `Min` or `Max` property is changed through code behind, update the `Value` property to set within the range.

Accessibility in Blazor TimePicker Component

The web accessibility makes web applications and its content more accessible to people with disabilities without any barriers. It especially tracks the dynamic value changes and DOM changes.

The TimePicker component has covered the [WAI-ARIA](#) specifications with the following list of WAI-ARIA attributes: `aria-haspopup`, `aria-selected`, `aria-disabled`, `aria-activedescendant`, `aria-expanded`, `aria-owns`, and `aria-autocomplete`.

In the TimePicker, the `combobox` plays the role of input element, and the `listbox` plays the role of popup element.

- **aria-haspopup**: Provides the information about whether this element display a pop-up window or not.
- **aria-selected**: Indicates the current selected value of the TimePicker component.
- **aria-disabled**: Indicates disabled state of the TimePicker component.
- **aria-expanded**: Indicates the expanded state of the popup.
- **aria-autocomplete**: Indicates whether user input completion suggestions are provided or not.
- **aria-owns**: Creates a parent/child relationship between two DOM element in the accessibility layer.
- **aria-activedescendent**: Helps in managing the current active child of the TimePicker component.
- **role**: Gives assistive technology information for handling each element in a widget.

Keyboard interaction

Keyboard accessibility is one of the most important aspects of web accessibility. Disabled people like blind and those who have motor disabilities or birth defects use keyboard shortcuts more than the mouse.

The TimePicker component has built-in keyboard accessibility support by following the [WAI-ARIA practices](#).

It supports the following list of shortcut keys to interact with the TimePicker component:

Keys	Description
--- ---	
Upper Arrow	Navigates and selects the previous item.
Down Arrow	Navigates and selects the next item.
Left Arrow	Moves the cursor towards arrow key pressed direction.
Right Arrow	Moves the cursor towards arrow key pressed direction.
Home	Navigates and selects the first item.
End	Navigates and selects the last item.
Enter	Selects the currently focused item and close the popup.
Alt + Upper Arrow	Closes the popup.
Alt + Down Arrow	Opens the popup.
Esc	Closes the popup.

To focusout the TimePicker component, use the **t** keys. For additional information about native event, [click](#) here.

ASPX-CS

```
@using Syncfusion.Blazor.Calendars
<SfTimePicker TValue="DateTime?" @onkeypress="@ (e => KeyPressed(e))"
@ref="TimeObj"></SfTimePicker>
@code {
public SfTimePicker<DateTime?> TimeObj;
public void KeyPressed(KeyboardEventArgs args)
{
if (args.Key == "t")
{
this.TimeObj.FocusOutAsync();
}
}
}
```

Events in Blazor TimePicker Component

This section explains the list of events of the TimePicker component which will be triggered for appropriate TimePicker actions.

Event Name(v17.1.) /Event Name(v17.2.)
----- -----
change ValueChange
close OnClose
itemRender OnItemRender
open OnOpen

Blur

Blur event triggers when the input loses the focus.

ASPX-CS

```
@using Syncfusion.Blazor.Calendars
<SfTimePicker TValue="DateTime?">
  <TimePickerEvents TValue="DateTime?" Blur="BlurHandler"></TimePickerEvents>
</SfTimePicker>
@code{
public void BlurHandler(Syncfusion.Blazor.Calendars.BlurEventArgs args)
{
  // Here you can customize your code
}
}
```

ValueChanged

ValueChanged event triggers when the Calendar value is changed.

ASPX-CS

```
@using Syncfusion.Blazor.Calendars
<SfTimePicker TValue="DateTime?">
  <TimePickerEvents TValue="DateTime?"
  ValueChange="ValueChangedHandler"></TimePickerEvents>
</SfTimePicker>
@code{
public void
ValueChangedHandler(Syncfusion.Blazor.Calendars.ChangeEventArgs<DateTime?>
args)
{
  // Here you can customize your code
}
}
```

OnClose

OnClose event triggers when popup is closed.

ASPX-CS

```
@using Syncfusion.Blazor.Calendars
<SfTimePicker TValue="DateTime?">
  <TimePickerEvents TValue="DateTime?"
  OnClose="OnCloseHandler"></TimePickerEvents>
</SfTimePicker>
@code{
public void OnCloseHandler(Syncfusion.Blazor.Calendars.PopupEventArgs args)
{
  // Here you can customize your code
}
}
```

Created

Created event triggers when the component is created.

ASPX-CS

```
@using Syncfusion.Blazor.Calendars
<SfTimePicker TValue="DateTime?">
  <TimePickerEvents TValue="DateTime?"
    Created="CreatedHandler"></TimePickerEvents>
</SfTimePicker>
@code{
public void CreatedHandler(object args)
{
    // Here you can customize your code
}
}
```

Destroyed

Destroyed event triggers when the component is destroyed.

ASPX-CS

```
@using Syncfusion.Blazor.Calendars
<SfTimePicker TValue="DateTime?">
  <TimePickerEvents TValue="DateTime?"
    Destroyed="DestroyHandler"></TimePickerEvents>
</SfTimePicker>
@code{
public void DestroyHandler(object args)
{
    // Here you can customize your code
}
}
```

Focus

Focus event triggers when the input gets focus.

ASPX-CS

```
@using Syncfusion.Blazor.Calendars
<SfTimePicker TValue="DateTime?">
  <TimePickerEvents TValue="DateTime" Focus="FocusHandler"></TimePickerEvents>
</SfTimePicker>
@code{
public void FocusHandler(Syncfusion.Blazor.Calendars.FocusEventArgs args)
{
    // Here you can customize your code
}
}
```

OnItemRender

OnItemRender event triggers while rendering the each popup list item.

ASPX-CS

```
@using Syncfusion.Blazor.Calendars
<SfTimePicker TValue="DateTime?">
```

```

<TimePickerEvents TValue="DateTime?"
OnItemRender="OnItemRenderHandler"></TimePickerEvents>
</SfTimePicker>
@code{
public void
OnItemRenderHandler(Syncfusion.Blazor.Calendar.ItemEventArgs<DateTime?>
args)
{
// Here you can customize your code
}
}

```

OnOpen

OnOpen event triggers when the popup is opened.

ASPX-CS

```

@using Syncfusion.Blazor.Calendar
<SfTimePicker TValue="DateTime?">
<TimePickerEvents TValue="DateTime?"
OnOpen="OpenHandler"></TimePickerEvents>
</SfTimePicker>
@code{
public void OpenHandler(Syncfusion.Blazor.Calendar.PopupEventArgs args)
{
// Here you can customize your code
}
}

```

TimePicker will be limited with these events and new events will be added in future based on the user requests. If the event you are looking for is not in the list, then please request [here](#).

How To

CSS Customization in Blazor TimePicker Component

TimePicker allows to customize the text box and popup list appearance to suit the application by using the [CssClass](#) property.

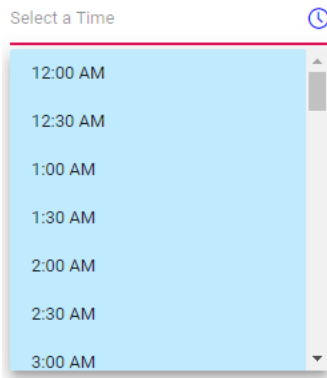
The following code demonstrates customization of text appearance in a text box, popup button, and popup list along with hover and active state by using the **e-custom-style** class. Following is the list of available classes used to customize the entire TimePicker component:

Class Name	Description
---	---
e-time-wrapper	Applied to TimePicker wrapper element.
e-timepicker	Applied to input element and TimePicker popup element.
e-time-wrapper.e-timepicker	Applied to input element only.
e-input-group-icon.e-time-icon	Applied to popup button.
e-float-text	Applied to floating label text element.
e-popup	Applied to popup element.

- | e-timepicker.e-popup | Applied to TimePicker popup element only. |
- | e-list-parent | Applied to popup UL element. |
- | e-timepicker.e-list-parent | Applied to TimePicker popup UL element only. |
- | e-list-item | Applied to LI elements. |
- | e-hover | Applied to LI element hovering time. |
- | e-active | Applied to active LI element. |

ASPX-CS

```
@using Syncfusion.Blazor.Calendars
<SfTimePicker TValue="DateTime?" CssClass="e-custom-style"
Placeholder="Select a Time"></SfTimePicker>
<style>
/*customize the input element text color*/
.e-time-wrapper.e-custom-style #element {
display: block;
color: blue;
}
/*customize the floating label and popup button text color*/
.e-time-wrapper.e-custom-style .e-float-text.e-label-bottom,
.e-time-wrapper.e-custom-style .e-input-group-icon.e-time-icon.e-icons {
color: blue;
}
/*customize the input element text selection*/
.e-time-wrapper.e-custom-style.e-input-group::before,
.e-time-wrapper.e-custom-style.e-input-group::after,
.e-time-wrapper.e-custom-style.e-input-group .e-timepicker::selection {
background: blue;
}
.e-timepicker.e-popup.e-custom-style .e-list-parent.e-ul,
.e-timepicker.e-popup.e-custom-style .e-list-parent.e-ul .e-list-item {
background-color: #c0ebff;
}
/*customize the list item hover color*/
.e-timepicker.e-popup.e-custom-style .e-list-parent.e-ul .e-list-item.e-
hover,
.e-timepicker.e-popup.e-custom-style .e-list-parent.e-ul .e-list-item.e-
active.e-hover {
background-color: blue;
color: #fff;
}
/*customize the active element text color*/
.e-timepicker.e-popup.e-custom-style .e-list-parent.e-ul .e-list-item.e-
active {
color: #333;
background-color: #fff;
}
</style>
```

Limitations of TimeSpan DataType in Blazor TimePicker Component

Based on [C# standard behavior](#), the custom TimeSpan format specifiers do not include placeholder separator symbols, such as the symbols that separate days from hours, hours from minutes, or seconds from fractional seconds. Instead, these symbols must be included in the custom format string as string literals. For example, "hh\\:mm\\:ss" defines a colon (:) as a separator between hours, minutes, and seconds.

CSHARP

```
@using Syncfusion.Blazor.Calendars
<SfTimePicker TValue="TimeSpan" @bind-Value="myTime"
Format="@ ("hh\\:mm\\:ss") "></SfTimePicker>
@code {
    TimeSpan myTime = new TimeSpan(12, 59, 59);
}
```



Toast

Getting Started with Blazor Toast Component

This section briefly explains how to include a Toast component in the Blazor Server-side application. Refer to Getting Started with [Syncfusion Blazor for Server-Side in Visual Studio page](#) for the introduction and configuring the common specifications.

To get start quickly with Blazor Toast component, check on this video:

{% youtube

"youtube:https://www.youtube.com/watch?v=tMa7JvcfNcY"%}

Importing Syncfusion Blazor component in the application

- Install **Syncfusion.Blazor.Notifications** NuGet package to the application by using the **NuGet Package Manager**.
- The client-side resources can be added through [CDN](#) or from [NuGet](#) package in the **HEAD** element of the `~/Pages/_Host.cshtml` page.

ASPX-CS

```
<head>
<environment include="Development">
  ....
  ....
<link href="_content/Syncfusion.Blazor/styles/fabric.css" rel="stylesheet"
/>
<!--CDN-->
@*<link href="https://cdn.syncfusion.com/blazor/18.4.42/styles/fabric.css"
rel="stylesheet" />*@
</environment>
</head>
```

For Internet Explorer 11 kindly refer the polyfills. Refer the [documentation](#) for more information.

ASPX-CS

```
<head>
<environment include="Development">
<link href="_content/Syncfusion.Blazor/styles/fabric.css" rel="stylesheet"
/>
<script
src="https://github.com/Daddoon/Blazor.Polyfill/releases/download/3.0.1/blaz
or.polyfill.min.js"></script>
</environment>
</head>
```

Adding component package to the application

Open `~/_Imports.razor` file and import the **Syncfusion.Blazor.Notifications** package.

ASPX-CS

```
@using Syncfusion.Blazor.Notifications
```

Add SyncfusionBlazor service in Startup.cs

Open the **Startup.cs** file and add services required by Syncfusion components using **services.AddSyncfusionBlazor()** method. Add this method in the **ConfigureServices** function as follows.

CSHARP

```
using Syncfusion.Blazor;
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void ConfigureServices(IServiceCollection services)
        {
            ....
            ....
            services.AddSyncfusionBlazor();
        }
    }
}
```

Add Toast component

To initialize the Toast component, add the below code to **Index.razor** view page which is present under **~/Pages** folder.

The following code explains how to initialize a simple Toast in Razor page.

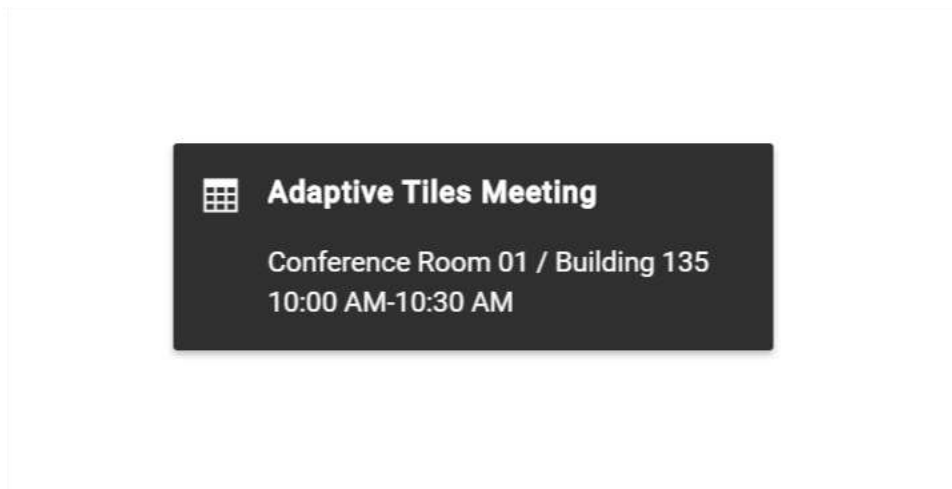
ASPX-CS

```
@using Syncfusion.Blazor.Notifications
<div class="col-lg-12 control-section toast-default-section">
<SfToast ID="toast_default" @ref="ToastObj" Title="Adaptive Tiles Meeting"
Content="@ToastContent" Timeout="5000" Icon="e-meeting">
<ToastPosition X="@ToastPosition"></ToastPosition>
</SfToast>
<div class="col-lg-12 col-sm-12 col-md-12 center">
<div id="toastBtnDefault" style="margin: auto;text-align: center">
<button class="e-btn" @onclick="@ShowOnClick">Show Toasts</button>
<button class="e-btn" @onclick="@HideOnClick">Hide All</button>
</div>
</div>
</div>
<style>
#toast_default .e-meeting::before {
content: "\e705";
font-size: 17px;
}
.bootstrap4 #toast_default .e-meeting::before {
content: "\e763";
font-size: 20px;
}
</style>
@code {
SfToast ToastObj;
private string ToastPosition = "Right";
```

```
private string ToastContent = "Conference Room 01 / Building 135 10:00 AM-10:30 AM";
private async Task ShowOnClick()
{
    await this.ToastObj.ShowAsync();
}
private async Task HideOnClick()
{
    await this.ToastObj.HideAsync("All");
}
}
```

Run the application

After successful compilation of the application, run the application.



See Also

- [Getting Started with Syncfusion Blazor for client-side in .NET Core CLI](#)
- [Getting Started with Syncfusion Blazor for server-side in Visual Studio](#)
- [Getting Started with Syncfusion Blazor for server-side in .NET Core CLI](#)

Configuring options in Blazor Toast Component

This section explains the steps required to customize the appearance of the toast using built-in APIs.

Title and content template

Toast can be created with the notification message. The message contains **Title** and content of the toasts. The title and contents are adaptable in any resolution.

The Title or **Content** property can be given as HTML Element/element ID to a string that can be displayed as a toast.

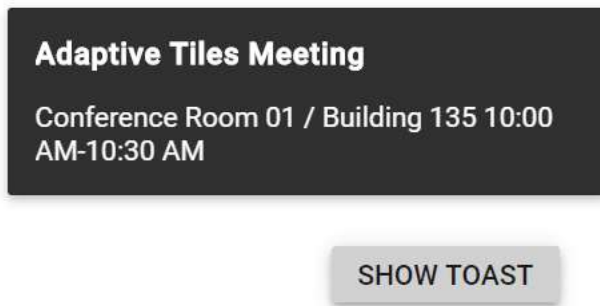
ASPX-CS

```
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.Notifications
<SfToast ID="toast_default" @ref="ToastObj" Title="Adaptive Tiles Meeting"
Content="@ToastContent">
<ToastPosition X="Center"></ToastPosition>
```

```

</SfToast>
<div class="col-lg-12 col-sm-12 col-md-12 center">
<div id="toastBtnDefault" style="margin: auto;text-align: center">
<SfButton @onclick="@ShowToast"> Show Toast </SfButton>
</div>
</div>
<style>
#toast_default .e-meeting::before {
content: "\e705";
font-size: 17px;
}
</style>
@code {
SfToast ToastObj;
private string ToastContent = "Conference Room 01 / Building 135 10:00 AM-
10:30 AM";
private async Task ShowToast()
{
await this.ToastObj.ShowAsync();
}
}

```



Specifying custom target

By default, the toast can be rendered in the document body. The target position can be changed for the toast rendering using the **Target** property. Based on the target, the **Position** will be updated.

Close button

By default, the **ShowCloseButton** is not enabled. It can be enabled by setting the true value. Before expiring the toast, use this button to close or destroy toasts manually.

Progress bar

By default, the **ShowProgressBar** is not enabled. If it is enabled, it can visually indicate how long it gets for the toast to expire. Based on the **Timeout** property, progress bar will appear.

Progress bar direction

By default, the **ProgressDirection** is set to "Rtl" and it will appear from right to left direction. The progressDirection can be changed to "Ltr" to make it appear from left to right direction.

Newest on top

By default, the newly created toasts will append next with existing toasts. The sequence can be changed like inserting before the toast by enabling the **NewestOnTop**.

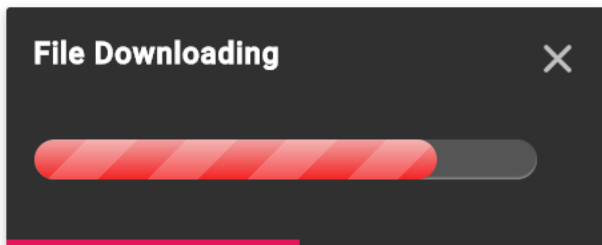
Here, the following sample demonstrates the combination of the `Target`, `ShowCloseButton`, `ShowProgressBar` and `NewestOnTop` properties in toast.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.Notifications
<div class="control-section toast-default-section">
<SfToast ID="toast_default" @ref="ToastObj" Title="File Downloading"
Content="@ToastContent"
ShowCloseButton="true" ProgressDirection="Ltr" Target="#toast_target"
NewestOnTop="true" ShowProgressBar="true">
<ToastPosition X="Center"></ToastPosition>
</SfToast>
<div class="col-lg-12 col-sm-12 col-md-12 center">
<div id="toastBtnDefault" style="margin: auto; text-align: center">
<SfButton @onclick="@ShowToast"> Show Toast </SfButton>
</div>
</div>
<br /><br />
<div id='toast_target'></div>
</div>
@code {
SfToast ToastObj;
private string ToastContent = "<div class='progress'><span style='width:
80%'></span></div>";
private async Task ShowToast()
{
await this.ToastObj.ShowAsync();
}
}
<style>
#toast_default .e-meeting::before {
content: "\e705";
font-size: 17px;
}
.progress {
height: 20px;
position: relative;
margin: 20px 0 20px 0;
background: #555;
-moz-border-radius: 25px;
-webkit-border-radius: 25px;
border-radius: 25px;
box-shadow: inset 0 -1px 1px rgba(255, 255, 255, 0.3);
}
.progress span {
background-color: #f0a3a3;
background-image: -webkit-gradient(linear, left top, left bottom, color-
stop(0, #f0a3a3), color-stop(1, #f42323));
display: block;
height: 100%;
border-radius: 10px;
width: 50%;
position: relative;
overflow: hidden;
}
```

```
.progress span::after {
background-image: -webkit-gradient(linear, 0 0, 100% 100%, color-stop(.25,
rgba(255, 255, 255, .2)), color-stop(.25, transparent), color-stop(.5,
transparent), color-stop(.5, rgba(255, 255, 255, .2)), color-stop(.75,
rgba(255, 255, 255, .2)), color-stop(.75, transparent), to(transparent));
content: "";
position: absolute;
top: 0;
left: 0;
bottom: 0;
right: 0;
background-size: 50px 50px;
-webkit-animation: moveAnimate 2s linear infinite;
overflow: hidden;
}
@@-webkit-keyframes moveAnimate {
0% {
background-position: 0 0;
}
100% {
background-position: 50px 50px;
}
}
</style>
```

SHOW TOAST



Width and height

The dimensions of the toast can be set using the `Width` and `Height` properties. This will individually set all toasts. Different custom dimension toasts can be created.

By default, the toast can be rendered with `300px` width with `auto` height.

In mobile devices, the default width of the toast gets '100%' width of the page. When the toast width is set as '100%', the toast occupies full width and will be displayed at the top or bottom based on the `position` property.

Both the width and height properties allow setting pixels/numbers/percentage. The number value is considered as pixels.

ASPX-CS

```

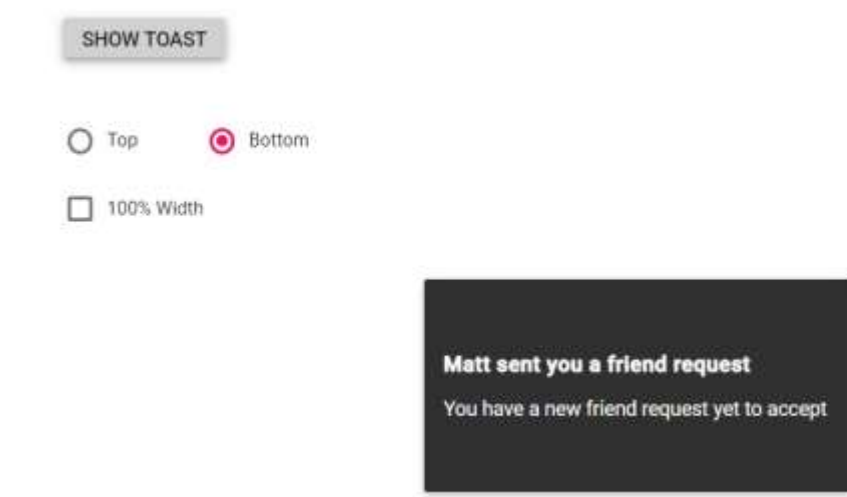
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.Notifications
<div class="control-section toast-default-section">
<SfToast @ref="ToastObj" Title="@Title" Width="@Width" Height="@Height"
Content="@ToastContent">
<ToastPosition X="Center" Y="@PositionY"></ToastPosition>
</SfToast>
<div class="col-lg-12 col-sm-12 col-md-12 center">
<div id="toastBtnDefault" style="margin: auto; text-align: center">
<SfButton @onclick="@ShowToast"> Show Toast </SfButton>
</div>
</div>
<div class='row' style="padding-top: 20px" id="toast_pos_target">
<table style="margin-left: 200px">
<tr>
<td>
<div style='padding:25px 0 0 0;*>
<SfRadioButton Name="toast" Label="Top" Value="Target" TChecked="string"
@bind-Checked="@RadioTopChecked"
ValueChange="@RadioButtonChange"></SfRadioButton>
</div>
</td>
<td>
<div style='padding:25px 0 0 0;*>
<SfRadioButton Name="toast" Label="Bottom" Value="Global" TChecked="string"
ValueChange="@RadioButtonChange" @bind-
Checked="@RadioBottomChecked"></SfRadioButton>
</div>
</td>
</tr>
<tr>
<td>
<div style='padding:25px 0 0 0;*>
<SfCheckBox Label="100% Width" TChecked="bool"
ValueChange="@CheckBoxChange"></SfCheckBox>
</div>
</td>
</tr>
</table>
</div>
<br /><br />
<div id='result'></div>
</div>
@code {
SfToast ToastObj;
private string Width { get; set; } = "400";
private string Height { get; set; } = "120";
private string PositionY { get; set; } = "Bottom";
private string RadioBottomChecked { get; set; } = "Global";
private string RadioTopChecked { get; set; } = "Target";
private string Title { get; set; } = "Matt sent you a friend request";
private string ToastContent { get; set; } = "You have a new friend request
yet to accept";
private async Task ShowToast()
{
await this.ToastObj.ShowAsync();
}
}

```



```
private async Task
CheckBoxChange(Syncfusion.Blazor.Buttons.ChangeEventArgs<bool> e)
{
    if (e.Checked)
    {
        await ToastObj.HideAsync();
        this.Width = "100%";
        this.Title = "";
        this.ToastContent = "<div class='e-custom'>Take a look at our next
generation <b>Javascript</b> <a
href='https://blazor.syncfusion.com/home/index.html' target='_blank'>LEARN
MORE</a></div>";
        StateHasChanged();
    }
    else
    {
        await ToastObj.HideAsync();
        this.Width = "300";
        this.Title = "Matt sent you a friend request";
        this.ToastContent = "You have a new friend request yet to accept";
        StateHasChanged();
    }
}

private async Task
RadioButtonChange(Syncfusion.Blazor.Buttons.ChangeArgs<string> e)
{
    if (e.Value == "Target")
    {
        this.PositionY = "Top";
        await ToastObj.HideAsync();
        StateHasChanged();
    }
    else if (e.Value == "Global")
    {
        this.PositionY = "Bottom";
        await ToastObj.HideAsync();
        StateHasChanged();
    }
}
}
```



Positioning in Blazor Toast Component

The toast position can be updated based on predefined positions or customizable positions. The predefined position combinations are updated in the **X** and **Y** position properties.

Predefined

X Positions

- Left
- Center
- Right

Y Positions

- Top
- Bottom

In multiple toast display, the new toast position will not be updated on dynamic change of property values until the old toast messages removed. The toast occupies full width when the width is set to '100%', so the X positions will not affect the changes when the width is '100%'.

Custom

Custom **X** and **Y** positions can be given as pixels/numbers/percentage. The number value is considered as pixels based on the top and left values updated in the toast.

ASPX-CS

```
@using Syncfusion.Blazor.Inputs
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.DropDowns
@using Syncfusion.Blazor.Notifications
<SfToast @ref="ToastObj" Target="@ToastTarget" Title="Matt sent you a friend
request" Height="@Height" Width="@Width" Content="@ToastContent">
  <ToastPosition X="@PositionX" Y="@PositionY"></ToastPosition>
</SfToast>
<div class="col-lg-12 col-sm-12 col-md-12 center">
```

```

<div id="toastBtnDefault" style="margin: auto; text-align: center">
  <SfButton @onclick="ShowToast"> Show Toast </SfButton>
  <SfButton @onclick="HideToast"> Hide All </SfButton>
</div>
</div>
<div class='row' style="padding-top: 20px" id="toast_pos_target">
  <div id="toast_pos"> </div>
  <div id="toast_pos_property">
    <table style="width: 100%">
      <tr>
        <td>
          <div style='padding:25px 0 0 0;'>
            <SfRadioButton Label="Position" Name="toastPos" Value=@("Position")
            Checked=@("Position") ValueChange="@RadioButtonChange"
            TChecked="string"></SfRadioButton>
          </div>
        </td>
        <td>
          <div style='padding:25px 0 0 0;'>
            <SfRadioButton Label="Custom" Name="toastPos" Value=@("Custom")
            ValueChange="@RadioButtonChange" Checked=@("Custom")
            TChecked="string"></SfRadioButton>
          </div>
        </td>
      </tr>
      <tr>
        <td id="dropdownChoose" colspan="2" style="display:@DropDownDisplay;">
          <SfDropDownList @ref="ListObj" Index="5" Placeholder="Select a position"
          PopupHeight="200PX" DataSource="@DropDownData" TValue="string"
          TItem="DropDownfields">
            <DropDownListFieldSettings Text="id"
            Value="text"></DropDownListFieldSettings>
            <DropDownListEvents ValueChange="@DropDownChange"
            TValue="string"></DropDownListEvents>
          </SfDropDownList>
        </td>
        <td colspan="2" id="customChoose" style="display:@TextBoxDisplay;">
          <form id="formId" class="form-horizontal">
            <div class='e-row'>
              <SfTextBox @ref="TextXpos" Value="50" Placeholder="X Position"
              FloatLabelType="@FloatLabelType.Auto"></SfTextBox>
            </div>
            <div class='e-row'>
              <SfTextBox @ref="TextYpos" Value="50" Placeholder="Y Position"
              FloatLabelType="@FloatLabelType.Auto"></SfTextBox>
            </div>
          </form>
        </td>
      </tr>
      <tr>
        <td>
          <div style='padding:25px 0 0 0;'>
            <SfRadioButton Label="Target" Name="toast" Value=@("Target")
            ValueChange="@RadioButtonChange" TChecked="string"></SfRadioButton>
          </div>
        </td>
      </tr>
    </table>
  </div>
</div>

```

```

</td>
<td>
<div style='padding:25px 0 0 0;*>
<SfRadioButton Label="Global" Name="toast" Value=@("Global")
Checked=@("Global") ValueChange="@RadioButtonChange"
TChecked="string"></SfRadioButton>
</div>
</td>
</tr>
</table>
</div>
</div>
<style>
#toast_default .e-meeting::before {
content: "\e705";
font-size: 17px;
}
#toast_pos_property {
width: 250px;
}
#toast_pos_property td {
width: 50%;
}
#toast_pos_target {
margin: 50px 200px;
}
</style>
@code {
SfToast ToastObj;
SfTextBox TextXpos;
SfTextBox TextYpos;
SfDropDownList<string, DropDownfields> ListObj;
private bool CustomFlag;
private string Width = "300";
private string Height = "auto";
private string ToastTarget = null;
private string PositionX = "Center";
private string PositionY = "Bottom";
private string DropDownDisplay { get; set; } = "";
private string TextBoxDisplay { get; set; } = "none";
private string ToastContent = "You have a new friend request yet to accept";
public class DropDownfields
{
public string id { get; set; }
public string text { get; set; }
}
private List<DropDownfields> DropDownData = new List<DropDownfields>()
{
new DropDownfields(){ id= "topleft", text= "Top Left" },
new DropDownfields(){ id= "topright", text= "Top Right" },
new DropDownfields(){ id= "topcenter", text= "Top Center" },
new DropDownfields(){ id= "bottomleft", text= "Bottom Left" },
new DropDownfields(){ id= "bottomright", text= "Bottom Right" },
new DropDownfields(){ id= "bottomcenter", text= "Bottom Center" }
};
private async Task ShowToast()
{

```

```
if (CustomFlag)
{
    this.SetCustomPosValue();
}
await ToastObj.ShowAsync();
}
private async Task HideToast()
{
    await ToastObj.HideAsync("All");
}
//Setting Toast Custom Position
private void SetCustomPosValue()
{
    PositionX = TextXpos.Value;
    PositionY = TextYpos.Value;
}
private async Task
DropDownChange(Syncfusion.Blazor.DropDowns.ChangeEventArgs<string> e)
{
    await ToastObj.HideAsync("All");
    this.SetToastPosition(e.Value.ToString());
}
private void SetToastPosition(string value)
{
    switch (value)
    {
        case "Top Left":
            PositionX = "Left";
            PositionY = "Top";
            break;
        case "Top Right":
            PositionX = "Right";
            PositionY = "Top";
            break;
        case "Top Center":
            PositionX = "Center";
            PositionY = "Top";
            break;
        case "Bottom Left":
            PositionX = "Left";
            PositionY = "Bottom";
            break;
        case "Bottom Right":
            PositionX = "Right";
            PositionY = "Bottom";
            break;
        case "Bottom Center":
            PositionX = "Center";
            PositionY = "Bottom";
            break;
    }
}
private async Task
RadioButtonChange(Syncfusion.Blazor.Buttons.ChangeArgs<string> e)
{
    if (e.Value == "Target")
    {

```

```

await ToastObj.HideAsync("All");
ToastTarget = "#toast_pos_target";
StateHasChanged();
}
else if (e.Value == "Global")
{
await ToastObj.HideAsync("All");
ToastTarget = null;
StateHasChanged();
}
else if (e.Value == "Position")
{
await ToastObj.HideAsync("All");
DropDownDisplay = "table-cell";
TextBoxDisplay = "none";
this.SetToastPosition(ListObj.Value.ToString());
CustomFlag = false;
}
else if (e.Value == "Custom")
{
await ToastObj.HideAsync("All");
TextBoxDisplay = "table-cell";
DropDownDisplay = "none";
this.SetCustomPosValue();
CustomFlag = true;
}
}
}
}

```

Action Buttons in Blazor Toast Component

Action buttons can be included to the toast control by adding the `ToastButton` property. The click event callback function can also be included for each button.

In the following code, toast buttons are configured using `ToastButton` property.

ASPX-CS

```

@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.Notifications
<SfToast @ref="ToastObj" Title="Anjolie Stokes" Width="280" Height="120"
Icon="e-laura" Content="@ToastContent">
  <ToastPosition X="Right" Y="Bottom"></ToastPosition>
  <ToastButtons>
    <ToastButton Content = "Ignore" OnClick="@HideToast"></ToastButton>
    <ToastButton Content = "reply" OnClick="@HideToast"></ToastButton>
  </ToastButtons>
</SfToast>
<div class="col-lg-12 col-sm-12 col-md-12 center">
  <div id="toastBtnDefault" style="margin: auto; text-align: center">
    <SfButton @onclick="@ShowToast"> Show Toast </SfButton>
  </div>
</div>
<style>
.e-toast-icon.e-laura.e-icons {
border-radius: 50%;
background-repeat: no-repeat;

```

```
background-size: cover;
background-image:
url (https://blazor.syncfusion.com/demos/images/toast/laura.png);
height: 44px !important;
margin: 0 10px 0 0;
width: 60px;
}
#elementToastTime .e-toast-message {
padding: 10px;
}
</style>
@code {
SfToast ToastObj;
private string ToastContent { get; set; } = "Thanks for the update!";
private async Task ShowToast()
{
await this.ToastObj.ShowAsync();
}
private async Task HideToast()
{
await this.ToastObj.HideAsync();
}
}
```

SHOW TOAST



See Also

- [Configuring options](#)

Timeout in Blazor Toast Component

The toast can be expired based on the **Timeout** property. The toast can live till the time out reaches without user interaction, a time out value is considered as a millisecond.

- The **Timeout** delay can be visually represented using [Progress Bar](#).
- The **ExtendedTimeOut** property determines how long the toast should be displayed after a user hovers over it.

The process can be terminated by using the **ShowCloseButton** property for destroying the toast at any time.

ASPX-CS

```

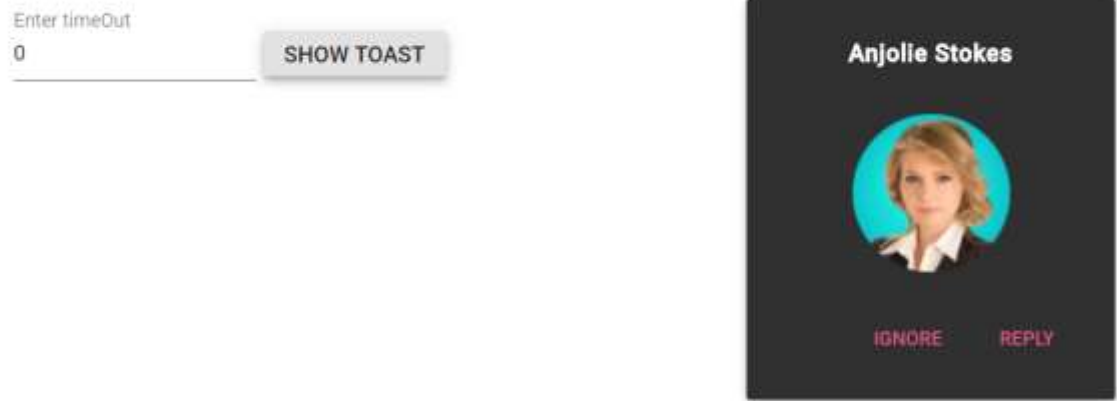
@using Syncfusion.Blazor.Inputs
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.Notifications
<div class="control-section toast-default-section">
<SfToast @ref="ToastObj" Title="Anjolie Stokes" Width="230" Height="250"
Content="@ToastContent" Timeout="@ToastTimeOut">
<ToastPosition X="Right" Y="Bottom"></ToastPosition>
<ToastButtons>
<ToastButton Content = "Ignore" OnClick="@HideToast"></ToastButton>
<ToastButton Content = "reply"></ToastButton>
</ToastButtons>
</SfToast>
<div class="col-lg-12 col-sm-12 col-md-12 center">
<div id="toastBtnDefault" style="margin: auto; text-align: center">
<div id="textbox-contain" style="text-align: initial; display: inline-
block;">
<SfTextBox @ref="TextBoxObj" FloatLabelType="FloatLabelType.Auto"
Placeholder="Enter timeOut" Value="@TextBoxVal"
ValueChange="@OnValChange"></SfTextBox>
</div>
<SfButton @onclick="@ShowToast"> Show Toast </SfButton>
</div>
</div>
<br /><br />
<div id='result'></div>
</div>
<style>
#elementToastTime .e-toast-message {
padding: 10px;
text-align: center;
}
#textbox-contain {
text-align: initial;
display: inline-block
}
</style>
@code {
SfToast ToastObj;
SfTextBox TextBoxObj;
private int ToastTimeOut { get; set; } = 0;
private string TextBoxVal { get; set; } = "0";
private string ToastContent { get; set; } = "<p><img
src='https://blazor.syncfusion.com/demos/images/toast/laura.png'></p>";
private async Task ShowToast()
{
await this.ToastObj.ShowAsync();
}
private void OnValChange()
{
this.ToastTimeOut = int.Parse(this.TextBoxObj.Value);
this.TextBoxVal = this.TextBoxObj.Value;
this.StateHasChanged();
}
private async Task HideToast()
{
await this.ToastObj.HideAsync();
}
}

```



```
}

```

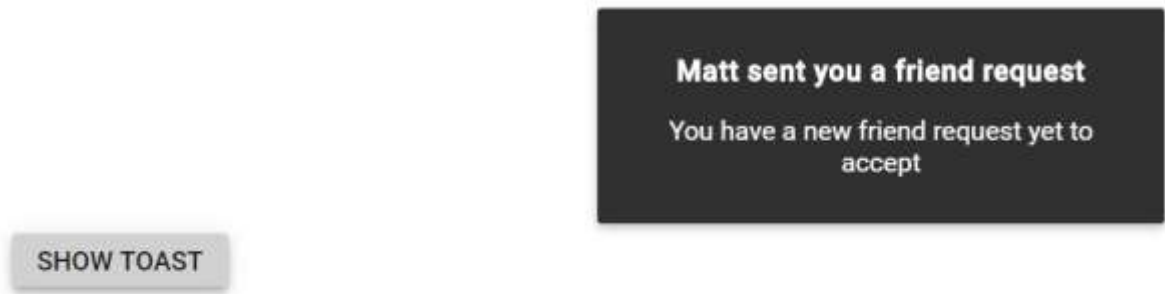


Static toast

Auto hiding can be prevented in a toast as visible like static by setting zero (0) value in the **Timeout** Property.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.Notifications
<SfToast @ref="ToastObj" Timeout=0 Title="Matt sent you a friend request"
Content="@ToastContent">
<ToastPosition X="Right"></ToastPosition>
</SfToast>
<div class="col-lg-12 col-sm-12 col-md-12 center">
<div id="toastBtnDefault" style="margin: auto; text-align: center">
<SfButton @onclick="@ShowToast"> Show Toast </SfButton>
</div>
</div>
<style>
#elementToastTime .e-toast-message {
padding: 10px;
text-align: center;
}
</style>
@code {
SfToast ToastObj;
private string ToastContent = "You have a new friend request yet to accept";
private async Task ShowToast()
{
await this.ToastObj.ShowAsync();
}
}
```



Template in Blazor Toast Component

Configure templates to display within a header, content, and footer section of Toast. The header, content and footer templates can be defined as **HTML element** inside **ToastTemplates**.

The following code explains how to initialize a Toast with **Template**.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.Notifications
<SfButton @onclick="@ShowToast"> Show Toast</SfButton>
<SfToast ID="toast_custom" @ref="@ToastObj" Width="400px" ExtendedTimeout=0
Timeout=120000>
<ToastPosition X="Right" Y="Bottom"></ToastPosition>
<ToastTemplates>
<Template>
<div>
Toast
</div>
</Template>
</ToastTemplates>
</SfToast>
@code {
SfToast ToastObj;
private async Task ShowToast()
{
await this.ToastObj.ShowAsync();
}
}
<style>
@@font-face {
font-family: 'Toast_icons';
src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgTlMvMj0gSRkAAAEoAAAAVmNtYXDNM+eRAAABsAAAAEpnBHlmzVn
mlwAAAhgAAAZAaGVhZBEYI18AAADQAAANmhoZWEHlgN3AAARrAAAACRobXR4LvgAAAAAYAAAAA
wbG9jYQnUCGIAAAH8AAAAGm1heHABHQBcAAABCAAAACBuYW11fUUTYwAACFgAAAKpcG9zdAxfTDg
AAAsEAAAAGgABAAADUv9qAFoEAAAAAAD6AABAAAAAAAAAAAAAAAAADAABAAAAQAACcU5MF8
PPPUACwPoAAAAANcI7skAAAAA1wjuyQAAAAAD6APoAAAACAACAAAAAAAAAAAAEAAAAMAFABwAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPqAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAUGZFZABA5wDnCcNS/2oAWgPoAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAAAAGAAAAAUAAMAAQA
AABQABAA2AAAABAAEAAEAOCk//8AAOCa//8AAAABAAQAAAAABAAIAAwAEAAUABgAHAAGACQAKAAs
AAAAAAAAAQgB8AMIA4gEcAZQCBgJwAo4DAAMgAAAAAwAAAAAD1AOUAAsAFwAjAAABFwcXNxc3Jzc
nBycFDgEHLgEnPgE3HgEFHgEXPgE3LgEnDgEBTXh4L3h4L3h4L3h4AbwDt4qKtwMDt4qKt/0eBeu
```

```

xsesFBeuxsesCbHh4L3h4L3h4L3h4p4q3AwO3ioq3AwO3irHrBQXrsbHrBQXrAAAAAwAAAAADlAO
UAAUAEQAdAAABJwcXAScXDgEHLgEnPgE3HgEFHgEXPgE3LgEnDgEBR2UylwEbMqADt4qKtwMDt4q
Kt/0eBeuxsesFBeuxsesBrGQylgEcMqKKtwMDt4qKtwMDt4qx6wUF67Gx6wUF6wAAAAFAAAAAAO
UA5cABQARAB0AIQAlAAABFzcnNSMFDgEHLgEnPgE3HgEFHgEXPgE3LgEnDgElFzcnBRc3JwHKxiC
nPwFOA6V8fKUDA6V8fKX9aATToJ/UBATUn5/UAh7ANsD9fja/NQGedzNj29F8pAMDpHx8pQMDpXy
f1AQE1J+g0wQE0/GhQKGhQKFAAAQAAAAA74DfgADAACACgANAAAlMzUjNTM1IwEhCQEhAQHLULJ
SUgFj/YwBOv42A5T+NuZUUqf+igIc/ZADfGAEAAAAAOUA5QAaWAhABMAHwAAATM1IzUzNSMFDgE
HLgEnPgE3HgEFHgEXPgE3LgEnDgEBYlRUVFQBbgO3ioq3AwO3ioq3/R4F67Gx6wUF67Gx6wEk+1N
T0Iq3AwO3ioq3AwO3irHrBQXrsbHrBQXrAAAAAACAAAAA+gDMQALABUAJQAUAdCAQQBLAAABFhc
VITUmJz4BMxYFFhcVITU+ATcWJQYHFSE1LgEjIgyHLgEjIgyEFAYiJjQ2MgUWFAYiJjQ2MiUGFBy
XPgE0JiIFBhQWFz4BNCYiAlxEBP6sAxUeRiRX/qxEBP45BilXV/7xZQsD6AvKUypvMzNvKlMCKxo
zTTMzTP6CGTNMNDRMAQItWUREWlqI/jstWkREWVmIAWMbFjc3IBgKDwQcGxY3NxY3BAQjJUt7e0t
KFxgyFwEMGU01NU0zGhlNNTVMxYthloCAlqGWy4thloCAlqGWwAAAAQAAAAA5wCxAIABQANAB
FAAABFByNjQmIgyXDgEHLgEnPgE3HgEfaQcOQ8BNz4BNS4BJw4BBxQWHwEnLgEvATc+ATc2FiU
OAQ8BFx4BNz4BPwEnJiciAb8fLR4eLR+wAkU0NEUBAUU0NEX8BgEemG0FBB8kAlZBQfCBKyUCCke
VTAYBH76RVMP+3bDPBwcKZclcu/AGCwrM2AoBxxYfHy0eHhc0RQEBRTQ1RQEBSgEARpWGAECFUI
oQVcCaldBLEYUAQEIQkAGASJsBwFCoRbFFaOJW0sBCo8LCgztaQAAAAIAAAAAA4ADbaA4AEAAAAE
EJCcmDgEWFx4BHwEVfAYHDgEnJg4BFhcWNjc2Fx4BBx4BFzc+ASc2JicmJzUzPgE3PgEnJicjIiU
UFjI2NCYiBgNM/tz+pwWMGxEDDAaMfAcSETKEQw8WBg8Og80hNSg4JwICEw0FDhECAjFJEBICPYh
KDQgGChQCB/5dMUGxMUGxAuB/ZRcIAxgbCQdHEQGTGi8TOVGKAw8dFwMNUdUFTGDCA0QAQECFQ8
Mnz8LCasJKiUHGG0SATMkMDBJMDAAAAAAGAAAAAC/QMkAAMADQAAQchJxMeATMhMjY3EYEC2x3
+bB0kBCQZAQQZJARH/ewDBuDg/fcZICAZaicaAwAAAAACzwPoACwAQwBPAAABERQfARYfAzMVHgE
7ATI2NRE0JisBNTeWOWeyNjQmJyMiJi8BLgErAQ4BAxUzNTQ2NzMeARcVMzUuAScjIgcjESM1HgE
XPgE3LgEnDgEBVQEBaWQCCAjXAREN0g0REQ2zDROVExoae2UQGAQfAxAKYg0RPR8RDZcNEQEeASI
alxANAR8CTTo6TQEBTTo6TQJ8/nYEBQIGBAIFArYNERENARENEUoNGicZARMPfQoNARH98H15DRE
BARENeXkaIgEIAe3Fok0CAk06Ok0BAU0AAAAAAGAAAAAC5gMyAAkAEQAAJRQWMyEyNjURITcjFSE
1IycjASApHgEaHin+WFBuAeR+JLD8HigoHgGfeT09HgAAAAAAEgDeAAEAAAAAAAAAAAAEAAAA
AAAEAEgABAAEAAAAAAAAIABwATAAEAAAAAAAAMAEgAaAAEAAAAAAAAQAEgAsAAEAAAAAAAAUACwA+AAE
AAAAAAAYAEgBJAAEAAAAAAAAoALABbAAEAAAAAAAAAEgCHAAMAAQQJAAAAAgCZAAMAAQQJAAEAJAC
bAAMAAQQJAAIADgC/AAMAAQQJAAAMAJADNAAMAAQQJAAQAADxAMAAQQJAAUAFgEVAAMAAQQJAAAY
AJAERAAAMAAQQJAAoAWAFPAAMAAQQJAAAsAJAGNIEZpbmFsIFRvYXN0IE1ldHJvcFJlZ3VsYXJGaW5
hbCBUb2FzdCBNZXRyb3BGaW5hbCBUb2FzdCBNZXRyb3BWZXJzaW9uIDEuMEZpbmFsIFRvYXN0IE1
ldHJvcEZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3RlZGlvd3d3LnN5bmN
mdXNpb24uY29tACAARgBpAG4AYQBsaCAAVABvAGEAcwB0ACAATQBlAHQAQcBvAHAAUgBlAGcAdQB
sAGEAcgBGAGkAbgBhAGwAIABUAG8AYQBzAHQAIAABNAGUAdABYAG8AcABGAGkAbgBhAGwAIABUAG8
AYQBzAHQAIAABNAGUAdABYAG8AcABWAGUAcgBzAGkAbwBuACAAMQAuADAARgBpAG4AYQBsaCAAVAB
vAGEAcwB0ACAATQBlAHQAQcBvAHAARgBvAG4AdAAgAGcAZQBwAGUAcgBhAHQAQcBkACAADQBzAGk
AbgBnACA AUwB5AG4AYwBmAHUAcwBpAG8AbgAgAE0AZQB0AHIAbwAgAFMAdABlAGQAAQbVAHcAdwB
3AC4AcwB5AG4AYwBmAHUAcwBpAG8AbgAuAGMabwBtAAAAAIAAAAAAAAAACgAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAADAECAMBAEFAQYBBwEIAQkBCgELAQwBDQAFRXJyb3IHU3VjY2VzcwVBbGFyYQd
XYXJuaW5nBEluZm8HTWVldGluZWVcbGluawdTdHJldGNoAlNpcANTaXQFVHJhc2gAAAAA)
format('trueType');
font-weight: normal;
font-style: normal;
}
.toast-icons {
font-family: 'Toast_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;
}

```

```

#toast_custom .e-toast .e-toast-title,
#toast_custom .e-toast .e-toast-message,
#toast_custom .e-toast .e-toast-message .e-toast-content,
#toast_custom .e-toast .e-toast-close-icon {
color: #fff;
}
#toast_custom .e-toast-container .e-toast .e-toast-message .e-toast-content
{
padding: 14px 0 0 0;
}
#toast_custom .e-toast-template {
display: inline-flex;
}
#toast_custom .e-toast-icon.e-toast-image {
border-radius: 50%;
background-repeat: no-repeat;
background-size: cover;
height: 50px !important;
width: 50px !important;
background-size: 50px 50px;
align-self: center;
}
#toast_custom .camden .e-toast-icon.e-toast-image,
#toast_custom .chase .e-toast-icon.e-toast-image {
width: 65px !important;
}
#template_toast .horizontal-align .toast-content .toast-title {
font-weight: 500;
color: #fff;
}
#template_toast .horizontal-align .toast-content .toast-message {
opacity: 0.75;
color: #fff;
}
#template_toast .toast-icons {
font-size: 35px;
height: auto;
margin: auto;
}
#template_toast .horizontal-align {
display: inline-flex;
flex-direction: row;
width: 100%;
}
#template_toast .horizontal-align .toast-content {
display: inline-flex;
flex: 1;
flex-direction: column;
margin-left: 10px;
}
</style>

```

The Template can be given as a RenderFragment type for the **ContentTemplate** property when updating the toast templates dynamically. The below code block explains the string template.

ASPX-CS

```

@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.Notifications
<SfButton @onclick="@ShowToast"> Show Toast </SfButton>
<SfToast @ref="ToastObj" />
@code {
    SfToast ToastObj;
    private RenderFragment template = @<div id='toastEmail_template'><div
class='e-toast-template'><img class='e-toast-icon e-toast-image'
src='https://blazor.syncfusion.com/demos/images/toast/laura.png' /><div
class='e-toast-message'><div class='e-toast-title'>Anjolie Stokes</div><div
class='e-toast-content'>Networking Referral</div></div></div></div></div>;
    private async Task ShowToast()
    {
        await this.ToastObj.ShowAsync(new ToastModel { ContentTemplate = template
    });
    }
}

```

Animation in Blazor Toast Component

The toast component supports custom animations for both show and hide actions from the provided `ToastHideAnimationSettings` and `ToastShowAnimationSettings` option of the `Animation` library. The default animation is given as `FadeIn` for showing the toast and `FadeOut` for hiding the toast.

The following sample demonstrates some types of animations that suit toast.

ASPX-CS

```

@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.DropDowns
@using Syncfusion.Blazor.Notifications
<SfToast @ref="ToastObj" Title="Matt sent you a friend request"
Content="@ToastContent">
    <ToastPosition X="Right" Y="Bottom"></ToastPosition>
    <ToastAnimationSettings>
        <ToastShowAnimationSettings>
            Effect="@ShowAnimation"></ToastShowAnimationSettings>
        <ToastHideAnimationSettings>
            Effect="@HideAnimation"></ToastHideAnimationSettings>
        </ToastAnimationSettings>
    </SfToast>
    <div class="col-lg-12 col-sm-12 col-md-12 center">
        <div id="toastBtnDefault" style="margin: auto; text-align: center">
            <div id="textbox-contain">
                <div>
                    <label> Show Animation </label>
                </div>
                <SfDropDownList Placeholder="Select a animate type" DataSource="@Effects"
                    TValue="string" TItem="DropDownFields">
                    <DropDownListEvents ValueChange="@ShowAnimationChange"
                        TValue="string"></DropDownListEvents>
                    <DropDownListFieldSettings Text="text"
                        Value="id"></DropDownListFieldSettings>
                </SfDropDownList>
                <div>
                    <label> Hide Animation </label>
                </div>
            </div>
        </div>
    </div>

```

```

<SfDropDownList Placeholder="Select a animate type" DataSource="@Effects"
TValue="string" TItem="DropDownFields">
<DropDownListEvents ValueChange="@HideAnimationChange"
TValue="string"></DropDownListEvents>
<DropDownListFieldSettings Text="text"
Value="id"></DropDownListFieldSettings>
</SfDropDownList>
</div>
<SfButton @onclick="@ShowToast"> Show Toast </SfButton>
</div>
</div>
<style>
#elementToastTime .e-toast-message {
padding: 10px;
text-align: center;
}
#textbox-contain {
text-align: initial;
display: inline-block;
width: 20%;
margin: 0 auto;
}
.top-row.px-4 {
display: none;
}
.content.px-4 {
margin-top: 103px;
margin-left: -12%;
}
</style>
@code {
SfToast ToastObj;
private ToastEffect ShowAnimation = ToastEffect.FadeIn;
private ToastEffect HideAnimation = ToastEffect.FadeOut;
private string ToastContent = "You have a new friend request yet to accept";
public class DropDownFields
{
public string id { get; set; }
public string text { get; set; }
}
private List<DropDownFields> Effects = new List<DropDownFields>()
{
new DropDownFields(){ id= "FadeIn", text= "Fade In" },
new DropDownFields(){ id= "FadeZoomIn", text= "Fade Zoom In" },
new DropDownFields(){ id= "FadeZoomOut", text= "Fade Zoom Out" },
new DropDownFields(){ id= "FlipLeftDownIn", text= "Flip Left Down In" },
new DropDownFields(){ id= "FlipLeftDownOut", text= "Flip Left Down Out" },
new DropDownFields(){ id= "FlipLeftUpIn", text= "Flip Left Up In" },
new DropDownFields(){ id= "FlipRightDownIn", text= "Flip Right Up In" },
new DropDownFields(){ id= "FlipRightDownOut", text= "Flip Right Down Out" },
new DropDownFields(){ id= "FlipRightUpIn", text= "Flip Right Up In" },
new DropDownFields(){ id= "FlipRightUpOut", text= "Flip Right Up Out" },
new DropDownFields(){ id= "SlideBottomIn", text= "Slide Bottom In" },
new DropDownFields(){ id= "SlideBottomOut", text= "Slide Bottom Out" },
new DropDownFields(){ id= "SlideLeftIn", text= "Slide Left In" },
new DropDownFields(){ id= "SlideLeftOut", text= "Slide Left Out" },
new DropDownFields(){ id= "SlideRightIn", text= "Slide Right In" },

```

```
new DropDownFields(){ id= "SlideRightOut", text= "Slide Right Out" },
new DropDownFields(){ id= "SlideTopIn", text= "Slide Top In" },
new DropDownFields(){ id= "ZoomIn", text= "Zoom In" },
new DropDownFields(){ id= "ZoomOut", text= "Zoom Out" }
};
private async Task ShowToast()
{
    await ToastObj.ShowAsync();
}
private void
ShowAnimationChange (Syncfusion.Blazor.DropDowns.ChangeEventArgs<string>
args)
{
    this.ShowAnimation = (ToastEffect) System.Enum.Parse (typeof (ToastEffect),
args.Value);
    StateHasChanged();
}
private void
HideAnimationChange (Syncfusion.Blazor.DropDowns.ChangeEventArgs<string>
args)
{
    this.HideAnimation = (ToastEffect) System.Enum.Parse (typeof (ToastEffect),
args.Value);
    StateHasChanged();
}
}
```

Style and appearance in Blazor Toast 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 toast title

Use the following CSS to customize the default toast's content properties like font-family, font-size and color.

CSS

```
/* To change color, font family and font size */
.e-toast-container .e-toast .e-toast-message .e-toast-title {
    color: red;
    font-size: 18px;
    font-weight: bold;
}
```

Customizing the toast content

Use the following CSS to customize the default toast's content properties like font-family, font-size and color.

CSS

```
/* To change color, font family and font size */
.e-toast-container .e-toast .e-toast-message .e-toast-content {
    color: aqua;
    font-size: 13px;
    font-weight: normal;
}
```

```
}
```

Customizing the toast icon

Use the following CSS to customize the default toast icon color.

CSS

```
/* To change icon color */
.e-toast-container .e-toast .e-toast-icon {
  color: yellow;
}
```

Customizing the toast background

Use the following CSS to customize the default toast's background color.

CSS

```
/* To change background color */
.e-toast-container .e-toast {
  background-color: navy;
}
```

Accessibility in Blazor Toast Component

The toast component has been designed with [WAI-ARIA](#) specifications in mind by applying the prompt WAI-ARIA roles, states, and properties with the keyboard support. It helps users who use assistive WAI-ARIA accessibility support, which is achieved using attributes.

It provides information about the elements in a document for assistive technology.

The toast component implements the keyboard navigation support by using the following [WAI-ARIA practices](#) and is tested in major screen readers.

ARIA attributes

<!-- markdownlint-disable MD033 -->

Class	Description
-------	-------------

-----	-----
-------	-------

role	alert: Identifies the element as a container when alert content will be added or updated.
------	--

ASPX-CS

```
@using Syncfusion.Blazor.Notifications
<SfToast @ref="ToastObj" Title="Matt sent you a friend request"
Content="@ToastContent" Timeout="0">
  <ToastEvents Created="@OnCreate"></ToastEvents>
</SfToast>
@code {
  SfToast ToastObj;
  private string ToastContent { get; set; } = "You have a new friend request yet to accept";
  private async Task OnCreate()
  {
    await this.ToastObj.ShowAsync();
  }
}
```



```
}  
}
```

Matt sent you a friend request

You have a new friend request yet to accept

Events in Blazor Toast Component

This section explains the list of events of the Toast component which will be triggered for appropriate Toast actions.

Created

Created event triggers after the Toast gets created.

ASPX-CS

```
@using Syncfusion.Blazor.Notifications  
<SfToast>  
<ToastEvents Created="@CreatedHandler" ></ToastEvents>  
</SfToast>  
@code{  
public void CreatedHandler(Object args)  
{  
    // Here you can customize your code  
}  
}
```

Destroyed

Destroyed event triggers after the Toast gets destroyed.

ASPX-CS

```
@using Syncfusion.Blazor.Notifications  
<SfToast>  
<ToastEvents Destroyed="@DestroyedHandler" ></ToastEvents>  
</SfToast>  
@code{  
public void DestroyedHandler(Object args)  
{  
    // Here you can customize your code  
}  
}
```

```
}
```

Opened

Opened event triggers after the Toast shown on the target container.

ASPX-CS

```
@using Syncfusion.Blazor.Notifications
<SfToast>
<ToastEvents Opened="@OpenedHandler" ></ToastEvents>
</SfToast>
@code{
public void OpenedHandler(ToastOpenArgs args)
{
// Here you can customize your code
}
}
```

OnOpen

OnOpen event triggers before the toast shown.

ASPX-CS

```
@using Syncfusion.Blazor.Notifications
<SfToast>
<ToastEvents OnOpen="@OnOpenHandler" ></ToastEvents>
</SfToast>
@code{
public void OnOpenHandler(ToastBeforeOpenArgs args)
{
// Here you can customize your code
}
}
```

Closed

Closed event triggers after the Toast hides.

ASPX-CS

```
@using Syncfusion.Blazor.Notifications
<SfToast>
<ToastEvents Closed="@ClosedHandler" ></ToastEvents>
</SfToast>
@code{
public void ClosedHandler(ToastCloseArgs args)
{
// Here you can customize your code
}
}
```

OnClick

OnClick event triggers while clicking on the Toast.

ASPX-CS

```
@using Syncfusion.Blazor.Notifications
<SfToast>
<ToastEvents OnClick="@OnClickHandler" ></ToastEvents>
</SfToast>
@code{
public void OnClickHandler(ToastClickEventArgs args)
{
// Here you can customize your code
}
}
```

How To

Show different types of toast in Blazor Toast Component

The Blazor toast has the following predefined styles that can be defined using the `CssClass` property for achieving different types of toast:

Class	Description
-----	-----
e-toast-success	Represents a positive toast
e-toast-info	Represents an informative toast
e-toast-warning	Represents a toast with caution
e-toast-danger	Represents a negative toast

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.Notifications
<SfToast @ref="ToastObj" Title="@ToastTitle" Content="@ToastContent"
CssClass="@ToastCssClass">
<ToastPosition X="Right" Y="Bottom"></ToastPosition>
</SfToast>
<div class="col-lg-12 col-sm-12 col-md-12 center">
<div style="margin: auto; text-align: center">
<SfButton @onclick="@ShowToast"> Show Toast </SfButton>
</div>
</div>
@code {
SfToast ToastObj;
private int ToastFlag = 0;
private string ToastTitle = "";
private string ToastContent = "";
private string ToastCssClass = "";
private class ToastOption
{
public string Title { get; set; }
public string Content { get; set; }
public string CssClass { get; set; }
}
private ToastOption[] Toasts = new ToastOption[] {
new ToastOption { Title = "Warning !", Content = "There was a problem with
your network connection.", CssClass = "e-toast-warning" },
```

```

new ToastOption { Title = "Success !", Content = "Your message has been sent
successfully.", CssClass = "e-toast-success" },
new ToastOption { Title = "Error !", Content = "A problem has been occurred
while submitting your data.", CssClass = "e-toast-danger" },
new ToastOption { Title = "Information !", Content = "Please read the
comments carefully.", CssClass = "e-toast-info" }
};
private async Task ShowToast()
{
this.ToastTitle = this.Toasts[this.ToastFlag].Title;
this.ToastContent = this.Toasts[this.ToastFlag].Content;
this.ToastCssClass = this.Toasts[this.ToastFlag].CssClass;
await Task.Delay(100);
await this.ToastObj.ShowAsync();
this.ToastFlag = ((this.ToastFlag == Toasts.Length - 1) ? 0 :
(this.ToastFlag + 1));
}
}

```

Show multiple toasts in various positions in Blazor Toast Component

By default, the positions of the new toasts are only updated after the visible toasts have been destroyed. If there is a need to display multiple toasts with different positions, initiate another toast.

The following sample demonstrates adding multiple toasts in different positions.

ASPX-CS

```

@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.Notifications
<SfToast @ref="WarningToastObj" Title="Warning !" Content="@WarningContent">
<ToastPosition X="Right" Y="Bottom"></ToastPosition>
<ToastEvents OnClick="@ToastClick"></ToastEvents>
</SfToast>
<SfToast @ref="SuccessToastObj" Title="Success !" Content="@SuccessContent">
<ToastPosition X="Left" Y="Bottom"></ToastPosition>
<ToastEvents OnClick="@ToastClick"></ToastEvents>
</SfToast>
<div class="col-lg-12 col-sm-12 col-md-12 center">
<div style="margin: auto; text-align: center">
<SfButton @onclick="@ShowWarningToast"> Show Warning Toast </SfButton>
<SfButton @onclick="@ShowSuccessToast"> Show Success Toast </SfButton>
</div>
</div>
@code {
SfToast WarningToastObj;
SfToast SuccessToastObj;
private string SuccessContent { get; set; } = "Your message has been sent
successfully.";
private string WarningContent { get; set; } = "There was a problem with your
network connection.";
private void ToastClick(ToastClickEventArgs args)
{
args.ClickToClose = true;
}
private async Task ShowWarningToast()
{

```

```
await this.WarningToastObj.ShowAsync();
}
private async Task ShowSuccessToast()
{
    await this.SuccessToastObj.ShowAsync();
}
}
```

Show toast content dynamically in Blazor Toast Component

The Toast component supports to change its content dynamically while displaying in newest toasts. The toast content can be changed by updating property value, before calling in the `show` method.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.Notifications
<SfToast @ref="ToastObj" Title="Alert" Content="@ToastContent">
    <ToastPosition X="Right" Y="Bottom"></ToastPosition>
</SfToast>
<div class="col-lg-12 col-sm-12 col-md-12 center">
    <div style="margin: auto; text-align: center">
        <SfButton @onclick="@ShowToast"> Show Toast </SfButton>
    </div>
</div>
@code {
    SfToast ToastObj;
    private int ToastFlag = 0;
    private string ToastContent = "";
    private string[] Contents = new string[] {
        "Welcome User",
        "Upload in progress",
        "Upload success",
        "Profile updated",
        "Profile picture changed",
        "Password changed"
    };
    private async Task ShowToast()
    {
        this.ToastContent = this.Contents[this.ToastFlag];
        await Task.Delay(100);
        await this.ToastObj.ShowAsync();
        this.ToastFlag = ((this.ToastFlag != 5) ? (this.ToastFlag + 1) : 0);
    }
}
```

Close the toast with click/tap in Blazor Toast Component

By default, the toasts are expired based on the `Timeout` value. The toast hiding process can be customized with click/tap action by setting the `ToastClickEventArgs` in the clicked callback function with static Toast.

The following sample demonstrates the click or tap action in toast.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
```

```

@using Syncfusion.Blazor.Notifications
<SfToast @ref="ToastObj" Title="Alert" Content="@ToastContent">
<ToastEvents OnClick="@OnClickHandler"></ToastEvents>
<ToastPosition X="Right" Y="Bottom"></ToastPosition>
</SfToast>
<div class="col-lg-12 col-sm-12 col-md-12 center">
<div style="margin: auto; text-align: center">
<SfButton @onclick="@ShowToast"> Show Toast </SfButton>
</div>
</div>
@code {
SfToast ToastObj;
private int ToastFlag = 0;
private string ToastContent = "";
private string[] Contents = new string[] {
"Welcome User",
"Upload in progress",
"Upload success",
"Profile updated",
"Profile picture changed",
"Password changed"
};
private async Task ShowToast()
{
this.ToastContent = this.Contents[this.ToastFlag];
await Task.Delay(100);
await this.ToastObj.ShowAsync();
this.ToastFlag = ((this.ToastFlag != 5) ? (this.ToastFlag + 1) : 0);
}
public void OnClickHandler(ToastClickEventArgs args)
{
args.ClickToClose = true;
}
}

```

Add dynamic template in Blazor Toast Component

Toast supports to change templates dynamically with displaying in multiple toasts. The toast properties can be changed while calling in the **Show** method.

ASPX-CS

```

@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.Notifications
<SfToast @ref="ToastObj">
<ToastPosition X="Right" Y="Bottom"></ToastPosition>
</SfToast>
<div class="col-lg-12 col-sm-12 col-md-12 center">
<div style="margin: auto; text-align: center">
<SfButton @onclick="@ShowToast"> Show Toast </SfButton>
</div>
</div>
@code {
SfToast ToastObj;
private int ToastFlag = 0;
private ToastModel[] Messages = new ToastModel[] {
new ToastModel() { ContentTemplate = @<p>2 Mail has received</p>},

```

```

new ToastModel() { ContentTemplate = @<p>User Guest Logged in</p>},
new ToastModel() { ContentTemplate = @<p>Logging in as Guest</p>},
new ToastModel() { ContentTemplate = @<p>Ticket has reserved</p>}
};
private async Task ShowToast()
{
    // Delay mandatory to update the dynamically changed Toast properties
    await Task.Delay(100);
    await this.ToastObj.ShowAsync(Messages[this.ToastFlag]);
    this.ToastFlag = ((this.ToastFlag == Messages.Length - 1) ? 0 :
    (this.ToastFlag + 1));
}
}

```

Access specific toast in Blazor Toast Component

In the toast, the particular toast can be accessed by passing the **Key** value in **ShowModes**, and the **Key** should be unique in **ShowModels**. To close the specific toast, you also need to pass the corresponding toast **Key** value in the **Hide** method. The added **Key** value can be got in the toast **Opened** and **Closed** event.

In the following example, Toast is closed by calling the **Hide** method with the key value that is returned in the **Opened** event

ASPX-CS

```

@using Syncfusion.Blazor.Notifications
<div id="target">
<SfToast @ref="toast" ShowCloseButton="true" Height="150px" Width="200px"
Timeout="60" Target="@target" Icon="e-meeting" Title="@title" >
<ToastEvents Opened="OnOpen"></ToastEvents>
</SfToast>
<div class="col-lg-12 col-sm-12 col-md-12 center">
<div id="toastBtnDefault" style="margin: auto;text-align: center">
<button class="e-btn" id="toastBtnShow" @onclick="showOnClick">Show
Toasts</button>
<button class="e-btn" id="toastBtnHide" @onclick="hideOnClick">Hide
All</button>
<button class="e-btn" id="toastBtnHide" @onclick="HideOnClick">Hide</button>
</div>
</div>
</div>
<style>
#toast_default .e-meeting::before {
content: "\e705";
font-size: 17px;
}
</style>
@code{
SfToast toast;
private string target = "#target";
private string title = "This is Toast Title";
private int key { get; set; } = 0;
private async Task showOnClick()
{

```

```

await this.toast.ShowAsync( new ToastModel {Key = key , Content =
key.ToString() , Timeout = 10000 });
key++;
}
private async Task OnOpen(ToastOpenArgs args)
{
await this.toast.HideAsync(args.Key);
}
private async Task HideOnClick()
{
await this.toast.HideAsync();
}
private async Task hideOnClick()
{
await this.toast.HideAsync("All");
}
}

```

Prevent toast close with mobile swipe in Blazor Toast Component

The toast close can be prevented with mobile swipe action by setting **OnClose** argument cancel value to true while argument type is swipe. The following code shows how to prevent toast close with mobile swipe.

The following sample demonstrates preventing toast close with mobile swipe element displaying with custom code blocks.

ASPX-CS

```

@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.Notifications
<SfToast ID="toast_default" @ref="ToastObj" Title="Adaptive Tiles Meeting"
Content="@ToastContent">
<ToastEvents OnClose="@OnClose"></ToastEvents>
<ToastPosition X="Center"></ToastPosition>
</SfToast>
<div class="col-lg-12 col-sm-12 col-md-12 center">
<div id="toastBtnDefault" style="margin: auto;text-align: center">
<SfButton @onclick="@ShowToast"> Show Toast </SfButton>
</div>
</div>
<style>
#toast_default .e-meeting::before {
content: "\e705";
font-size: 17px;
}
</style>
@code {
SfToast ToastObj;
private string ToastContent = "Conference Room 01 / Building 135 10:00 AM-
10:30 AM";
private async Task ShowToast()
{
await this.ToastObj.ShowAsync();
}
private void OnClose(ToastBeforeCloseArgs args)
{

```



```
if (args.RequestType == "swipe")
{
    args.Cancel = true;
}
}
```

Toggle Switch Button

<!-- markdownlint-disable MD024 -->

Getting Started with Blazor Toggle Switch Button Component

This section briefly explains about how to include Toggle Switch Button Component in the Blazor server-side application. Refer [Getting Started with Syncfusion Blazor for Server-side in Visual Studio](#) page for the introduction and configuring the common specifications.

To get start quickly with Toggle Switch Button Component using Blazor, check on this video:

{% youtube

"youtube:https://www.youtube.com/watch?v=u_5050Ws9JM"%}

Importing Syncfusion Blazor component in the application

1. Install the **Syncfusion.Blazor** NuGet package to the application by using the **NuGet Package Manager**.
2. Add the client-side style resources through [CDN](#) or from [NuGet](#) package in the `~/Pages/_Host.cshtml` page.

Please ensure to check the **Include prerelease** option.

HTML

```
<head>
<link href="_content/Syncfusion.Blazor/styles/bootstrap4.css"
rel="stylesheet" />
@*<link href="https://cdn.syncfusion.com/blazor/{{ site.blazorversion
}}/styles/bootstrap4.css" rel="stylesheet" />*@
</head>
```

For Internet Explorer 11 kindly refer the polyfills. Refer the [documentation](#) for more information.

HTML

```
<head>
<link href="_content/Syncfusion.Blazor/styles/bootstrap4.css"
rel="stylesheet" />
<script
src="https://github.com/Daddoon/Blazor.Polyfill/releases/download/3.0.1/blaz
or.polyfill.min.js"></script>
</head>
```

Adding component package to the application

Open `/_Imports.razor` file and import the **Syncfusion.Blazor.Buttons** package.

CSHARP

```
@using Syncfusion.Blazor.Buttons
```

Add SyncfusionBlazor service in Startup.cs

Open the **Startup.cs** file and add services required by Syncfusion components.

Add **services.AddSyncfusionBlazor()** method in the `ConfigureServices` function as follows.

CSHARP

```
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void ConfigureServices(IServiceCollection services)
        {
            ....
            ....
            services.AddSyncfusionBlazor();
        }
    }
}
```

To enable custom client side resource loading from CRG or CDN. Resource loading has to be disabled by **AddSyncfusionBlazor(true)** and load the scripts in the HEAD element of the `~/Pages/_Host.cshtml` page.

ASPX-CS

```
<head>
<environment include="Development">
<script src="https://cdn.syncfusion.com/blazor/{{ site.blazorversion
}}/syncfusion-blazor.min.js">
</script>
</environment>
</head>
```

Adding Toggle Switch Button component to the application

Now, add the Syncfusion Blazor Toggle Switch Button component in `razor` page in the `Pages` folder. For example the Toggle Switch Button component is added in the `~/Pages/Index.razor` page.

ASPX-CS

```
<label for="checked" style="padding: 10px 10px 10px 0">USB Tethering</label>
<SfSwitch @bind-Checked="isChecked"></SfSwitch>
@code {
    private bool isChecked = true;
}
```

Run the application

After successful compilation of the application, simply press F5 to run the application. The Blazor Toggle Switch Button component will render in the web browser as shown below.



Set text on Switch

This section explains how to set `OnLabel` and `OffLabel` texts on Switch. In the following example, `onLabel` is set as `ON` and `offLabel` is set as `OFF`.

ASPX-CS

```
<SfSwitch @bind-Checked="isChecked" OnLabel="On" OffLabel="Off"></SfSwitch>
@code {
    private bool isChecked = true;
}
```

Switch does not have text support for material themes, and does not support long custom text.



See Also

- [Getting Started with Syncfusion Blazor for Client-Side in .NET Core CLI](#)
- [Getting Started with Syncfusion Blazor for Server-Side in Visual Studio](#)
- [Getting Started with Syncfusion Blazor for Server-Side in .NET Core CLI](#)

Events in Blazor Toggle Switch Button Component

ValueChanged Event

The `ValueChanged` event will trigger when switch state has been changed by user interaction.

CSHARP

```
@using Syncfusion.Blazor.Buttons
<SfSwitch @bind-Checked="isChecked" OffLabel="OFF" OnLabel="ON"
    ValueChange="Change" TChecked="bool?" ></SfSwitch>
@code{
    private bool? isChecked = null;
    private void Change(Syncfusion.Blazor.Buttons.ChangeEventArgs<bool?> args)
    {
        // Your code here.
    }
}
```

Toggle Switch Button has support for nullable boolean

Native Events

The native event can be defined using `event` attribute in component. The value of attribute is treated as an event handler. The event specific data will be available in event arguments.

The different event argument types for each event are,

- Focus Events - `UIFocusEventArgs`
- Mouse Events - `UIMouseEventArgs`
- Keyboard Events - `UIKeyboardEventArgs`
- Touch Events – `UITouchEventArgs`

List of Native events supported

The following native event support has been provided to the Toggle Switch Button component:

| List of Native events | | | | |

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

| onfocus | onfocusout | onfocusin | onkeydown | onkeyup | Onkeypress |

How to bind focus event to Toggle Switch Button

The `onfocus` attribute is used to bind the focus event for Toggle Switch Button. Here, we have explained about the sample code snippets of Toggle Switch Button.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
<label>onfocus</label>
<SfSwitch @onfocus="onFocus" @bind-Checked="isChecked"></SfSwitch>
@code {
    private bool isChecked = true;
}
@code {
    private void onFocus(Microsoft.AspNetCore.Components.Web.FocusEventArgs
args)
    {
        //onfocus Event triggered
    }
}
```

Accessibility in Blazor Toggle Switch Button Component

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.

Toggle Switch Button provides built-in compliance with `WAI-ARIA` specifications. `WAI-ARIA` support is achieved through the attributes like `aria-checked` and `aria-disabled`. It helps the people with disabilities by providing information about the widget for assistive technology in the screen readers.

| Properties | Functionality |

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

| role | Indicates the Toggle Switch Button component. |

| aria-checked | Indicates whether the input is checked or unchecked. |

| aria-disabled | Indicates that the element is perceivable but disabled, so it is not editable or otherwise operable. |

Keyboard interaction

<!-- markdownlint-disable MD033 -->

Keyboard shortcuts	Actions
Space	When the Toggle Switch Button has focus, pressing the Space key changes the state of the Toggle Switch Button.

Styles and Appearances in Blazor Toggle Switch Button Component

To modify the Switch appearance, the default CSS of Switch component has to be overridden. Please find the list of CSS classes and its corresponding section in Switch. Also, you have an option to create your own custom theme for the controls using our [Theme Studio](#).

| CSS Class | Purpose of Class |

| ---- | ---- |

| .e-switch-wrapper .e-switch-inner | To customize the line of the switch in off mode. |

| .e-switch-wrapper .e-switch-handle | To customize the handle of the switch in off mode. |

| .e-switch-wrapper:not(.e-switch-disabled):hover .e-switch-handle:not(.e-switch-active) | To customize the handle of the switch in off mode when hover. |

| .e-switch-wrapper:not(.e-switch-disabled):hover .e-switch-inner:not(.e-switch-active) | To customize the line of the switch in off mode when hover. |

| .e-switch-wrapper .e-switch-handle.e-switch-active | To customize the handle of the switch in on mode. |

| .e-switch-wrapper .e-switch-on | To customize the line of the switch in on mode. |

| .e-switch-wrapper:hover .e-switch-handle.e-switch-active | To customize the handle of the switch in on mode when hover. |

| .e-switch-wrapper:hover .e-switch-inner.e-switch-active .e-switch-on | To customize the line of the switch in on mode when hover. |

How To

Change Size in Blazor Toggle Switch Button Component

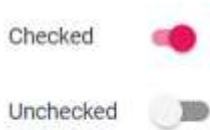
The different Toggle Switch Button sizes available are default and small. To reduce the size of default Toggle Switch Button to small,

set the [CssClass](#) property to `e-small`.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
```

```
<label for='switch1'>Checked</label>
<SfSwitch CssClass="e-small" @bind-Checked="isSmallChecked"></SfSwitch>
<label for='switch2'>Unchecked</label>
<SfSwitch @bind-Checked="isChecked"></SfSwitch>
@code {
private bool isSmallChecked = true;
private bool isChecked = false;
}
```



Customize the appearance of a Blazor Toggle Switch Button Component

The appearance of the Toggle Switch Button component can be customized using the CSS rules. Define your own CSS rules according to your requirement and assign the class name to the [CssClass](#) property.

Customize Toggle Switch Button bar and handle

Toggle Switch Button bar and handle can be customized as per requirement using CSS rules. Toggle Switch Button bar and handle customized using [CssClass](#) property. In the following sample, the [border-radius](#) CSS property for `e-switch-inner` and `e-switch-handle` elements was changed border radius circle to square shape.

For this customization, refer the `fabric.css` file. This could be found from our [CDN](#) link.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
<SfSwitch CssClass="square" @bind-Checked="isSquareChecked"></SfSwitch><br />
<SfSwitch CssClass="custom-switch" @bind-Checked="isCustomChecked"></SfSwitch><br />
<SfSwitch CssClass="handle-text" @bind-Checked="isHandleChecked"></SfSwitch>
@code {
private bool isSquareChecked = true;
private bool isCustomChecked = false;
private bool isHandleChecked = false;
}
<style>
/* Square Switch */
.e-switch-wrapper.square .e-switch-inner,
.e-switch-wrapper.square .e-switch-handle {
border-radius: 0;
}
/* Customize Handle and Bar Switch */
.e-switch-wrapper.custom-switch {
width: 50px;
height: 24px;
}
.e-switch-wrapper.custom-switch .e-switch-handle {
width: 20px;
height: 16px;
```

```

}
.e-switch-wrapper.custom-switch .e-switch-inner,
.e-switch-wrapper.custom-switch .e-switch-handle {
border-radius: 0;
}
.e-switch-wrapper.custom-switch .e-switch-handle.e-switch-active {
left: 42px;
}
/* Customize Handle and Bar Switch */
.e-switch-wrapper.handle-text {
width: 58px;
height: 24px;
}
.e-switch-wrapper.handle-text .e-switch-handle {
width: 26px;
height: 20px;
left: 2px;
background-color: #fff;
}
.e-switch-wrapper.handle-text .e-switch-inner,
.e-switch-wrapper.handle-text .e-switch-handle {
border-radius: 0;
}
.e-switch-wrapper.handle-text .e-switch-handle.e-switch-active {
left: 46px;
}
.e-switch-wrapper.handle-text .e-switch-inner.e-switch-active,
.e-switch-wrapper.handle-text:hover .e-switch-inner.e-switch-active .e-switch-on {
background-color: #4d841d;
border-color: #4d841d;
}
.e-switch-wrapper.handle-text .e-switch-inner,
.e-switch-wrapper.handle-text .e-switch-off {
background-color: #e3165b;
border-color: #e3165b;
}
.e-switch-wrapper.handle-text .e-switch-inner:after,
.e-switch-wrapper.handle-text .e-switch-inner:before {
font-size: 10px;
position: absolute;
line-height: 21px;
font-family: "Helvetica", sans-serif;
z-index: 1;
height: 100%;
transition: all 200ms cubic-bezier(0.445, 0.05, 0.55, 0.95);
}
.e-switch-wrapper.handle-text .e-switch-inner:before {
content: "OFF";
color: #e3165b;
left: 3px;
}
.e-switch-wrapper.handle-text .e-switch-inner:after {
content: "ON";
right: 5px;
color: #fff;
}
}

```

```
.e-switch-wrapper.handle-text .e-switch-inner.e-switch-active:before {
color: #fff;
}
.e-switch-wrapper.handle-text .e-switch-inner.e-switch-active:after {
color: #4d841d;
}
.e-switch-wrapper.handle-text:not(.e-switch-disabled):hover .e-switch-
handle:not(.e-switch-active) {
background-color: #fff;
}
</style>
```



Color the Toggle Switch Button

Toggle Switch Button colors can be customized as per the requirement using CSS rules. Toggle Switch Button bar and handle colors customized using [CssClass](#) property. In the following sample, the `e-switch-inner` and `e-switch-off` elements background and border colors were changed from default colors.

For this customization you need to refer the `bootstrap.css` file. This could be found from our [CDN](#) link.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
<SfSwitch CssClass="bar-color" @bind-Checked="isBarCheked"></SfSwitch><br />
<SfSwitch CssClass="handle-color" @bind-Checked="isHandleChecked"></SfSwitch><br />
<SfSwitch CssClass="custom-iOS" @bind-Checked="isCustomChecked"></SfSwitch>
@code {
private bool isBarCheked = false;
private bool isHandleChecked = true;
private bool isCustomChecked = true;
}
<style>
/* Custom color Switch */
.e-switch-wrapper.bar-color .e-switch-inner.e-switch-active,
.e-switch-wrapper.bar-color:hover .e-switch-inner.e-switch-active .e-switch-
on {
background-color: #4d841d;
border-color: #4d841d;
}
.e-switch-wrapper.bar-color .e-switch-inner,
.e-switch-wrapper.bar-color .e-switch-off {
background-color: #e3165b;
```



```
border-color: #e3165b;
}
.e-switch-wrapper.bar-color .e-switch-handle {
background-color: #fff;
}
/* handle color Switch */
.e-switch-wrapper.handle-color .e-switch-handle {
background-color: #e3165b;
}
.e-switch-wrapper.handle-color .e-switch-handle.e-switch-active {
background-color: #4d841d
}
.e-switch-wrapper.handle-color .e-switch-inner.e-switch-active,
.e-switch-wrapper.handle-color:hover .e-switch-inner.e-switch-active .e-
switch-on {
background-color: #fff;
border-color: #ccc;
}
.e-switch-wrapper.handle-color .e-switch-inner,
.e-switch-wrapper.handle-color .e-switch-off {
background-color: #fff;
border-color: #ccc;
}
/* iOS Switch */
.e-switch-wrapper.custom-iOS .e-switch-inner.e-switch-active,
.e-switch-wrapper.custom-iOS:hover .e-switch-inner.e-switch-active .e-
switch-on {
background-color: #3df865;
border-color: #3df665;
}
.e-switch-wrapper.custom-iOS {
width: 42px;
height: 24px;
}
.e-switch-wrapper.custom-iOS .e-switch-handle {
width: 20px;
height: 20px;
}
.e-switch-wrapper.custom-iOS .e-switch-handle.e-switch-active {
margin-left: -22px;
}
</style>
```



Enable RTL in Blazor Toggle Switch Button Component

Toggle Switch 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 Toggle Switch Button component.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
<SfSwitch EnableRtl="true" @bind-Checked="isChecked"></SfSwitch>
@code {
    private bool isChecked = false;
}
```



Change Blazor Toggle Switch Button state using toggle method

This section explains about how to toggle between the Toggle Switch Button states using [Toggle](#) method.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
<SfSwitch @bind-Checked="isChecked" OffLabel="OFF" OnLabel="ON"
    Created="create" @ref="SwitchObj" TChecked="bool"></SfSwitch>
@code{
    private bool isChecked = false;
    SfSwitch<bool> SwitchObj;
    private void create(object obj)
    {
        //SwitchObj.Toggle();
    }
}
```



Switch triggers [OnChange](#) event on every state stage to perform custom operations.

Set disabled state in Blazor Toggle Switch Button Component

Toggle Switch Button can be disabled by setting the [Disabled](#) property to true.

The following example illustrates how to disable support in Toggle Switch Button component.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
<SfSwitch Disabled="true" @bind-Checked="isChecked"></SfSwitch>
@code {
private bool isChecked = false;
}
```



Model binding in Blazor Toggle Switch Button Component

This section demonstrates the strongly typed extension support in Toggle Switch Button. The view that can bind with any model is called as strongly typed view. You can bind any class as model to view. The model properties can be accessed on that view. The data associated with model can be used to render the component.

In this sample, first check the option and click the submit button to post the selected value in the Toggle Switch Button. When value is not checked, validation error message will be shown below the Toggle Switch Button.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons
@using System.ComponentModel.DataAnnotations
<EditForm Model="Annotate">
<DataAnnotationsValidator></DataAnnotationsValidator>
<div class="form-group">
<SfSwitch @bind-Checked="@Annotate.Check">I agree to receive
newsletter</SfSwitch>
<ValidationMessage For="@(() => Annotate.Check)" />
</div>
<SfButton HtmlAttributes="@Submit" Content="Submit"
IsPrimary="true"></SfButton>
</EditForm>
@code {
public Annotation Annotate = new Annotation();
public class Annotation
{
[Range(typeof(bool), "true", "true", ErrorMessage = "You need to agree to
receive newsletter")]
public bool Check { get; set; }
}
public Dictionary<string, object> Submit = new Dictionary<string, object>()
{
{ "type", "submit"}
};
}
```



Toolbar

Getting Started with Blazor Toolbar Component

This section briefly explains about how to include a **Toolbar** in the Blazor server-side application. Refer [Getting Started with Syncfusion Blazor for Server-Side in Visual Studio](#) page for the introduction and configuring the common specifications.

To get started quickly with Blazor Toolbar, check on this video:

{% youtube

"youtube:https://www.youtube.com/watch?v=3brT3ztZErE"%}

Importing Syncfusion Blazor component in the application

Use any one of the below standards to install the Syncfusion Blazor library in the application.

Using Syncfusion Blazor individual NuGet Packages [New standard]

Starting with Volume 4, 2020 (v18.4.0.30) release, Syncfusion provides [individual NuGet packages](#) for the Syncfusion Blazor components. This new standard is highly recommended for the Blazor production applications. Refer to [this section](#) to know the benefits of the individual NuGet packages.

1. Install **Syncfusion.Blazor.Navigations** NuGet package to the application using the **NuGet Package Manager**.
2. Add the client-side style resources through [CDN](#) or from [NuGet](#) package in the `~/Pages/_Host.cshtml` page.

HTML

```
<head>
....
....
<link href="_content/Syncfusion.Blazor.Themes/bootstrap4.css"
rel="stylesheet" />
</head>
```

Waring: Syncfusion.Blazor package should not be installed along with the [individual NuGet packages](#). Hence, the above **Syncfusion.Blazor.Themes** static web assets (styles) have to be added in the application.

Using Syncfusion.Blazor NuGet Package [Old standard]

Waring: If the above new standard (individual NuGet packages) is preferred, then skip this section. Using both old and new standards in the same application will throw ambiguous compilation errors.

1. Install **Syncfusion.Blazor** NuGet package to the newly created application by using the **NuGet Package Manager**.
2. The client-side style resources can be added through [CDN](#) or from [NuGet](#) package in the element of the `~/Pages/_Host.cshtml` page.

HTML

```
<head>
....
....
<link href="_content/Syncfusion.Blazor/styles/bootstrap4.css"
rel="stylesheet" />
@*<link href="https://cdn.syncfusion.com/blazor/{{ site.blazorversion
}}/styles/bootstrap4.css" rel="stylesheet" />*@
</head>
```

For Internet Explorer 11 kindly refer the polyfills. Refer the [documentation](#) for more information.

HTML

```
<head>
...
<link href="https://cdn.syncfusion.com/blazor/{{ site.blazorversion
}}/styles/bootstrap4.css" rel="stylesheet" />
<script
src="https://github.com/Daddoon/Blazor.Polyfill/releases/download/3.0.1/blaz
or.polyfill.min.js"></script>
...
</head>
```

Adding component package to the application

Open `~/_Imports.razor` file and import the **Syncfusion.Blazor.Navigations** package.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
```

Add SyncfusionBlazor service in Startup file

Open the **Startup.cs** file and add services required by Syncfusion components using **services.AddSyncfusionBlazor()** method. Add this method in the **ConfigureServices** function as follows.

CSHARP

```
using Syncfusion.Blazor;
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void ConfigureServices(IServiceCollection services)
        {
            ....
            ....
        }
    }
}
```

```
services.AddSyncfusionBlazor();  
}  
}
```

Adding Toolbar component to the application

Now, add the Syncfusion Blazor Toolbar component in any web page (razor) in the **Pages** folder. For example, the Toolbar component is added in the `~/Pages/Index.razor` page.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations  
<SfToolbar>  
<ToolbarItems>  
<ToolbarItem Text="Cut"></ToolbarItem>  
<ToolbarItem Text="Copy"></ToolbarItem>  
<ToolbarItem Text="Paste"></ToolbarItem>  
//To separate the Toolbar items  
<ToolbarItem></ToolbarItem>  
<ToolbarItem Text="Bold"></ToolbarItem>  
<ToolbarItem Text="Underline"></ToolbarItem>  
<ToolbarItem Text="Italic"></ToolbarItem>  
<ToolbarItem Text="Color-Picker"></ToolbarItem>  
</ToolbarItems>  
</SfToolbar>
```

Run the application

After successful compilation of the application, simply press **F5** to run the application.



Cut Copy Paste Bold Underline Italic Color-Picker

See Also

1. [Getting Started with Syncfusion Blazor for client-side in .NET Core CLI](#)
2. [Getting Started with Syncfusion Blazor for client-side in Visual Studio 2019](#)
3. [Getting Started with Syncfusion Blazor for server-side in .NET Core CLI](#)

Item Configuration in Blazor Toolbar Component

The Toolbar can be rendered by defining a list of items. Items can be constructed with the following built-in command types or item template.

- Align
- CssClass
- Disabled
- HtmlAttributes
- Id
- Overflow
- PrefixIcon

- ShowAlwaysInPopup
- ShowTextOn
- SuffixIcon
- Template
- Text
- TooltipText
- Type
- Visible
- Width

Align

It specifies the location for aligning Toolbar items on the Toolbar. Each command will be aligned according to the **Align** property. The possible values are:

1. **Left:** To align commands to the left side of the Toolbar.
2. **Center:** To align commands at the center of the Toolbar.
3. **Right:** To align commands to the right side of the Toolbar.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<SfToolbar>
<ToolbarItems>
<ToolbarItem Text="Home" Align="ItemAlign.Left"></ToolbarItem>
<ToolbarItem Text="My Home Page" Align="ItemAlign.Center"></ToolbarItem>
<ToolbarItem Text="Search" Align="ItemAlign.Right"></ToolbarItem>
<ToolbarItem Text="Settings" Align="ItemAlign.Right"></ToolbarItem>
</ToolbarItems>
</SfToolbar>
```



CssClass

Single or multiple CSS classes can be added to the Toolbar commands using the Toolbar item **CssClass** property. Refer [Set command customization](#) for snippet and output.

Disabled

It specifies whether an item should be disabled or not. Refer [Enable/Disable Toolbar item](#) for snippet and output.

HtmlAttributes

It is used to add custom attributes to the Toolbar command. Supports HTML attributes such as style, class, etc. Refer [Set command customization](#) for snippet and output.

Id

It specifies the unique ID to be used with button or input element of the Toolbar items.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<SfToolbar>
```

```
<ToolbarItems>
<ToolbarItem Id="Home" Text="Home" Align="ItemAlign.Left"></ToolbarItem>
<ToolbarItem Text="My Home Page" Align="ItemAlign.Center"></ToolbarItem>
<ToolbarItem Text="Search" Align="ItemAlign.Right"></ToolbarItem>
<ToolbarItem Text="Settings" Align="ItemAlign.Right"></ToolbarItem>
</ToolbarItems>
</SfToolbar>
```



Overflow

It Specifies the Toolbar command display area when an element's content is too large to fit available space. This is applicable only to **Popup** mode. The possible values are:

1. **Show**: Always shows the item as the primary priority on the Toolbar.
2. **Hide**: Always shows the item as the secondary priority on the popup.
3. **None**: No priority for display, and as per normal order moves to popup when content exceeds.

Refer [Responsive Mode](#) for snippet and output.

PrefixIcon

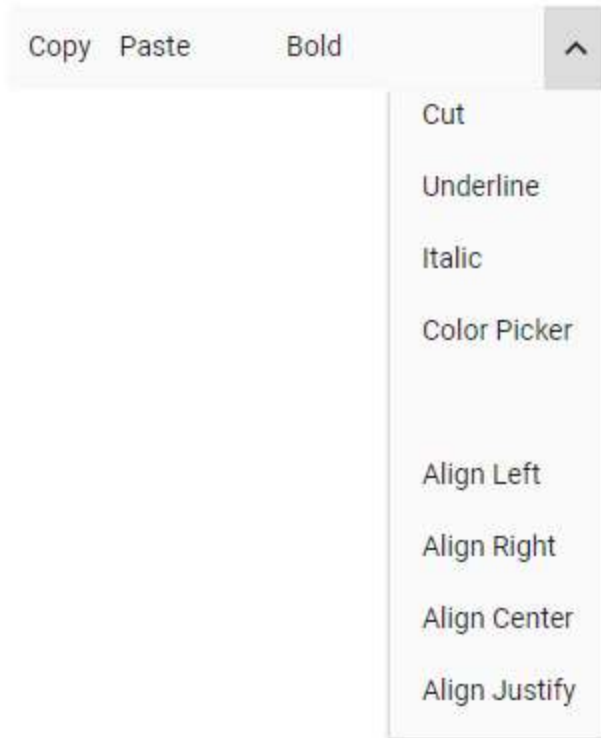
It defines single or multiple classes separated by space used to specify an icon for the button. The icon will be positioned before the text content if the text is available, otherwise the icon alone will be rendered. Refer [Customize the Scrolling distance](#) for snippet and output.

ShowAlwaysInPopup

It defines the priority of items to display it in popup always. It allows to maintain toolbar item on popup always but it does not work for the toolbar priority items.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<SfToolbar Width="300px" OverflowMode="OverflowMode.Popup">
<ToolbarItems>
<ToolbarItem Text="Cut" ShowAlwaysInPopup=true
TooltipText="Cut"></ToolbarItem>
<ToolbarItem Text="Copy" TooltipText="Copy"></ToolbarItem>
<ToolbarItem Text="Paste" TooltipText="Paste"></ToolbarItem>
<ToolbarItem></ToolbarItem>
<ToolbarItem Text="Bold" TooltipText="Bold"></ToolbarItem>
<ToolbarItem Text="Underline" TooltipText="Underline"></ToolbarItem>
<ToolbarItem Text="Italic" TooltipText="Italic"></ToolbarItem>
<ToolbarItem Text="Color Picker" TooltipText="Color-Picker"></ToolbarItem>
<ToolbarItem></ToolbarItem>
<ToolbarItem Text="Align Left" TooltipText="Align-Left"></ToolbarItem>
<ToolbarItem Text="Align Right" TooltipText="Align-Right"></ToolbarItem>
<ToolbarItem Text="Align Center" TooltipText="Align-Center"></ToolbarItem>
<ToolbarItem Text="Align Justify" TooltipText="Align-Justify"></ToolbarItem>
</ToolbarItems>
</SfToolbar>
```

ShowTextOn

It specifies where the button text will be displayed on popup mode of the Toolbar. The possible values are:

1. **Toolbar:** Text will be displayed on Toolbar only.
2. **Overflow:** Text will be displayed only when content overflows to popup.
3. **Both:** Text will be displayed on popup and Toolbar.

Refer [Responsive Mode](#) for snippet and output.

SuffixIcon

It defines single/multiple classes separated by space used to specify an icon for the button. The icon will be positioned after the text content if text is available.

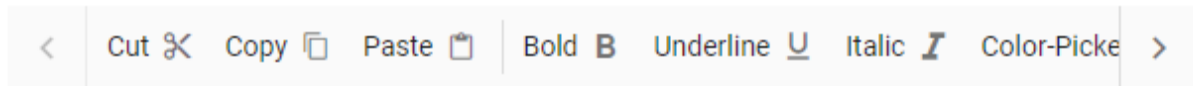
ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<SfToolbar Width="600">
<ToolbarItems>
<ToolbarItem SuffixIcon="e-icons e-cut" Text="Cut"></ToolbarItem>
<ToolbarItem SuffixIcon="e-icons e-copy" Text="Copy"></ToolbarItem>
<ToolbarItem SuffixIcon="e-icons e-paste" Text="Paste"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem SuffixIcon="e-icons e-bold" Text="Bold"></ToolbarItem>
<ToolbarItem SuffixIcon="e-icons e-underline"
Text="Underline"></ToolbarItem>
<ToolbarItem SuffixIcon="e-icons e-italic" Text="Italic"></ToolbarItem>
<ToolbarItem SuffixIcon="e-icons e-paint-bucket" Text="Color-
Picker"></ToolbarItem>
```

```

<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem SuffixIcon="e-icons e-list-unordered"
Text="Bullets"></ToolbarItem>
<ToolbarItem SuffixIcon="e-icons e-list-ordered"
Text="Numbering"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem SuffixIcon="e-icons e-sort-ascending" Text="Sort A -
Z"></ToolbarItem>
<ToolbarItem SuffixIcon="e-icons e-sort-descending" Text="Sort Z -
A"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem SuffixIcon="e-icons e-upload-1" Text="Upload"></ToolbarItem>
<ToolbarItem SuffixIcon="e-icons e-download" Text="Download"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem SuffixIcon="e-icons e-increase-indent" Text="Text
Indent"></ToolbarItem>
<ToolbarItem SuffixIcon="e-icons e-decrease-indent" Text="Text
Outdent"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem SuffixIcon="e-icons e-erase" Text="Clear"></ToolbarItem>
<ToolbarItem SuffixIcon="e-icons e-refresh" Text="Reload"></ToolbarItem>
<ToolbarItem SuffixIcon="e-icons e-export" Text="Export"></ToolbarItem>
</ToolbarItems>
</SfToolbar>

```



you can refer [here](#) to integrate the syncfusion icons in toolbar component.

Template

The Toolbar supports adding template commands using the `Template` property. Template property can be given as the `HTML element` or `RenderFragment`. Refer [Set item-wise custom template](#) for snippet and output.

Text

It is used to specify the text to be displayed on the Toolbar button. Refer [Getting Started](#) for the snippet and output.

TooltipText

It is used to specify the text to be displayed on hovering the Toolbar button.

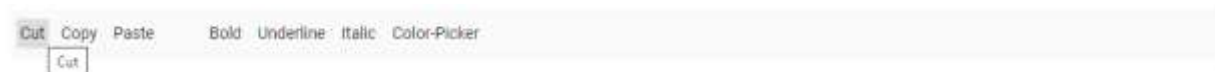
ASPX-CS

```

@using Syncfusion.Blazor.Navigations
<SfToolbar>
<ToolbarItems>
<ToolbarItem Text="Cut" TooltipText="Cut"></ToolbarItem>
<ToolbarItem Text="Copy" TooltipText="Copy"></ToolbarItem>
<ToolbarItem Text="Paste" TooltipText="Paste"></ToolbarItem>
//To separate the Toolbar items
<ToolbarItem></ToolbarItem>
<ToolbarItem Text="Bold" TooltipText="Bold"></ToolbarItem>
<ToolbarItem Text="Underline" TooltipText="Underline"></ToolbarItem>

```

```
<ToolbarItem Text="Italic" TooltipText="Italic"></ToolbarItem>
<ToolbarItem Text="Color-Picker" TooltipText="Color Picker"></ToolbarItem>
</ToolbarItems>
</SfToolbar>
```



Type

It specifies the types of command to be rendered in the Toolbar. Supported types are:

1. **Button:** Creates the Button control with its given properties like text, prefixIcon, etc.
2. **Separator:** Adds a horizontal line that separates the Toolbar commands.
3. **Input:** Creates an input element that is applicable to template rendering with Syncfusion controls like DropDownList, AutoComplete, etc.

Button

Button is the default command [Type](#), and it can be rendered by using the **Text** property.

Properties of the button command type:

Property | Description

Text | The text to be displayed for the button.

ID | The ID of the button to be rendered. If the ID is not given, auto ID is generated.

PrefixIcon | Defines the class used to specify an icon for the button. The icon is positioned before the text if the text is available or the icon alone button is rendered.

SuffixIcon | Defines the class used to specify an icon for the button. The icon is positioned after the text if text is available. If both [PrefixIcon](#) and [SuffixIcon](#) are specified, only the [PrefixIcon](#) is considered.

Width | Used to set the width of the button.

Refer [Set Blazor Tooltip to the commands](#) for snippet and output.

Separator

The **Separator** type adds a vertical separation between the Toolbar's Single/Multiple commands.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<SfToolbar>
<ToolbarItems>
<ToolbarItem Text="Cut"></ToolbarItem>
<ToolbarItem Text="Copy"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem Text="Paste"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem Text="Undo"></ToolbarItem>
<ToolbarItem Text="Redo"></ToolbarItem>
</ToolbarItems>
</SfToolbar>
```

If **Separator** is added as the first or the last item, it will not be visible.



Input

The **Input** type is only applicable for adding **Template** elements when the **Template** property is defined as an object. Input type creates an input element internally that acts as the container for Syncfusion input based components.

NumericTextBox

- The **NumericTextBox** component can be included by importing the **@using Syncfusion.Blazor.Inputs** package into the **~/_Imports.razor** file.
- Initialize the **NumericTextBox** in **Template** property, where the Toolbar item **Type** is set as **Input**.
- Related **NumericTextBox** component properties can also be configured as given below.

CSHARP

```
<SfNumericTextBox Format="n2"></SfNumericTextBox>
```

DropDownList

- The **DropDownList** component can be included by importing the **@using Syncfusion.Blazor.DropDowns** package into the **~/_Imports.razor** file.
- Initialize the **DropDownList** in **Template** property, where the Toolbar item **Type** is set as **Input**.
- Related **DropDownList** component properties can also be configured as given below.

CSHARP

```
<SfDropDownList Width="100"></SfDropDownList>
```

RadioButton

- The **RadioButton** component can be included by importing the **@using Syncfusion.Blazor.Buttons** package into the **~/_Imports.razor** file.
- Initialize the **RadioButton** in **Template** property, where the Toolbar item **Type** is set as **Input**.
- Related **RadioButton** component properties can also be configured as given below.

CSHARP

```
<SfRadioButton Label="Option 1" Name="Default"></SfRadioButton>
```

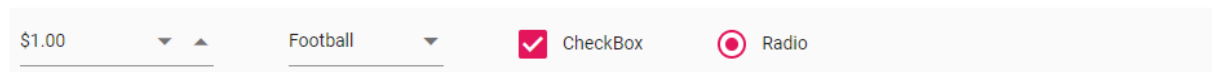
The following code explains how to add **NumericTextBox**, **DropDownList**, **RadioButton** and **CheckBox** components to the Toolbar.

ASPX-CS

```

@using Syncfusion.Blazor.Navigations
@using Syncfusion.Blazor.Inputs
@using Syncfusion.Blazor.DropDowns
@using Syncfusion.Blazor.Buttons
<SfToolbar>
<ToolbarItems>
<ToolbarItem Type="ItemType.Input">
<Template>
<SfNumericTextBox Width="150" Value=1 Format="c2"></SfNumericTextBox>
</Template>
</ToolbarItem>
<ToolbarItem></ToolbarItem>
<ToolbarItem Type="ItemType.Input">
<Template>
<SfDropDownList TValue="string" ID="Games" DataSource="@SportsData"
Width="120" Index="2" TItem="Games">
<DropDownListFieldSettings Value="Text"></DropDownListFieldSettings>
</SfDropDownList>
</Template>
</ToolbarItem>
<ToolbarItem></ToolbarItem>
<ToolbarItem Type="ItemType.Input">
<Template>
<SfCheckBox Checked="true" Label="CheckBox"></SfCheckBox>
</Template>
</ToolbarItem>
<ToolbarItem></ToolbarItem>
<ToolbarItem Type="ItemType.Input">
<Template>
<SfRadioButton Label="Radio" Name="default" Value="Radio" @bind-
Checked="CheckedValue"></SfRadioButton>
</Template>
</ToolbarItem>
</ToolbarItems>
</SfToolbar>
@code {
private string CheckedValue = "Radio";
public class Games
{
public string Id { get; set; }
public string Text { get; set; }
}
List<Games> SportsData = new List<Games> {
new Games() { Id= "Game2", Text= "Badminton" },
new Games() { Id= "Game4", Text= "Cricket" },
new Games() { Id= "Game5", Text= "Football" },
new Games() { Id= "Game6", Text= "Golf" },
new Games() { Id= "Game10", Text= "Tennis" }
};
}

```

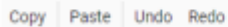


Visible

It specifies whether an item should be hidden or not.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<SfToolbar>
<ToolbarItems>
<ToolbarItem Text="Cut" Visible=false></ToolbarItem>
<ToolbarItem Text="Copy"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem Text="Paste"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem Text="Undo"></ToolbarItem>
<ToolbarItem Text="Redo"></ToolbarItem>
</ToolbarItems>
</SfToolbar>
```



Width

It specifies the width of the Toolbar button commands.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<SfToolbar>
<ToolbarItems>
<ToolbarItem Text="Cut" Width="50px"></ToolbarItem>
<ToolbarItem Text="Copy"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem Text="Paste"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem Text="Undo"></ToolbarItem>
<ToolbarItem Text="Redo"></ToolbarItem>
</ToolbarItems>
</SfToolbar>
```



Responsive Mode in Blazor Toolbar Component

This section explains the supported display modes of the Toolbar when the content exceeds the viewing area. Possible modes are:

- Scrollable
- Popup
- MultiRow
- Extended

Scrollable

The default [OverflowMode](#) of the Toolbar is **Scrollable**. Scrollable display mode supports display of commands in a single line with horizontal scrolling enabled when commands overflow to available space.

- The right and left navigation arrows are added to the start and end of the Toolbar to navigate to hidden commands.
- The hidden commands can also be seen using touch swipe action.
- By default, if navigation icon in the **left** side is disabled, see the hidden commands by moving to the **right**.
- By clicking the arrow or by holding the arrow continuously, hidden commands will become visible.
- If device navigation icons are not available, touch swipe to see the hidden commands of the Toolbar.
- Once the Toolbar reaches the last or first command, the corresponding navigation icon will be disabled and it can be moved to the opposite direction.



- The Toolbar content can be continuously scrolled by holding the navigation icon.



ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<SfToolbar Width="600">
<ToolbarItems>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-cut"
Text="Cut"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-copy"
Text="Copy"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-paste"
Text="Paste"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-bold"
Text="Bold"></ToolbarItem>
```

```

<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-underline"
Text="Underline"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-italic"
Text="Italic"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-paint-bucket"
Text="Color-Picker"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-list-unordered"
Text="Bullets"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-list-ordered"
Text="Numbering"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-sort-ascending"
Text="Sort A - Z"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-sort-descending"
Text="Sort Z - A"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-upload-1"
Text="Upload"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-download"
Text="Download"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-increase-indent"
Text="Text Indent"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-decrease-indent"
Text="Text Outdent"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-erase"
Text="Clear"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-refresh"
Text="Reload"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-export"
Text="Export"></ToolbarItem>
</ToolbarItems>
</SfToolbar>

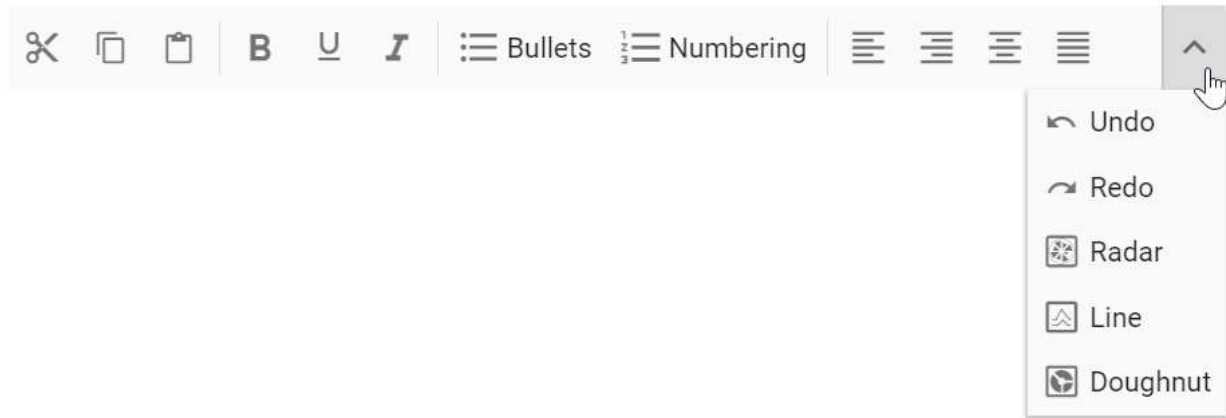
```



Popup

Popup is another type of **OverflowMode** in which the Toolbar container holds the commands that can be placed in the available space. The rest of the overflowing commands that do not fit within the viewing area moves to the overflow popup container.

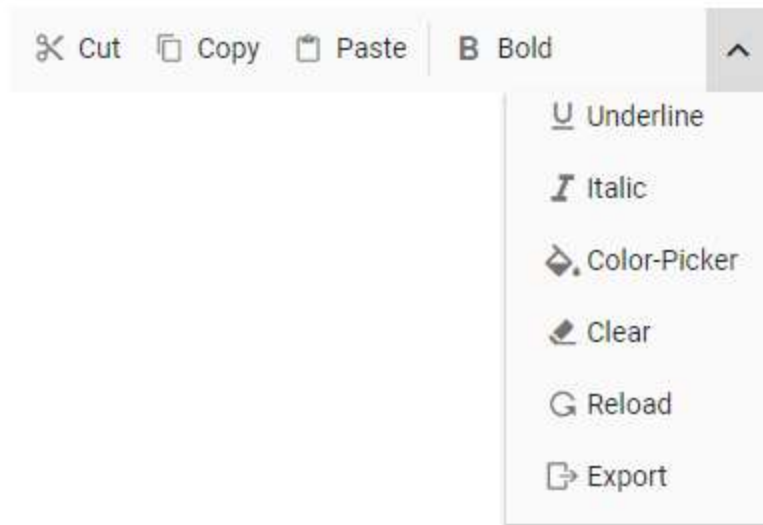
The commands placed in the popup can be viewed by opening the popup using the drop down icon given at the end of the Toolbar.



If the popup content overflows the height of the page, then the rest of the commands will be hidden.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<SfToolbar Width="380" OverflowMode="OverflowMode.Popup">
<ToolbarItems>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-cut"
Text="Cut"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-copy"
Text="Copy"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-paste"
Text="Paste"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-bold"
Text="Bold"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-underline"
Text="Underline"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-italic"
Text="Italic"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-paint-bucket"
Text="Color-Picker"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-erase"
Text="Clear"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-refresh"
Text="Reload"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-export"
Text="Export"></ToolbarItem>
</ToolbarItems>
</SfToolbar>
```



Priority of commands

Default popup priority is set as **None**, and when the commands of the Toolbar overflow, the ones listed last will be moved to the popup.

The priority of commands can be customized to be displayed on the Toolbar and popup by using the **Overflow** property.

Property | Description

Both | Button text is visible in both **Toolbar** and **Popup**.

Overflow | Button text is only visible in **Popup**.

Toolbar | Button text is only visible on the **Toolbar**.

In the following code sample, text is only visible in the popup container and not in the Toolbar container.

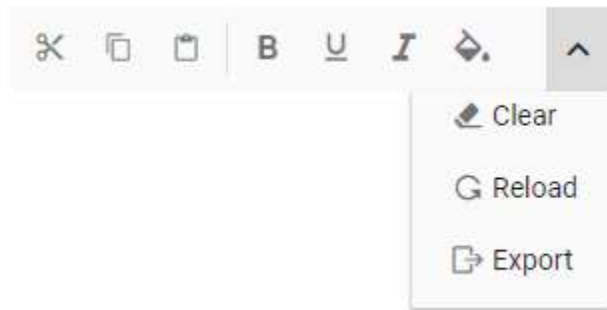
ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<SfToolbar Width="300" OverflowMode="OverflowMode.Popup">
<ToolbarItems>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-cut" Text="Cut"
ShowTextOn="DisplayMode.Overflow"
Overflow="OverflowOption.Show"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-copy" Text="Copy"
ShowTextOn="DisplayMode.Overflow"
Overflow="OverflowOption.Show"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-paste"
Text="Paste" ShowTextOn="DisplayMode.Overflow"
Overflow="OverflowOption.Show"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-bold" Text="Bold"
ShowTextOn="DisplayMode.Overflow"
Overflow="OverflowOption.Show"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-underline"
Text="Underline" ShowTextOn="DisplayMode.Overflow"
Overflow="OverflowOption.Show"></ToolbarItem>
```

```

<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-italic"
Text="Italic" ShowTextOn="DisplayMode.Overflow"
Overflow="OverflowOption.Show"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-paint-bucket"
Text="Color-Picker" ShowTextOn="DisplayMode.Overflow"
Overflow="OverflowOption.Show"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-erase"
Text="Clear" ShowTextOn="DisplayMode.Overflow"
Overflow="OverflowOption.Show"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-refresh"
Text="Reload" ShowTextOn="DisplayMode.Overflow"
Overflow="OverflowOption.Show"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-export"
Text="Export" ShowTextOn="DisplayMode.Overflow"
Overflow="OverflowOption.Show"></ToolbarItem>
</ToolbarItems>
</SfToolbar>

```



MultiRow

MultiRow is another type of **OverflowMode** in which the Toolbar container holds the commands that can be placed in the available space. The rest of the overflowing commands that do not fit within the viewing area will be displayed as an in-line of a toolbar.

ASPX-CS

```

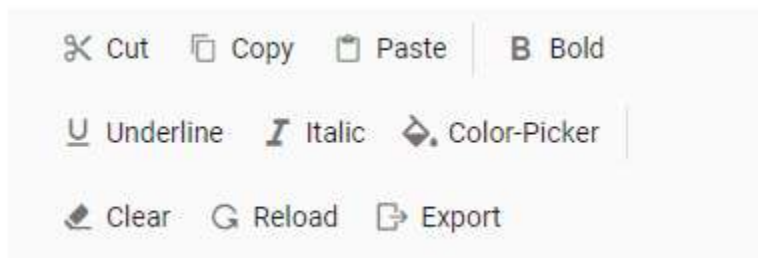
@using Syncfusion.Blazor.Navigations
<SfToolbar Width="380" OverflowMode="OverflowMode.MultiRow">
<ToolbarItems>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-cut"
Text="Cut"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-copy"
Text="Copy"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-paste"
Text="Paste"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-bold"
Text="Bold"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-underline"
Text="Underline"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-italic"
Text="Italic"></ToolbarItem>

```

```

<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-paint-bucket"
Text="Color-Picker"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-erase"
Text="Clear"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-refresh"
Text="Reload"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-export"
Text="Export"></ToolbarItem>
</ToolbarItems>
</SfToolbar>

```



Extended

Extended is another type of **OverflowMode** in which the Toolbar container holds the commands that can be placed in the available space. The rest of the overflowing commands that do not fit within the viewing area will be displayed in the next row.

ASPX-CS

```

@using Syncfusion.Blazor.Navigations
<SfToolbar Width="380" OverflowMode="OverflowMode.Extended">
<ToolbarItems>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-cut"
Text="Cut"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-copy"
Text="Copy"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-paste"
Text="Paste"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-bold"
Text="Bold"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-underline"
Text="Underline"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-italic"
Text="Italic"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-paint-bucket"
Text="Color-Picker"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-erase"
Text="Clear"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-refresh"
Text="Reload"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" PrefixIcon="e-icons e-export"
Text="Export"></ToolbarItem>
</ToolbarItems>
</SfToolbar>

```



Accessibility in Blazor Toolbar Component

The Toolbar component 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-orientation`, `aria-disabled` and `aria-haspopup`. It provides information about elements in a document for assistive technology. The component implements keyboard navigation support by following the [WAI-ARIA practices](#), and has been tested in major screen readers.

ARIA attributes

The Toolbar element is assigned the role of `toolbar`.

| Property | Functionalities |

| --- | --- |

| `role="toolbar"` | This attribute added to the ToolBar element describes the actual role of the element. |

| `aria-orientation` | Indicates the ToolBar orientation. Default value is `horizontal`. |

| `aria-haspopup` | Indicates the popup mode of the Toolbar. Default value is false. When popup mode is enabled, attribute value has to be changed to `true`. |

| `aria-disabled` | Indicates the disabled state of the ToolBar. |

Keyboard interaction

Keyboard navigation is enabled, by default. Possible keys are:

| Key | Description |

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

| Left | Focuses the previous element. |

| Right | Focuses the next element. |

| Enter | When focused on a ToolBar command, clicking the key triggers the click of Toolbar element. When popup drop-down icon is focused, the popup opens. |

| Esc(Escape) | Closes popup. |

| Down | Focuses the next popup element. |

| Up | Focuses the previous popup element. |

Style and Appearance in Blazor Toolbar Component

The following content provides the exact CSS structure that can be used to modify the control's appearance based on user preference.

Customizing Toolbar

Use the following CSS to customize the Toolbar.

CSS

```
.e-toolbar {  
border: 5px solid rgb(173, 255, 47);  
}
```

Customizing the Toolbar items

Use the following CSS to customize the items of Toolbar.

CSS

```
.e-toolbar .e-toolbar-item {  
background: #add8e6;  
border: 1px solid #5a70cc;  
}
```

Use the following CSS to customize the button in the items of Toolbar.

CSS

```
.e-toolbar .e-tbar-btn {  
background: #add8e6;  
border: 1px solid #5a70cc;  
}
```

Customizing Toolbar's item icon

Use the following CSS to customize the item icon of Toolbar control.

CSS

```
.e-toolbar .e-tbar-btn .e-icons {  
background: #185655;  
color: #d7f9d4;  
}
```

Customizing the hover state of Toolbar control

Use the following CSS to customize the toolbar item when hovering.

CSS

```
.e-toolbar .e-tbar-btn:hover {
background: #c0e3a1;
border: 1px solid green;
}
```

Customizing selected item of Toolbar control

Use the following CSS to customize the selected toolbar item.

CSS

```
.e-toolbar .e-tbar-btn:focus {
background: #add8e6;
border: 1px solid #5a70cc;
}
```

How To

Add/Remove Toolbar items in Blazor Toolbar Component

Toolbar items can be added or removed dynamically by iteration of Toolbar Items using conditional **foreach** loop.

In the following demo, initially there are five toolbar as the **ToolbarItems** has five items. On clicking the **Add Item** button, a new item is added to the **ToolbarItems** results in adding sixth toolbar to the Toolbar component. On clicking the **Remove Item**, the first item of the **ToolbarItems** has been removed which results in removing the first toolbar of the Toolbar component.

CSHARP

```
@using Syncfusion.Blazor.Navigations
@using Syncfusion.Blazor.Buttons
<SfButton @onclick="AddItemClick" Content="Add Item"></SfButton>
<SfButton @onclick="RemoveItemClick" Content="Remove Item"></SfButton>
<br />
<br />
<SfToolbar>
<ToolbarItems>
@foreach (ToolbarData Item in ToolbarItems)
{
<ToolbarItem PrefixIcon=@(Item.PrefixIcon) Text=@(Item.Text)
TooltipText=@(Item.TooltipText)></ToolbarItem>
}
</ToolbarItems>
</SfToolbar>
@code {
public class ToolbarData
{
public string PrefixIcon { get; set; }
public string Text { get; set; }
public string TooltipText { get; set; }
}
List<ToolbarData> ToolbarItems = new List<ToolbarData>()
```

```

{
    new ToolbarData
    {
        PrefixIcon = "e-icons e-cut",
        Text = "Cut",
        TooltipText = "Cut"
    },
    new ToolbarData
    {
        PrefixIcon = "e-icons e-copy",
        Text = "Copy",
        TooltipText = "Copy"
    },
    new ToolbarData
    {
        PrefixIcon = "e-icons e-paste",
        Text = "Paste",
        TooltipText = "Paste"
    },
    new ToolbarData
    {
        PrefixIcon = "e-icons e-bold",
        Text = "Bold",
        TooltipText = "Bold"
    },
    new ToolbarData
    {
        PrefixIcon = "e-icons e-underline",
        Text = "Underline",
        TooltipText = "Underline"
    }
};

void AddItemClick()
{
    ToolbarItems.Add(new ToolbarData
    {
        PrefixIcon = "e-icons e-italic",
        Text = "Italic",
        TooltipText = "Italic"
    });
}

void RemoveItemClick()
{
    if (ToolbarItems.Count > 0)
    {
        ToolbarItems.RemoveAt(0);
    }
}
}

```

ADD ITEM REMOVE ITEM

 Cut  Copy  Paste  Bold  Underline

Show/Hide Toolbar item in Blazor Toolbar Component

The **Visible** property of the Toolbar item is used to show or hide the item by setting true or false value to the property. In the following code example initially paste action will be hide. On clicking the cut button, the paste button will be show.

CSHARP

```
@using Syncfusion.Blazor.Navigations
<SfToolbar>
<ToolbarItems>
<ToolbarItem PrefixIcon="e-icons e-cut" OnClick="@OnItemClick" Text="Cut"
TooltipText="Cut"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-paste" Visible="@ShowItem" Text="Paste"
TooltipText="Paste"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-bold" Text="Bold"
TooltipText="Bold"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-underline" Text="Underline"
TooltipText="Underline"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-italic" Text="Italic"
TooltipText="Italic"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-align-left" Text="Left"
TooltipText="Align-Left"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-align-right" Text="Right"
TooltipText="Align-Right"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-align-center" Text="Center"
TooltipText="Align-Center"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-justify" Text="Justify"
TooltipText="Align-Justify"></ToolbarItem>
</ToolbarItems>
</SfToolbar>
@code{
public bool ShowItem { get; set; } = false;
public void OnItemClick()
{
ShowItem = !ShowItem;
}
}
```



Enable/Disable Toolbar item in Blazor Toolbar Component

The **Disabled** property of the toolbar item is used to enable or disable the item by setting false or true value to the property. In the following code example initially paste action will be disabled. On clicking the cut button, the paste button will be enabled.

CSHARP

```
@using Syncfusion.Blazor.Navigations
<SfToolbar>
<ToolbarItems>
<ToolbarItem PrefixIcon="e-icons e-cut" OnClick="@OnItemClick" Text="Cut"
TooltipText="Cut"></ToolbarItem>
```

```

<ToolbarItem PrefixIcon="e-icons e-paste" Disabled="@ShowIcon" Text="Paste"
TooltipText="Paste"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-bold" Text="Bold"
TooltipText="Bold"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-underline" Text="Underline"
TooltipText="Underline"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-italic" Text="Italic"
TooltipText="Italic"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-align-left" Text="Left"
TooltipText="Align-Left"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-align-right" Text="Right"
TooltipText="Align-Right"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-align-center" Text="Center"
TooltipText="Align-Center"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-justify" Text="Justify"
TooltipText="Align-Justify"></ToolbarItem>
</ToolbarItems>
</SfToolbar>
@code{
public bool ShowIcon { get; set; } = true;
public void OnItemClick()
{
ShowIcon = !ShowIcon;
}
}

```



Set command customization in Blazor Toolbar Component

The `HtmlAttributes` property of the Toolbar item is used to set the HTML attributes ('ID', 'class', 'style', 'role') for the commands.

When style attributes are added, if the same attributes were already present, they will be replaced. This is not so in the case of `class` attribute. Classes will be added to the element instead of replacing the existing ones.

Single or multiple CSS classes can be added to the Toolbar commands using the Toolbar item `CssClass` property.

CSHARP

```

@using Syncfusion.Blazor.Navigations
<SfToolbar Width="500">
<ToolbarItems>
<ToolbarItem Text="Bold" Type="ItemType.Button"
HtmlAttributes="@Item1"></ToolbarItem>
<ToolbarItem Text="Italic" Type="ItemType.Button"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" Text="Underline"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem Text="Uppercase" CssClass="e-txt-casing"></ToolbarItem>
</ToolbarItems>
</SfToolbar>

```

```
@code {
Dictionary<string, object> Item1 = new Dictionary<string, object>()
{
{ "class" , "custom_bold" },
{ "id" , "itemId" }
};
}
<style>
.custom_bold .e-tbar-btn-text {
font-weight: 900;
}
.e-txt-casing .e-tbar-btn-text {
font-variant: small-caps;
}
</style>
```



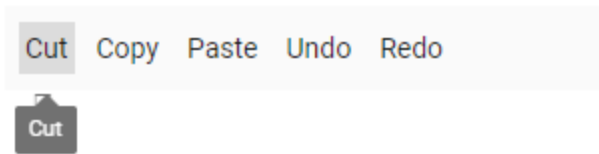
Set Tooltip to the commands in Blazor Toolbar Component

The **TooltipText** property of the Toolbar item is used to set the HTML Tooltip to the commands that can be viewed as hint texts on mouse hover.

Initialize the Tooltip with the Toolbar target.

CSHARP

```
@using Syncfusion.Blazor.Navigations
@using Syncfusion.Blazor.Popups
<SfTooltip ID="Tooltip" Target="#Element [title]">
<SfToolbar ID="Element" Width="300">
<ToolbarItems>
<ToolbarItem Type="ItemType.Button" Text="Cut"
TooltipText="Cut"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" Text="Copy"
TooltipText="Copy"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" Text="Paste"
TooltipText="Paste"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" Text="Undo"
TooltipText="Undo"></ToolbarItem>
<ToolbarItem Type="ItemType.Button" Text="Redo"
TooltipText="Redo"></ToolbarItem>
</ToolbarItems>
</SfToolbar>
</SfTooltip>
```



Set item-wise custom template in Blazor Toolbar Component

The Toolbar supports adding template commands using the `Template` property. Template property can be given as the `HTML element` or `RenderFragment`.

CSHARP

```
@using Syncfusion.Blazor.Navigations
<SfToolbar Width="300">
<ToolbarItems>
<ToolbarItem Text="Cut" TooltipText="Cut"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem Text="Paste"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem>
<Template>
<div><input type='checkbox' title="Accept" checked/>
</div>
</Template>
</ToolbarItem>
<ToolbarItem Text="Undo"></ToolbarItem>
<ToolbarItem Text="Redo"></ToolbarItem>
<ToolbarItem>
<Template>
<button id="template" class="e-btn">Template</button>
</Template>
</ToolbarItem>
</ToolbarItems>
</SfToolbar>
```



Add Toggle Button in Blazor Toolbar Component

Toolbar supports adding a toggle button by using the `Template` property.

- By using Toolbar template property, pass required HTML String to render the toggle button.

ASPX-CS

```
<Template>
<SfButton></SfButton>
</Template>
```

- Now render the toggle Button into the targeted element in the Toolbar created event handler and bind click event for it.

On clicking the toggle Button, change the required icon and content based on current active state.

CSHARP

```
@using Syncfusion.Blazor.Navigations
@using Syncfusion.Blazor.Buttons
<SfToolbar>
<ToolbarItems>
<ToolbarItem>
<Template>
<div><SfButton @ref="MediaButton" CssClass="e-flat" IconCss="@MediaIconCss"
IsToggle="true" @onclick="@OnMediaClick"></SfButton></div>
</Template>
</ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem>
<Template>
<div><SfButton @ref="ZoomButton" CssClass="e-flat" IconCss="@ZoomIconCss"
IsToggle="true" @onclick="@OnZoomClick"></SfButton></div>
</Template>
</ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem>
<Template>
<div><SfButton @ref="UndoButton" CssClass="e-flat" IconCss="@UndoIconCss"
IsToggle="true" @onclick="@OnUndoClick"></SfButton></div>
</Template>
</ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem>
<Template>
<div><SfButton @ref="FilterButton" CssClass="e-flat"
IconCss="@FilterIconCss" IsToggle="true"
@onclick="@OnFilterClick"></SfButton></div>
</Template>
</ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem>
<Template>
<div><SfButton @ref="VisibleButton" CssClass="e-flat"
IconCss="@VisibleIconCss" IsToggle="true" Content="@Content"
@onclick="@OnVisibleClick"></SfButton></div>
</Template>
</ToolbarItem>
</ToolbarItems>
</SfToolbar>
<br />
<div id="content" style="display:@DisplayValue">
This content will be hidden, when you click on hide button and toggle will
get an active state as shown, otherwise it will be visible.
</div>
<style>
@@font-face {
font-family: 'Toolbar';
```

```
src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKIAAAAwAgTlMvMvltCfsAAAEoAAAAVmNtYXCnKKeOAAABrAAAAEhnbHlm5Ce
ZPwAAAgwAAAPsaGVhZBKvqTUAADQAAAAANmhoZWEIUQQMAAAArAAAACRobXR4LAAAAAAYAAAAA
sbG9jYRGwFFwAAAH0AAAAGG1heHABGgFNAAABCAAAACBuYW1lQozdSwAADHgAAAIlcG9zdFkKJftQ
AAA6gAAAA1AABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAACwABAAAAQAAtluoWF8
PPPUACwQAAAAANfosiIAAAAA186yIgAAAAAD9AP0AAAACAACAAAAAAAAAAAAAALAUeABQAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAABQAAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABApwCnCQQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAAACAaaaaaWAAABQAAwABAAAAFAA
EADQAAAAEAAQAAQAApwn//wAApWd//wAAAAEABAAAAEAAgADAAQABQAGAAcACAAJAAoAAAAAATw
CsAMyA6wDugPQBGE+gUkBTYABAAAAAAD8wMoAEAAgQDHARwAAAE7AR8ODw8vDjU/DgcdAR8OPw8
1Lw4PDjcfEA8OKwEvDj8SMycPEhUfEz8TNS8TIw8BAf8BDg4NDQ0LCwsJCAgFBQMCACQECawUGBwg
JCgsLDQ0NDg8PDg0NDQwLCgoIBwYEAwEBAwQFBgcICQoLCwwMDQ28AwQGCaoMDQ8RERITFBQVFRQ
UEXIQEA4NDAoIBwUDAQMFbWgLDa0PEBESEhMUExcTEXIREBAODQwKCQcGBOYODh0CHR0dHR8WfHY
VFRUUFb0eHx8fISAIhFhCWfHwUfHwUfHwUfHwUfHwUfHwUfHwUfHwUfHwUfHwUfHwUfHwUfHw
VFhYWFhYWFhcfHh4dHRwbGwYEAwEDAwQfICEhIiIkJBscGxsbGxsbIAwZGhkZGRogJSUkIyIiIRM
FAwICAwUZGBcYGBkZGSiHICEgICEgHBUWFQKMAgQEBgcICQsLCw0NDQ4ODw4NDQ0LCwoJCAcGBQM
CAQECBAQHbWkJCwwNDQ0ODw4ODQ0MDAsKCggIBgYEBAKBCwsUFBQTEhEQDgwLCQcFAwEBAwUGCQo
LDQ8PERITExUVFBQTEhXIREA8NDAoJBgUDAQEDBQYICgsMDg8QERETE8sBAGYHCgwOEBMPEBASEXQ
UFh8dGxkXFRMRcGgIBgQEAgIEBQcJCgwQExQXGBkChHoyFyUUEXIRCwoJCAcGBQQDAgE4BQYHBwk
KDAwTFBYGRocHggIBwgGBwYFIyAeHBkYFRMNDakIBgQDAQEBAwQGCakLDxQXGBwdICMWBwcICAc
IBx0YfYUVEEXISFBIQDQsIBwMCAGIAAAFAAAAAAP0A9QAIaA+AIIXagFAAAABDw8jLwY3HwYnMx8
DBY8FPQE/DiUfBw8PLwc3Hwc/DzUvBjcxJx8EBY8GKwEPdUfBgcvBz8SMYUHLwgjDxQVHwsPBB8
HPwQfBzM/EjUvDD8DPQEVBg8CAo0BAGQEAgICQoLDaWNDQ8ODAsLCwoLCQnFBgYFBAMCAY0BERE
QD74EAwMCAgEBAwQFBgcICQoKDAwMDQ0BFBYWFhUVFRQUHR4fHx8gISIXFhCWfXyWFhUVFBUVFRY
VJw4PDxAQERISFRQUEXEREA4NDAoIBwUDAQMDBQcICQsyGswPDh4dHSMNDA4NDg4PDhCTExIREBA
ODQwKCQcGBAIBAwMFBgcIORSaGhkZGRgZGBkYGHkaGxwUEXQUExQTFBITEhIbE74UFBQUFBQUFBw
dHB0WFhYWFhYWFx8fHR4cHBwbBQQAQMDBBoaGhsbHBwc2gQDAQEBAQMEBQUGBgYGBQXoDRwbGxs
bGhsaGhkaGRoaGiAlJSQjIiIhFAQDAgIDBRkYfXgYGRkZFRWvBAMCAGMEBQUGBgYGBQIQDw4ODQw
MCgoJCAcGBQMCAQICAwQFBQfFCQoKCwoMC4EBAwUHVgJCAkKCQkKDg0NDAsLCwkJBwcGBAMCBw8
QERISFBUWHx0bGRcVEhELCAGBAQBAQEBAwQFBwgJJwoKBwcFBAIBAQMFbWgKDA0OEBESEhQUFRI
SERIQEA8OMQ9GAgIFCAokCAYGBAQCAgMFBwgJCw0NDxAREhITFBAQE8PDw4OOBAREXQWFhkaGRg
XFhQTEhALCgkIBwYFBQMCAAdK/CQcHBQUEAwIBAwQFBQcICQoLDBQUFhczGxweCACIBwcGBgYcGxk
YfXQUEToFBgYFBgYGBQQDAQEBAgIE6AcLCwgGBQQAQMFbWgJCxwUfHkbHiAjFQgHCAcIBwgdGBY
WFBQSEQ4MsAQGBgYGBQYFBAMBAQEACgAAAAACAAAAA0A/QACwBtAAABFTMVIxUjNSM1MzUndxE
VHxk/BAE3AT8GNS8YDwoBxZ+faJ+fmQ0NDQsLCgkICAcGBgQEAWMBAQEACgQEBQUHBwkJCQsKCgo
LFhcYGHkbGxscHBscGxoBCpP+8w8OCwkHBQMBBAYICg0OCQgODg4PEBARERISEhMTEhMTHBwcGw4
NDQ0MDQwDbqNkn59noDsLDAwNDQ4ODg8PEBAPERAQERARERAREBEQEBAPEA8ODgwLCgoSEQ4MCQg
GAwEBAwYHC/6udAFUGBgZGhobGxwbGxsaGhkYDAwQDw0NDAsKCAgHBQUdAwEBAQQGCAUFBgcIBwk
AAAAIAAAAA74D9AADAGUAAAEVITU3DxEVHxk/BAE3AT8GNS8YDwoCZ/5XBg0NDQsLCgkICAcGBgQ
EAWMBAQEACgQEBQUHBwkJCQsKCgoLFhcYGHkbGxscHBscGxoBCpP+8w8OCwkHBQMBBAYICg0OCQg
ODg4PEBARERISEhMTEhMTHBwcGw4NDQ0MDQwCy2dn3gsMDA0NDg4ODw8QEA8REBAREREBEQEBA
QEA8QDw4ODAsKChIRDgwJCAYDAQEDBgcL/q50AVQYGBkaGhsbHBsbGxoaGRgMDBAPDQ0MCwoICAc
FBQMDAQEBBAYIBQUGBwgHCQABAAAAANhA/QAAGAAANwKBngLE/TwMAfQB9AAAAgAAAAADxwP0AAM
ABwAAJSERIQEhESECaQFe/qL90AFe/qIMA+j8GAPoAAABAAAAAAP0AxUAGAAAAQ8HJwMhJz8XHx8
PBzM/Ay8eKwEPDQEQDg0MDAsKCQiKIQGHkAUHCAkLDQ4PCwsLDA0MDg0NDg4PDg8PDw8PDg8ODg4
ODQ0NDawMCwsKCgkICAgGBgUFBAMCAGQEBQECawMFBQaOBwYDAQEBawMFBgYICaOKDAwNDg8QEBE
REhITExQUFBUVFRYVfHwUfHwUfHwUfHwUfHwUfHwUfHwUfHwUfHwUfHwUfHwUfHwUfHwUfHwUfHw
GBQUEAwICAQEBAQICAwQFBQYGCAGICQoKCwsMDaWNDQ0ODg4ODw8ODxAQDxAPDg8OHR4fHxUWFRU
VFBQUEXMSEREBEBAQDg0NCwoKCAgHBQUdAwICawMFBQcICaOKCw0NDgAAAAIAAAAA/QDFQACAIM
AADc5AQMPDx8DMY8HPx8fFwchAwcvFisBDw2rIA8ODQwMCgoICAYGBQMCAQEBAwYIjQgFBAMDAQE
BAQEBAwMEBQUGBgICakKCgsLDAwMDQ0NDg4ODg8PDg8PDw8ODw4ODQ4NDA0MCwwLDw0NCwkIBwW
QAaEhiggJCgsMDA0OEBARERITEhQTFBUFRYVfHwUfHwUfHwUfHwUfHwUfHwUfHwUfHwUfHwUfHwUfHw
UFRUVFhUfHx4dFA4ODg8ODw8PDg8PDg4ODg0NDQwMDAsLCgoJCAgIBgYFBQQDAgIBAQEBAgIDBAU
FBgYICAgJCgoQEhETFBQUFaEBRZoSEhEQEQ8QDg8ODQ0LCgoICAcFBQMDAgIDAuUGBggICgoMDA0
OAAUAAAAA98D9AADAAYADQARABQAAEXNycxASETETcRnWjNxczJzkCAqGeJKABHP2HKuAx/nv
7xzRKWwIsliWYAUV+Vf4ZUGGVOQFyMTFWAAAAAEEAAAAA8gD9AFAAABETcRASEBqLABCPxwAh/
```

```

97aIBcQHVAAAAEgDeAAEAAAAAAAAAAQAAAAEAAAAAAAAEABwABAAEAAAAAAAAIABwAIAAEAAAAAAAAAM
ABwAPAAEAAAAAAAAQABwAWAAEAAAAAAAAUACwAdAAEAAAAAAAAAYABwAoAAEAAAAAAAAoALAavAAEAAAA
AAAsAEgBbAAMAAQQAIAAAAgBtAAMAAQQAIAEADgBvAAMAAQQAIAIADgB9AAMAAQQAIAAMADgCLAAM
AAQQAIAAQADgCZAAMAAQQAIAUAFgCnAAMAAQQAIAAYADgC9AAMAAQQAIAoAWADLAAMAAQQAIAAsAJAE
jIHRvb2xiYXJSZWd1bGFydG9vbGJhcnRvb2xiYXJWZXJzaW9uIDEuMHRvb2xiYXJGb250IGd1bmV
yYXR1ZCB1c2luZyBTew5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5jZnVzaW9uLmNvbQAgAHQ
AbwBvAGwAYgBhAHIAUgBlAGcAdQBsAGEAcgB0AG8AbwBsAGIAYQByAHQAbwBvAGwAYgBhAHIAVgB
lAHIAcwBpAG8AbgAgADEALgAwAHQAbwBvAGwAYgBhAHIAArgBvAG4AdAAgAGcAZQBwAGUAcgBhAHQ
AZQBkACAAAdQBzAGkAbgBnACAAUwB5AG4AYwBmAHUAcwBpAG8AbgAgAE0AZQB0AHIAbwAgAFMAdAB
lAGQAaQBvAHcAdwB3AC4AcwB5AG4AYwBmAHUAcwBpAG8AbgAuAGMABwBtAAAAAIAAAAAAAAAACgA
AAAAAAAAAAAAAAAAAAAAAAAAAAACwECAQMBAEFAQYBBwEIAQkBCgELAQwACnNob3ctMDEtd2Y
IaGlkZTEtd2YHem9vbS1pbgh6b29tLW91dAptZWRpYS1wbGF5C21lZG1hLXBhdXNlBHVuZG8EcmV
kbwtmaWx0ZXItbm9uZQZmaWx0ZXIAAA==) format('true');
font-weight: normal;
font-style: normal;
}
.e-toolbar .e-btn .e-icons {
font-family: 'Toolbar' !important;
font-size: 16px;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
}
.e-hide-icon::before {
content: '\a701';
}
.e-show-icon::before {
content: '\a700';
}
.e-filter-icon::before {
content: '\a709';
}
.e-filternone-icon::before {
content: '\a708';
}
.e-undo-icon::before {
content: '\a706';
}
.e-redo-icon::before {
content: '\a707';
}
.e-play-icon::before {
content: '\a704';
}
.e-pause-icon::before {
content: '\a705';
}
.e-zoomin-icon::before {
content: '\a702';
}
.e-zoomout-icon::before {
content: '\a703';
}
</style>
@code{
SfButton MediaButton;

```

```
SfButton ZoomButton;
SfButton UndoButton;
SfButton FilterButton;
SfButton VisibleButton;
public string Content = "Hide";
public string DisplayValue { get; set; } = "block";
public string MediaIconCss = "e-icons e-play-icon";
public string ZoomIconCss = "e-icons e-zoomin-icon";
public string UndoIconCss = "e-icons e-undo-icon";
public string FilterIconCss = "e-icons e-filter-icon";
public string VisibleIconCss = "e-icons e-hide-icon";
public void OnMediaClick(MouseEventArgs args)
{
    if (MediaButton.IconCss == "e-icons e-play-icon")
    {
        MediaIconCss = "e-icons e-pause-icon";
    }
    else
    {
        MediaIconCss = "e-icons e-play-icon";
    }
}
public void OnZoomClick(MouseEventArgs args)
{
    if (ZoomButton.IconCss == "e-icons e-zoomin-icon")
    {
        ZoomIconCss = "e-icons e-zoomout-icon";
    }
    else
    {
        ZoomIconCss = "e-icons e-zoomin-icon";
    }
}
public void OnUndoClick(MouseEventArgs args)
{
    if (UndoButton.IconCss == "e-icons e-undo-icon")
    {
        UndoIconCss = "e-icons e-redo-icon";
    }
    else
    {
        UndoIconCss = "e-icons e-undo-icon";
    }
}
public void OnFilterClick(MouseEventArgs args)
{
    if (FilterButton.IconCss == "e-icons e-filter-icon")
    {
        FilterIconCss = "e-icons e-filternone-icon";
    }
    else
    {
        FilterIconCss = "e-icons e-filter-icon";
    }
}
public void OnVisibleClick(MouseEventArgs args)
{

```



```

if (VisibleButton.Content == "Hide")
{
    DisplayValue = "none";
    Content = "Show";
    VisibleIconCss = "e-icons e-show-icon";
}
else
{
    DisplayValue = "block";
    Content = "Hide";
    VisibleIconCss = "e-icons e-hide-icon";
}
}
}

```



This content will be hidden, when you click on hide button and toggle get an active state as show, otherwise it will be visible.

Customize the Scrolling distance in Blazor Toolbar Component

The toolbar **ScrollStep** property supports to customize the scrolling distance, by clicking the left and right side navigation icons. A required value can be passed through the **ScrollStep** property to customize toolbar scrolling distance.

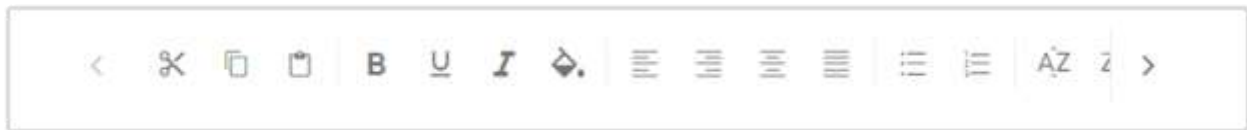
CSHARP

```

@using Syncfusion.Blazor.Navigations
<SfToolbar Width="600" ScrollStep="50">
<ToolbarItems>
<ToolbarItem PrefixIcon="e-icons e-cut" Tooltiptext="Cut"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-copy" Tooltiptext="Copy"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-paste" Tooltiptext="Paste"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-bold" Tooltiptext="Bold"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-underline"
Tooltiptext="Underline"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-italic"
Tooltiptext="Italic"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-paint-bucket" Tooltiptext="Color-
Picker"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-align-left" Tooltiptext="Align-
Left"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-align-right" Tooltiptext="Align-
Right"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-align-center" Tooltiptext="Align-
Center"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-justify" Tooltiptext="Align-
Justify"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-list-unordered"
Tooltiptext="Bullets"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-list-ordered"
Tooltiptext="Numbering"></ToolbarItem>

```

```
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-sort-ascending" Tooltiptext="Sort A -
Z"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-sort-descending" Tooltiptext="Sort Z -
A"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-upload-1"
Tooltiptext="Upload"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-download"
Tooltiptext="Download"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-increase-indent" Tooltiptext="Text
Indent"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-decrease-indent" Tooltiptext="Text
Outdent"></ToolbarItem>
<ToolbarItem Type="ItemType.Separator"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-erase" Tooltiptext="Clear"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-refresh"
Tooltiptext="Reload"></ToolbarItem>
<ToolbarItem PrefixIcon="e-icons e-export"
Tooltiptext="Export"></ToolbarItem>
</ToolbarItems>
</SfToolbar>
```



Tooltip

Getting Started with Blazor Tooltip Component

This section briefly explains how to include a **Tooltip** in the Blazor Server-Side application. Refer to the [Getting Started with Syncfusion Blazor for Server-Side in Visual Studio page](#) for the introduction and configuring the common specifications.

To get start quickly with the Blazor Tooltip, check on this video.

{% youtube

"youtube:https://www.youtube.com/watch?v=Lzpm6x6TXO8"%}

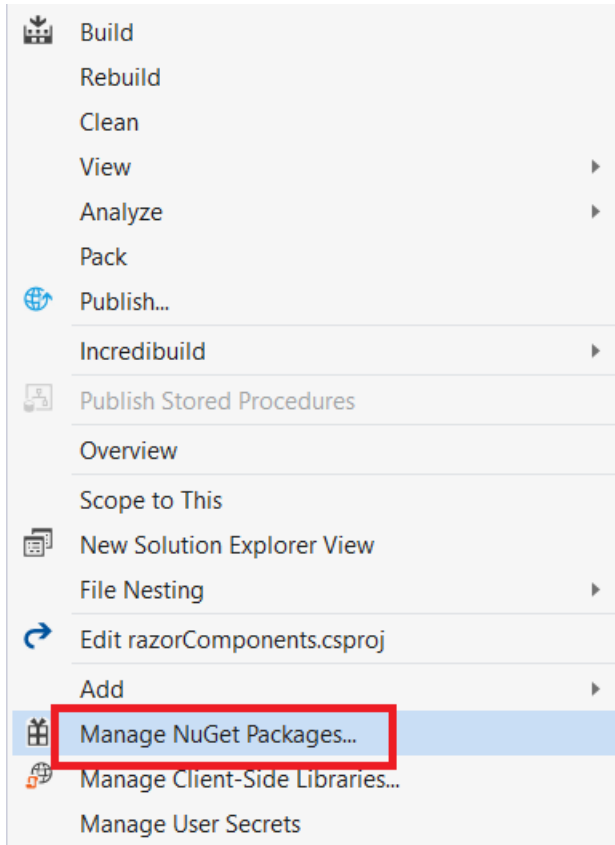
Importing Syncfusion Blazor component in the application

Any one of the below standards can be used to install the Syncfusion Blazor library in the application.

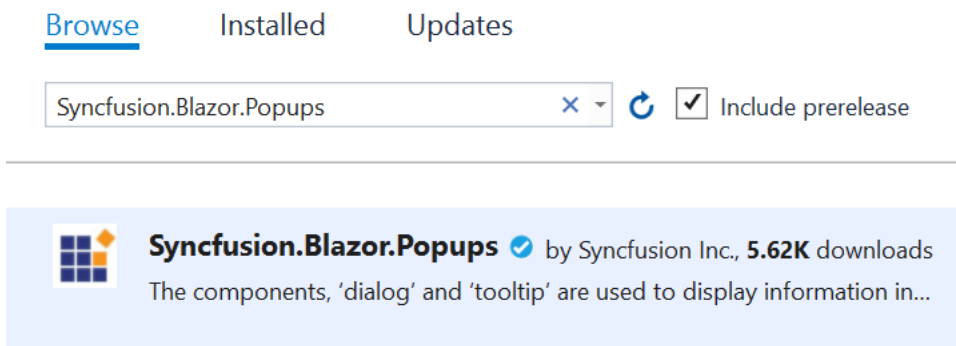
Using Syncfusion Blazor Individual NuGet Packages [New standard]

Starting with Volume 4, 2020 (v18.4.0.30) release, Syncfusion provides [individual NuGet packages](#) for our Syncfusion Blazor components. We highly recommend this new standard for your Blazor production applications. Refer to [this section](#) to know the benefits of the individual NuGet packages.

1. Install **Syncfusion.Blazor.Popups** NuGet package to the application by using the **NuGet Package Manager**. Refer to the [Individual NuGet Packages](#) section for the available NuGet packages.



2. Search **Syncfusion.Blazor.Popups** keyword in the Browse tab and install **Syncfusion.Blazor.Popups** NuGet package in the application.



3. Once the installation process is completed, the Syncfusion Blazor Popups package will be installed in the project.
4. The client-side style resources can be added through [CDN](#) or from [NuGet](#) package in the element of the ~/Pages/_Host.cshtml page.

HTML



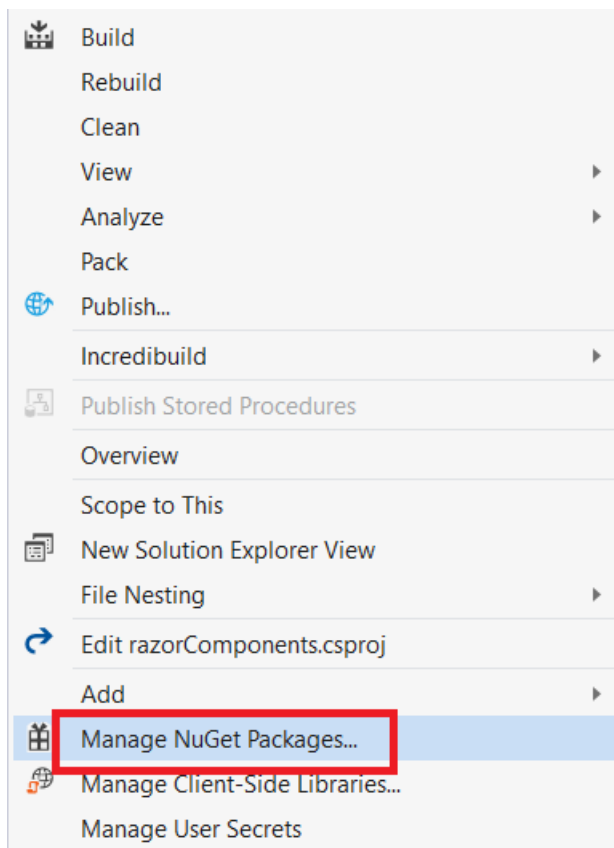
```
<link href="_content/Syncfusion.Blazor.Themes/bootstrap4.css"
rel="stylesheet" />
</head>
```

Warning: Syncfusion.Blazor package should not be installed along with the [individual NuGet packages](#). Hence, add the above Syncfusion.Blazor.Themes static web assets (styles) in the application.

Using Syncfusion.Blazor NuGet Package [Old standard]

Warning: If you prefer the above new standard (individual NuGet packages), then skip this section. Using both old and new standards in the same application will throw ambiguous compilation errors.

1. Install **Syncfusion.Blazor** NuGet package to the application by using the **NuGet Package Manager**. Right-click the project and then select **Manage NuGet Packages**.



2. Search **Syncfusion.Blazor** keyword in the Browse tab and install **Syncfusion.Blazor** NuGet package in the application.

[Browse](#)

Installed

Updates

Syncfusion.Blazor



Include prerelease

**Syncfusion.Blazor** by Syncfusion Inc., **192K** downloads

Syncfusion Blazor Components are built from the ground up to be lightw...

- Once the installation process is completed, the Syncfusion Blazor package will be installed in the project.
- The client-side style resources can be added using NuGet package to the `element` of the `~/wwwroot/index.html` page in Blazor WebAssembly app or `~/Pages/_Host.cshtml` page in Blazor Server app.

HTML

```
<head>
...
...
<link href="_content/Syncfusion.Blazor/styles/bootstrap4.css"
rel="stylesheet" />
</head>
```

HTML

```
<head>
<link href="https://cdn.syncfusion.com/blazor/{{ site.blazorversion
}}/styles/bootstrap4.css" rel="stylesheet" />
</head>
```

For Internet Explorer 11 kindly refer the polyfills. Refer the [documentation](#) for more information.

ASPX-CS

```
<head>
<link href="_content/Syncfusion.Blazor/styles/bootstrap4.css"
rel="stylesheet" />
<script
src="https://github.com/Daddoon/Blazor.Polyfill/releases/download/3.0.1/blaz
or.polyfill.min.js"></script>
</head>
```

Add Syncfusion Blazor service in Startup.cs (Server-side application)

Open the **Startup.cs** file and add services required by Syncfusion components using `services.AddSyncfusionBlazor()` method. Add this method in the `ConfigureServices` function as follows.

CSHARP

```
using Syncfusion.Blazor;
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void ConfigureServices(IServiceCollection services)
        {
            ....
            ....
            services.AddSyncfusionBlazor();
        }
    }
}
```

Add Syncfusion Blazor service in Program.cs (Client-side application)

Open the **Program.cs** file and add services required by Syncfusion components using `builder.services.AddSyncfusionBlazor()` method. Add this method in the **Main** function as follows.

C#

```
using Syncfusion.Blazor;
namespace BlazorApplication
{
    public class Program
    {
        public static async Task Main(string[] args)
        {
            ....
            ....
            builder.Services.AddSyncfusionBlazor();
        }
    }
}
```

Adding component package to the application

Open `~//_Imports.razor` file and import the `Syncfusion.Blazor.Popups` package.

ASPX-CS

```
@using Syncfusion.Blazor
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.Popups
```

Adding Tooltip component to the application

Now, add the Syncfusion Blazor Tooltip component in any web page razor in the **Pages** folder. For example, the Tooltip component is added in the `~/Pages/Index.razor` page.

ASPX-CS

```
@using Syncfusion.Blazor.Popups
@using Syncfusion.Blazor.Buttons
<SfTooltip ID="Tooltip" Target="#btn" Content="@Content">
```

```
<SfButton ID="btn" Content="Show Tooltip"></SfButton>
</SfTooltip>
@code
{
    string Content = "Lets go green & Save Earth !!";
}
```

Run the application

After successful compilation of your application, simply press **F5** to run the application.



See Also

- [Positioning Tooltip](#)
- [Tooltip Open Mode](#)
- [Customize the Tooltip](#)
- [Getting Started with Syncfusion Blazor for Client-Side in .NET Core CLI](#)
- [Getting Started with Syncfusion Blazor for Client-Side in Visual Studio 2019](#)
- [Getting Started with Syncfusion Blazor for Server-Side in .NET Core CLI](#)

Setting Dimension in Blazor Tooltip Component

Height and width

The Tooltip can either be assigned with auto height and width values or specific pixel values. The **Width** and **Height** properties are used to set the outer dimension of the Tooltip element. The default value for both the properties is **auto**. It also accepts string and number values in pixels.

ASPX-CS

```
@using Syncfusion.Blazor.Popups
@using Syncfusion.Blazor.Buttons
```

```

<SfTooltip ID="Tooltip" Height="50px" Width="200px" Target="#btn"
Content="@Content">
<SfButton ID="btn" Content="Show Tooltip"></SfButton>
</SfTooltip>
@code
{
string Content="Lets go green & Save Earth !!";
}

```



Scroll mode

When **Height** is specified with a certain pixel value and the Tooltip content overflows, the scrolling mode gets enabled.

ASPX-CS

```

@using Syncfusion.Blazor.Popups
<SfTooltip ID="tooltip" Height="60px" Width="200px" IsSticky="true"
Target="#target" Content="@Content">
<div id='container'>
<p>
A green home is a type of house designed to be
<a id="target">
<u>environmentally friendly</u>
</a> and sustainable. And also focuses on the efficient use of "energy,
water, and building materials." As green homes
have become more prevalent we have also seen the emergence of green
affordable housing.
</p>
</div>
</SfTooltip>
@code
{
string Content = "<div><b>Environmentally friendly</b> or environment-
friendly, (also referred to as eco-friendly, nature-friendly, and green) are
marketing and sustainability terms referring to goods and services, laws,

```



```
guidelines and policies that inflict reduced, minimal, or no harm upon
ecosystems or the environment.</div>";
}
```

A green home is a type of house designed to be environmentally friendly and sustainable. And also focuses on the efficient use of "energy, water, and building materials." As green homes have become more prevalent we have also seen the emergence of green affordable housing.

The scrolling mode can best be seen when the sticky mode of the Tooltip is enabled. To enable sticky mode, set the `IsSticky` property to true.

Content in Blazor Tooltip Component

A text or a piece of information assigned to the Tooltip's `Content` property will be displayed as the main text stream of the Tooltip. It can be a string or a template content. If the `Content` property is not provided with any specific value, then it takes the value assigned to the attribute of the target element on which the Tooltip was initialized. The content can also dynamically be assigned to the Tooltip via AJAX.

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.

Refer to the following code example to add formatted HTML content to the Tooltip.

ASPX-CS

```
@using Syncfusion.Blazor.Popups
<SfTooltip ID="tooltip" IsSticky="true" Target="#target" Content="@Content">
<div id='container'>
<p>
A green home is a type of house designed to be
<a id="target">
<u>environmentally friendly</u>
</a> and sustainable. And also focuses on the efficient use of "energy,
water, and building materials." As green homes
have become more prevalent we have also seen the emergence of green
affordable housing.
</p>
</div>
</SfTooltip>
@code
{
string Content = "<div><b>Environmentally friendly</b> or environment-
friendly, (also referred to as eco-friendly, nature-friendly, and green) are
marketing and sustainability terms referring to goods and services, laws,
guidelines and policies that inflict reduced, minimal, or no harm upon
ecosystems or the environment.</div>";
}
```

A green home is a type of house designed to be environmentally friendly and sustainable. And also focuses on the efficient use of "energy, water, and building materials." As green homes have become more prevalent we have also seen the emergence of green affordable housing.



Position in Blazor Tooltip Component

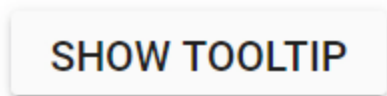
Tooltips can be attached to 12 static locations around the target. On initializing the Tooltip, set the position property with any one of the following values:

- TopLeft
- TopCenter
- TopRight
- BottomLeft
- BottomCenter
- BottomRight
- LeftTop
- LeftCenter
- LeftBottom
- RightTop
- RightCenter
- RightBottom

By default, Tooltip is placed at the TopCenter of the target element.

ASPX-CS

```
@using Syncfusion.Blazor.Popups
@using Syncfusion.Blazor.Buttons
<SfTooltip Target="#target" Content="@Content"
Position="Position.RightCenter">
<SfButton ID="target" Content="Show Tooltip"></SfButton>
</SfTooltip>
@code
{
    string Content = "Tooltip Content";
}
```



Mouse trailing

Tooltips can be positioned relative to the mouse pointer. This behavior can be enabled or disabled by using the `MouseTrail` property. By default, it is set to `false`.

ASPX-CS

```
@using Syncfusion.Blazor.Popups
@using Syncfusion.Blazor.Buttons
<SfTooltip MouseTrail=true Content="@Content" Target="#target">
<SfButton ID="target" Content="Show Tooltip"></SfButton>
</SfTooltip>
@code
{
string Content="Tooltip Content";
}
<style>
#target {
background-color: #ecec;
border: 1px solid #c8c8c8;
box-sizing: border-box;
margin: 80px auto;
padding: 20px;
width: 200px;
}
</style>
```



SHOW TOOLTIP

When mouse trailing option is enabled, the tip pointer position gets auto adjusted based on the target, and other position values like start, end, and middle are not applied (to prevent the pointer from moving out of target).

Setting offset values

Offset values are set to specify the distance between the target and tooltip element. `OffsetX` and `OffsetY` properties are used to specify the offset left and top values.

- `OffsetX` specifies the distance between the target and Tooltip element in X axis.
- `OffsetY` specifies the distance between the target and Tooltip element in Y axis.

ASPX-CS

```
@using Syncfusion.Blazor.Popups
@using Syncfusion.Blazor.Buttons
<SfTooltip Target="#target" ShowTipPointer="false" MouseTrail="true"
Content="@Content" OffsetX="30" OffsetY="30">
<SfButton ID="target" Content="Show Tooltip"></SfButton>
</SfTooltip>
@code
{
    string Content="Tooltip Content";
}
<style>
#target {
background-color: #ecec;
border: 1px solid #c8c8c8;
box-sizing: border-box;
margin: 80px auto;
padding: 20px;
width: 200px;
}
</style>
```



By default, collision is handled automatically and therefore when collision is detected the Tooltip fits horizontally and flips vertically.

Open Mode in Blazor Tooltip Component

The mode on which the Tooltip is to be opened on a page, i.e., on hovering, focusing, or clicking on the target elements can be decided.

On mobile devices, Tooltips appear when you tap and hold the element, even if the **OpensOn** option is assigned with **Hover**.

Tooltips are also displayed as long as the element is continued to tap and hold. On lift, it disappears in the next 1.5 seconds.

If there is another action before that time ends, then the Tooltip disappears.

The **OpensOn** property can take either a single or a combination of multiple values, separated by space from the following options. The table below explains how the Tooltip opens on both desktop and mobile based on the value(s) assigned to the **OpensOn** property. By default, it takes **Auto** value.

Values	Desktop	Mobile
Auto	Tooltip appears when you hover over the target or when the target element receives the focus.	Tooltip opens on tap and hold of the target element.
Hover	Tooltip appears when you hover over the target.	Tooltip opens on tap and hold of the target element.
Click	Tooltip appears when you click a target element.	Tooltip appears when you single tap the target element.
Focus	Tooltip appears when you focus (say through tab key) on a target element.	Tooltip appears with a single tap on the target element.
Custom	Tooltip is not triggered by any default action. So, you have to bind your own events and use either open or close public methods. Same as Desktop.	

To open the Tooltip for multiple actions, say while hovering over or clicking on a target element, the **OpensOn** property can be assigned with multiple values, separated by space as **Hover Click**.

Auto value cannot be used with any combination for multiple values.

ASPX-CS

```
@using Syncfusion.Blazor.Popups
<SfTooltip ID="tooltiphover" Target=".blocks" Content="@Content"
OpensOn="Click">
<div class="blocks"><span>Click Me !</span></div>
</SfTooltip>
<SfTooltip ID="tooltipclick" Target=".blocks" Content="@Content"
OpensOn="Hover">
<div class="blocks"><span>Hover Me !(Default)</span></div>
</SfTooltip>
<SfTooltip ID="tooltipfocus" Target=".e-info" Content="@Content"
OpensOn="Focus">
<div class="blocks"><span><input class="e-info" type="text"
placeholder="Focus and blur" /></span></div>
</SfTooltip>
@code
{
string Content="Tooltip content";
}
<style>
#tooltiphover {
width: 200px;
box-sizing: border-box;
display: inline-block;
}
#tooltipclick {
width: 200px;
box-sizing: border-box;
display: inline-block;
}
#tooltipfocus {
width: 200px;
box-sizing: border-box;
display: inline-block;
line-height: 17px;
}
#tooltipfocus .blocks span {
line-height: 17px;
}
#tooltipcustom .blocks #tooltipopen {
line-height: 17px;
}
.blocks {
background-color: #ecec;
border: 1px solid #c8c8c8;
box-sizing: border-box;
display: inline-block;
line-height: 50px;
margin: 0 10px 10px 0;
overflow: hidden;
text-align: center;
vertical-align: middle;
width: 200px;
}
```

```
</style>
```

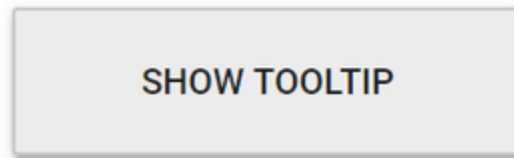


Sticky mode

With this mode set to `true`, Tooltips can be made to show up on the screen as long as the close icon is pressed. In this mode, close icon is attached to the Tooltip located at the top right corner. This mode can be enabled or disabled using the `IsSticky` property.

ASPX-CS

```
@using Syncfusion.Blazor.Popups
@using Syncfusion.Blazor.Buttons
<SfTooltip Target="#target" Content="@Content" IsSticky="true">
<SfButton ID="target" Content="Show Tooltip"></SfButton>
</SfTooltip>
@code
{
string Content="Click close icon to close me";
}
<style>
#target {
background-color: #ececec;
border: 1px solid #c8c8c8;
box-sizing: border-box;
margin: 80px auto;
padding: 20px;
width: 200px;
}
</style>
```



Open or Close tooltip with delay

The Tooltips can be opened or closed after some delay by using the `OpenDelay` and `CloseDelay` properties.

ASPX-CS

```
@using Syncfusion.Blazor.Popups
@using Syncfusion.Blazor.Buttons
<SfTooltip Target="#target" Content="@Content" OpenDelay="1000"
CloseDelay="1000">
<SfButton ID="target" Content="Show Tooltip"></SfButton>
</SfTooltip>
@code
{
    string Content="Tooltip with delay";
}
<style>
#target {
background-color: #ecec;
border: 1px solid #c8c8c8;
box-sizing: border-box;
margin: 80px auto;
padding: 20px;
width: 200px;
}
</style>
```




Customization in Blazor Tooltip Component

The Tooltip can be customized by using the `CssClass` property, which accepts custom CSS class names that defines the specific user-defined styles and themes to be applied on the Tooltip element.

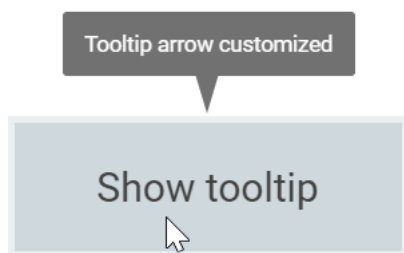
Tip pointer customization

Styling the tip pointer's size, background, and border colors can be done using the `CssClass` property, as given below.

ASPX-CS

```
@using Syncfusion.Blazor.Popups
@using Syncfusion.Blazor.Buttons
<SfTooltip Target="#target" CssClass="customtip" Content="@Content">
<SfButton ID="target" Content="Show Tooltip"></SfButton>
</SfTooltip>
@code
{
string Content = "Tooltip arrow customized";
}
<style>
#target {
background-color: #ececec;
border: 1px solid #c8c8c8;
box-sizing: border-box;
margin: 70px auto;
padding: 20px;
width: 200px;
}
/* csslint ignore:start */
.customtip.e-tooltip-wrap {
padding: 4px;
}
.customtip.e-tooltip-wrap .e-arrow-tip.e-tip-bottom {
height: 20px;
width: 12px;
}
.customtip.e-tooltip-wrap .e-arrow-tip.e-tip-top {
height: 20px;
```

```
width: 12px;
}
.customtip.e-tooltip-wrap .e-arrow-tip.e-tip-left {
height: 12px;
width: 20px;
}
.customtip.e-tooltip-wrap .e-arrow-tip.e-tip-right {
height: 12px;
width: 20px;
}
.customtip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-bottom {
border-left: 6px solid transparent;
border-right: 6px solid transparent;
border-top: 20px solid #616161;
}
.customtip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-top {
border-bottom: 20px solid #616161;
border-left: 6px solid transparent;
border-right: 6px solid transparent;
}
.customtip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-left {
border-bottom: 6px solid transparent;
border-right: 20px solid #616161;
border-top: 6px solid transparent;
}
.customtip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-right {
border-bottom: 6px solid transparent;
border-left: 20px solid #616161;
border-top: 6px solid transparent;
}
</style>
```



Tooltip customization

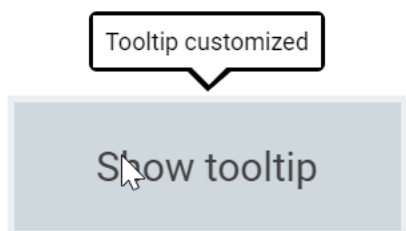
The complete look and feel of the Tooltip can be customized by changing its background color, opacity, content font, etc.

ASPX-CS

```
@using Syncfusion.Blazor.Popups
@using Syncfusion.Blazor.Buttons
<SfTooltip Target="#target" CssClass="customtooltip" Content="@Content">
<SfButton ID="target" Content="Show Tooltip"></SfButton>
</SfTooltip>
@code
{
```

```
string Content="Tooltip customized";
}
<style>
#target {
background-color: #ececce;
border: 1px solid #c8c8c8;
box-sizing: border-box;
margin: 70px auto;
padding: 20px;
width: 200px;
}
.customtooltip.e-tooltip-wrap .e-tip-content {
line-height: 20px;
}
.customtooltip.e-tooltip-wrap .e-arrow-tip.e-tip-bottom {
height: 12px;
left: 50%;
top: 100%;
width: 24px;
}
.customtooltip.e-tooltip-wrap .e-arrow-tip.e-tip-top {
height: 12px;
left: 50%;
top: -9px;
width: 24px;
}
.customtooltip.e-tooltip-wrap .e-arrow-tip.e-tip-left {
height: 24px;
left: -9px;
top: 48%;
width: 12px;
}
.customtooltip.e-tooltip-wrap .e-arrow-tip.e-tip-right {
height: 24px;
left: 100%;
top: 50%;
width: 12px;
}
.customtooltip.e-tooltip-wrap {
border-radius: 4px;
opacity: 1;
}
.customtooltip.e-tooltip-wrap.e-popup {
background-color: #fff;
border: 2px solid #000;
}
.customtooltip.e-tooltip-wrap .e-tip-content {
color: #000;
font-size: 12px;
}
.customtooltip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-bottom {
border-left: 12px solid transparent;
border-right: 14px solid transparent;
border-top: 12px solid #000;
}
.customtooltip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-top {
border-bottom: 12px solid #000;
```

```
border-left: 12px solid transparent;
border-right: 12px solid transparent;
}
.customtooltip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-left {
border-bottom: 12px solid transparent;
border-right: 12px solid #000;
border-top: 12px solid transparent;
}
.customtooltip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-right {
border-bottom: 12px solid transparent;
border-left: 12px solid #000;
border-top: 12px solid transparent;
}
.customtooltip.e-tooltip-wrap .e-arrow-tip-inner.e-tip-bottom {
color: #fff;
font-size: 25.9px;
}
</style>
```



Styles and Appearance in Blazor Tooltip 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 tooltip

Use the following CSS to customize the tooltip.

CSS

```
.e-tooltip-wrap {
border-radius: 4px;
opacity: 1;
}
```

Customizing the tooltip popup

Use the following CSS to customize the tooltip popup properties.

CSS

```
.e-tooltip-wrap.e-popup {
background-color: #fff;
border: 2px solid #000;
}
```

Customizing the tooltip content

Use the following CSS to customize the tooltip content.

CSS

```
.e-tooltip-wrap .e-tip-content {  
  color: red;  
  font-size: 12px;  
  line-height: 20px;  
}
```

Customizing the tooltip arrow tip

Use the following CSS to customize the tooltip arrow tip.

CSS

```
/* To customize the arrow tip at bottom */  
.e-tooltip-wrap .e-arrow-tip.e-tip-bottom {  
  height: 12px;  
  left: 50%;  
  top: 100%;  
  width: 24px;  
}  
/* To customize the arrow tip at top */  
.e-tooltip-wrap .e-arrow-tip.e-tip-top {  
  height: 12px;  
  left: 50%;  
  top: -9px;  
  width: 24px;  
}  
/* To customize the arrow tip at left */  
.e-tooltip-wrap .e-arrow-tip.e-tip-left {  
  height: 24px;  
  left: -9px;  
  top: 48%;  
  width: 12px;  
}  
/* To customize the arrow tip at right */  
.e-tooltip-wrap .e-arrow-tip.e-tip-right {  
  height: 24px;  
  left: 100%;  
  top: 50%;  
  width: 12px;  
}
```

Customizing the tooltip inner tip

Use the following CSS to customize the tooltip inner tip.

CSS

```
.e-tooltip-wrap .e-arrow-tip-inner.e-tip-bottom {  
  color: #fff;  
  font-size: 25.9px;  
}
```

Customizing the tooltip outer tip

Use the following CSS to customize the tooltip outer tip.

CSS

```
/* To customize the arrow tip at bottom */
.e-tooltip-wrap .e-arrow-tip-outer.e-tip-bottom {
border-left: 12px solid transparent;
border-right: 14px solid transparent;
border-top: 12px solid #000;
}
/* To customize the arrow tip at top */
.e-tooltip-wrap .e-arrow-tip-outer.e-tip-top {
border-bottom: 12px solid #000;
border-left: 12px solid transparent;
border-right: 12px solid transparent;
}
/* To customize the arrow tip at left */
.e-tooltip-wrap .e-arrow-tip-outer.e-tip-left {
border-bottom: 12px solid transparent;
border-right: 12px solid #000;
border-top: 12px solid transparent;
}
/* To customize the arrow tip at right */
.e-tooltip-wrap .e-arrow-tip-outer.e-tip-right {
border-bottom: 12px solid transparent;
border-left: 12px solid #000;
border-top: 12px solid transparent;
}
```

Template in Blazor Tooltip Component

Tooltip Template property can be used to customize the tooltip layout or to add any custom elements within the tooltip. Any content or HTML elements can be added as tooltip content by specifying them in the `<content>` tag of the tooltip templates.

Refer to the following code example to add and display a HTML page to the Tooltip.

ASPX-CS

```
@using Syncfusion.Blazor.Buttons;
@using Syncfusion.Blazor.Popups;
<SfTooltip CssClass="e-tooltip-css" OpensOn="Click" Target="#btn">
<TooltipTemplates>
<Content>
<div id='democontent' class='democontent'>
<div class='info'>
<h3 style='margin-top:10px'>Eastern Bluebird <hr style='margin-
top:10px'></h3>
<div style='margin-top: -10px'>
<div style='float:left;width:57%'>
The<a href='https://en.wikipedia.org/wiki/Eastern_bluebird' target='blank'>
Eastern Bluebird</a>
is easily found in open fields and sparse woodland areas, including along
woodland edges. These are<i>cavity-nesting birds</i> and a pair of eastern
bluebirds will raise 2-3 broods annually, with 2-8 light blue or whitish
eggs per brood.
```

```

</div>
<div id='bird' style='float:right;width:42%'><img
src='https://blazor.syncfusion.com/demos/css/tooltip/bird.png' width='125'
height='125' /></div>
</div>
<div style='margin-top:160px'>
<hr><p style='margin-top:-11px'> Eastern bluebirds can be very vocal in
flocks.Their calls include a rapid, mid-tone chatter and several long
dropping pitch calls.</p>
</div><p>Source:<br /><a
href='https://en.wikipedia.org/wiki/Eastern_bluebird'
target='_blank'>https://en.wikipedia.org/wiki/Eastern_bluebird</a></p>
</div>
</div>
</Content>
</TooltipTemplates>
<SfButton ID="btn" CssClass="e-outline text" IsPrimary="true" Content="HTML
Template"></SfButton>
</SfTooltip>
<style>
.e-tooltip-css {
filter: drop-shadow(2px 5px 5px rgba(0, 0, 0, 0.25));
}
.text {
text-transform: capitalize;
width: 155px;
}
.democontent {
border: 0.5px solid grey;
}
#bird {
padding-top: 4px;
}
.info a {
color: #2FA1E3;
}
.info {
padding-left: 12px;
padding-right: 5px;
}
</style>

```

HTML Template



TreeGrid

<!-- markdownlint-disable MD024 -->

Getting Started with Blazor TreeGrid Component

This section briefly explains about how to include a **Tree Grid** Component in the Blazor server-side application. It also includes initializing Tree Grid Blazor component and usage of features such as paging and sorting using the [Visual Studio 2019](#). Refer [Getting Started with Syncfusion Blazor for Server-Side in Visual Studio 2019](#) page for the introduction and configuring the common specifications.

Importing Syncfusion Blazor component in the application

<!-- markdownlint-disable MD033 -->

1. Install **Syncfusion.Blazor.TreeGrid** NuGet package to the application by using the **NuGet Package Manager**. Please ensure to check the **Include prerelease** option.
2. The client-side resources can be added through CDN or from NuGet package in the **HEAD** element of the `~/Pages/_Host.cshtml` page.

HTML

```
<head>  
....  
....
```



```
<link href="_content/Syncfusion.Blazor.Themes/bootstrap4.css"
rel="stylesheet" />
</head>
```

For Internet Explorer 11 kindly refer the polyfills. Refer the [documentation](#) for more information.

HTML

```
<head>
<link href="_content/Syncfusion.Blazor.Themes/bootstrap4.css"
rel="stylesheet" />
<script
src="https://github.com/Daddoon/Blazor.Polyfill/releases/download/3.0.1/blaz
or.polyfill.min.js"></script>
</head>
```

Add SyncfusionBlazor service in Startup.cs

Open the **Startup.cs** file and add services required by Syncfusion components using **services.AddSyncfusionBlazor()** method. Add this method in the **ConfigureServices** function as follows.

C#

```
using Syncfusion.Blazor;
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void ConfigureServices(IServiceCollection services)
        {
            ....
            ....
            services.AddSyncfusionBlazor();
        }
    }
}
```

Adding component package to the application

Open **~/_Imports.razor** file and import the **Syncfusion.Blazor.TreeGrid** packages.

ASPX-CS

```
@using Syncfusion.Blazor.TreeGrid
```

Add Tree Grid component to the application

Now, add the Syncfusion Blazor Tree Grid component in any web page (razor) in the **Pages** folder. For example, the Tree Grid component is added in the **~/Pages/Index.razor** page.

ASPX-CS

```
<SfTreeGrid>
</SfTreeGrid>
```

Defining columns

The Tree Grid has an option to define columns using the [TreeGridColumn](#) directive. In the **TreeGridColumn** directive there are properties to customize columns.

Let's check the properties used here:

- [Field](#) to map with a property name in datasource is been added.
- [HeaderText](#) to change the title of columns is been added.
- [TextAlign](#) to change the alignment of columns is been used. By default, columns will be left aligned. To change columns to right align, define **TextAlign** as *Right*.

ASPX-CS

```
<SfTreeGrid DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
}
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
Duration = 10, Progress = 70, ParentId = null, Priority = "High" });
TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
Duration = 4, Progress = 80, ParentId = 1, Priority = "Normal" });
TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
Duration = 5, Progress = 65, ParentId = 1, Priority = "Critical" });
TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
Duration = 6, Progress = 77, ParentId = null, Priority = "Low" });
TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
Duration = 9, Progress = 25, ParentId = 4, Priority = "Normal" });
TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
Duration = 9, Progress = 7, ParentId = 5, Priority = "Normal" });
TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
Duration = 4, Progress = 45, ParentId = null, Priority = "High" });
}
```

```
TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
Duration = 3, Progress = 38, ParentId = 7, Priority = "Critical" });
TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
Duration = 7, Progress = 70, ParentId = 7, Priority = "Low" });
}
}
```

In the above code example, the [Self-Referential](#) data binding is represented in which the [IdMapping](#) and [ParentIdMapping](#) properties denotes the hierarchy relationship; whereas in [Hierarchical](#) data binding [ChildMapping](#) denotes the hierarchy relationship.

Enable paging

The paging feature enables users to view the tree grid record in a paged view. It can be enabled by setting the [AllowPaging](#) property to true. The pager can be customized using the [PageSettings](#) property.

In root-level paging mode, paging is based on the root-level rows only, i.e., it ignores the child row count and it can be enabled by using the [PageSettings.PageSizeMode](#) property.

ASPX-CS

```
<SfTreeGrid DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" AllowPaging="true">
<TreeGridPageSettings PageSizeMode="PageSizeMode.Root"
PageSize="2"></TreeGridPageSettings>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code {
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
}
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
Duration = 10, Progress = 70, ParentId = null, Priority = "High" });
TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
Duration = 4, Progress = 80, ParentId = 1, Priority = "Normal" });
TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
Duration = 5, Progress = 65, ParentId = 1, Priority = "Critical" });
}
```

```

TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
Duration = 6, Progress = 77, ParentId = null, Priority = "Low" });
TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
Duration = 9, Progress = 25, ParentId = 4, Priority = "Normal" });
TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
Duration = 9, Progress = 7, ParentId = 5, Priority = "Normal" });
TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
Duration = 4, Progress = 45, ParentId = null, Priority = "High" });
TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
Duration = 3, Progress = 38, ParentId = 7, Priority = "Critical" });
TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
Duration = 7, Progress = 70, ParentId = 7, Priority = "Low" });
}
}

```

Enable sorting

The sorting feature enables to order the records. It can be enabled by setting the [AllowSorting](#) property to **true**.

ASPX-CS

```

<SfTreeGrid DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1"
AllowPaging="true" AllowSorting="true">
  <TreeGridPageSettings PageSizeMode="PageSizeMode.Root"
  PageSize="2"></TreeGridPageSettings>
  <TreeGridColumns>
    <TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
    TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
    <TreeGridColumn Field="TaskName" HeaderText="Task Name"
    Width="160"></TreeGridColumn>
    <TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
    TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
    <TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
    TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
    <TreeGridColumn Field="Priority" HeaderText="Priority"
    Width="80"></TreeGridColumn>
  </TreeGridColumns>
</SfTreeGrid>
@code {
  public class BusinessObject
  {
    public int TaskId { get; set; }
    public string TaskName { get; set; }
    public int Duration { get; set; }
    public int Progress { get; set; }
    public string Priority { get; set; }
    public int? ParentId { get; set; }
  }
  public List<BusinessObject> TreeData = new List<BusinessObject>();
  protected override void OnInitialized()
  {
    TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
    Duration = 10, Progress = 70, ParentId = null, Priority = "High" });
    TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
    Duration = 4, Progress = 80, ParentId = 1, Priority = "Normal" });
  }
}

```

```

TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
Duration = 5, Progress = 65, ParentId = 1, Priority = "Critical" });
TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
Duration = 6, Progress = 77, ParentId = null, Priority = "Low" });
TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
Duration = 9, Progress = 25, ParentId = 4, Priority = "Normal" });
TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
Duration = 9, Progress = 7, ParentId = 5, Priority = "Normal" });
TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
Duration = 4, Progress = 45, ParentId = null, Priority = "High" });
TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
Duration = 3, Progress = 38, ParentId = 7, Priority = "Critical" });
TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
Duration = 7, Progress = 70, ParentId = 7, Priority = "Low" });
}
}

```

Run the application

After successful compilation of the application, simply press F5 to run the application. The Blazor Tree Grid component will render in the web browser as shown below.

Task ID	Task Name	Duration	Progress	Priority
1	▼ Parent Task 1	10	70	High
2	Child task 1	4	80	Normal
3	Child Task 2	5	65	Critical
4	▼ Parent Task 2	6	77	Low
5	▼ Child Task 5	9	25	Normal
6	Child Task 6	9	7	Normal
<< < 1 2 > >>				1 of 2 pages (3 items)

See also

- [Getting Started with Syncfusion Blazor for Server-Side in .NET Core CLI](#)
- [Getting Started with Syncfusion Blazor for Client-Side in Visual Studio 2019](#)
- [Getting Started with Syncfusion Blazor for Client-Side in .NET Core CLI](#)

Data Binding in Blazor TreeGrid Component

The Tree Grid uses **SfDataManager**, which supports both RESTful Web Services binding and List binding. The [DataSource](#) property can be assigned either using the **SfDataManager** as child component of the Tree Grid Blazor component or list of business objects.

It supports two kinds of data binding method:

- List binding
- Remote service binding

List binding

In List binding, data source for rendering the Tree Grid component is retrieved from the same application locally.

Two types of Data binding are possible with the Tree Grid component.

- Self-Referential Data binding (Flat Data)
- Hierarchical Data binding

For Self-Referential data binding, assign the list of business objects to the [DataSource](#) property.

For Hierarchy Data binding, the data-source should be assigned as an object array to the **Json** property of the **SfDataManager** and the **Adaptor** property of the SfDataManager should be either **RemoteSaveAdaptor** or **JsonAdaptor**.

Self-Referential data binding/Flat Data

Tree Grid is rendered from Self-Referential data structures by providing two fields, [IdMapping](#) field and [ParentIdMapping](#) field.

- [IdMapping](#): This field contains unique values used to identify nodes.
- [ParentIdMapping](#): This field contains values that indicate parent nodes.

ASPX-CS

```
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1">
  <TreeGridColumns>
    <TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
    TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
    <TreeGridColumn Field="TaskName" HeaderText="Task Name"
    Width="160"></TreeGridColumn>
    <TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
    TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
    <TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
    TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
  </TreeGridColumns>
</SfTreeGrid>
@code{
public class BusinessObject
{
    public int TaskId { get; set; }
    public string TaskName { get; set; }
    public int Duration { get; set; }
    public int Progress { get; set; }
    public string Priority { get; set; }
    public int? ParentId { get; set; }
}
```

```

public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
    TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
    Duration = 10, Progress = 70, ParentId = null, Priority = "High" });
    TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
    Duration = 4, Progress = 80, ParentId = 1, Priority = "Normal" });
    TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
    Duration = 5, Progress = 65, ParentId = 1, Priority = "Critical" });
    TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
    Duration = 6, Progress = 77, ParentId = null, Priority = "Low" });
    TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
    Duration = 9, Progress = 25, ParentId = 4, Priority = "Normal" });
    TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
    Duration = 9, Progress = 7, ParentId = 5, Priority = "Normal" });
    TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
    Duration = 4, Progress = 45, ParentId = null, Priority = "High" });
    TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
    Duration = 3, Progress = 38, ParentId = 7, Priority = "Critical" });
    TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
    Duration = 7, Progress = 70, ParentId = 7, Priority = "Low" });
}
}

```

Hierarchy data binding

The [ChildMapping](#) property is used to map the child records in hierarchy data source.

The following code example shows how to bind the hierarchical list data into the Tree Grid component.

ASPX-CS

```

@using Syncfusion.Blazor.TreeGrid;
@using Syncfusion.Blazor.Data;
<SfTreeGrid ChildMapping="Children" TreeColumnIndex="1"
DataSource="@TreeData" TValue="BusinessObject" >
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public class BusinessObject
{
    public int TaskId { get; set; }
    public string TaskName { get; set; }
    public int Duration { get; set; }
    public int Progress { get; set; }
    public string Priority { get; set; }
}

```

```

public List<BusinessObject> Children { get; set; }
}
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
    List<BusinessObject> Record1 = new List<BusinessObject>();
    BusinessObject Child1 = new BusinessObject() { TaskId = 2, TaskName = "Plan
    timeline", Progress = 100, Duration = 5, Priority = "Normal" };
    BusinessObject Child2 = new BusinessObject() { TaskId = 3, TaskName = "Plan
    budget", Duration = 5, Progress = 100, Priority = "Low" };
    BusinessObject Child3 = new BusinessObject() { TaskId = 4, TaskName =
    "Allocate resources", Duration = 5, Progress = 100, Priority = "Critical" };
    Record1.Add(Child1);
    Record1.Add(Child2);
    Record1.Add(Child3);
    TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Planning",
    Duration = 10, Progress = 70, Children = Record1, Priority = "High" });
    List<BusinessObject> Record2 = new List<BusinessObject>();
    BusinessObject Child4 = new BusinessObject() { TaskId = 6, TaskName =
    "Software Specification", Progress = 60, Duration = 3, Priority = "Normal"
    };
    BusinessObject Child5 = new BusinessObject() { TaskId = 7, TaskName =
    "Develop Prototype", Duration = 3, Progress = 100, Priority = "Critical" };
    Record2.Add(Child4);
    Record2.Add(Child5);
    TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Design",
    Duration = 3, Progress = 86, Children = Record2, Priority = "High" });
    }
}

```

* ExpandCollapse State maintenance is not supported for Hierarchy Data.

* Batch Editing is not supported for Hierarchy Data.

* **PageSizeMode** --> **All** is not supported for Hierarchy Data.

* Row Drag and Drop feature is not supported for Hierarchy Data.

DynamicObject binding

Tree Grid is a generic component which is strongly bound to a model type. There are cases when the model type is unknown during compile type. In such cases the data can be bound to the tree grid as list of **DynamicObject**.

DynamicObject can be bound to tree grid by assigning to the [DataSource](#) property. Tree Grid can also perform all kind of supported data operations and editing in DynamicObject.

The [GetDynamicMemberNames](#) method of DynamicObject class must be overridden and return the property names to render and perform data operations, editing etc., while using DynamicObject.

ASPX-CS

```

@using Syncfusion.Blazor.TreeGrid
@using System.Dynamic
<SfTreeGrid DataSource="@TreeData" @ref="TreeGrid" AllowPaging="true"
Toolbar="@((new List<string>()) { "Add", "Edit", "Delete", "Update", "Cancel"
})" IdMapping="TaskID" ParentIdMapping="ParentID" TreeColumnIndex="1">

```



```

<TreeGridEditSettings AllowEditing="true" AllowAdding="true"
AllowDeleting="true" NewRowPosition="RowPosition.Below" />
<TreeGridPageSettings PageSize="2"
PageSizeMode="PageSizeMode.Root"></TreeGridPageSettings>
<TreeGridColumns>
<TreeGridColumn Field="TaskID" HeaderText="Task ID" Width="80"
IsPrimaryKey="true"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="240"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText="StartDate" Format="d"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code {
SfTreeGrid<DynamicDictionary> TreeGrid;
public List<DynamicDictionary> TreeData { get; set; }
protected override void OnInitialized()
{
this.TreeData = GetData().ToList();
}
public static List<DynamicDictionary> Data = new List<DynamicDictionary>();
public static int ParentRecordID { get; set; }
public static int ChildRecordID { get; set; }
public static List<DynamicDictionary> GetData()
{
Data.Clear();
ParentRecordID = 0;
ChildRecordID = 0;
for (var i = 1; i <= 3; i++)
{
Random ran = new Random();
DateTime start = new DateTime(1992, 06, 07);
int range = (DateTime.Today - start).Days;
DateTime startingDate = start.AddDays(ran.Next(range));
dynamic ParentRecord = new DynamicDictionary();
ParentRecord.TaskID = ++ParentRecordID;
ParentRecord.TaskName = "Parent Task " + i;
ParentRecord.StartDate = startingDate;
ParentRecord.Progress = ParentRecordID % 2 == 0 ? "In Progress" : "Open";
ParentRecord.Priority = ParentRecordID % 2 == 0 ? "High" : "Low";
ParentRecord.Duration = ParentRecordID % 2 == 0 ? 32 : 76;
ParentRecord.ParentID = null;
Data.Add(ParentRecord);
AddChildRecords(ParentRecordID);
}
return Data;
}
public static void AddChildRecords(int ParentId)
{
for (var i = 1; i < 3; i++)
{

```

```

Random ran = new Random();
DateTime start = new DateTime(1992, 06, 07);
int range = (DateTime.Today - start).Days;
DateTime startingDate = start.AddDays(ran.Next(range));
dynamic ChildRecord = new DynamicDictionary();
ChildRecord.TaskID = ++ParentRecordID;
ChildRecord.TaskName = "Child Task " + ++ChildRecordID;
ChildRecord.StartDate = startingDate;
ChildRecord.Progress = ParentRecordID % 3 == 0 ? "Validated" : "Closed";
ChildRecord.Priority = ParentRecordID % 3 == 0 ? "Low" : "Critical";
ChildRecord.Duration = ParentRecordID % 3 == 0 ? 64 : 98;
ChildRecord.ParentID = ParentId;
Data.Add(ChildRecord);
}
}
public class DynamicDictionary : DynamicObject
{
    Dictionary<string, object> dictionary = new Dictionary<string, object>();
    public override bool TryGetMember(GetMemberBinder binder, out object result)
    {
        string name = binder.Name;
        return dictionary.TryGetValue(name, out result);
    }
    public override bool TrySetMember(SetMemberBinder binder, object value)
    {
        dictionary[binder.Name] = value;
        return true;
    }
    public override System.Collections.Generic.IEnumerable<string>
    GetDynamicMemberNames()
    {
        return this.dictionary?.Keys;
    }
}
}

```

ExpandoObject binding

Tree Grid is a generic component which is strongly bound to a model type. There are cases when the model type is unknown during compile type. In such cases bind data to the tree grid as list of ExpandoObject.

ExpandoObject can be bound to Tree grid by assigning to the [DataSource](#) property. Tree Grid can also perform all kind of supported data operations and editing in ExpandoObject.

ASPX-CS

```

@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeData" @ref="TreeGrid" AllowPaging="true"
IdMapping="TaskID" ParentIdMapping="ParentID" TreeColumnIndex="1"
Toolbar="@ (new List<string>() { "Add", "Edit", "Delete", "Update", "Cancel"
}) ">
<TreeGridPageSettings PageSize="2"></TreeGridPageSettings>
<TreeGridEditSettings AllowAdding="true" AllowEditing="true"
AllowDeleting="true"
Mode="Syncfusion.Blazor.TreeGrid.EditMode.Cell"></TreeGridEditSettings>
<TreeGridColumns>

```

```

<TreeGridColumn Field="TaskID" HeaderText="Task ID" Width="80"
IsPrimaryKey="true"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText="StartDate" Format="d"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code {
SfTreeGrid<ExpandoObject> TreeGrid;
public List<ExpandoObject> TreeData { get; set; }
protected override void OnInitialized()
{
this.TreeData = GetData().ToList();
}
public static List<ExpandoObject> Data = new List<ExpandoObject>();
public static int ParentRecordID { get; set; }
public static int ChildRecordID { get; set; }
public static List<ExpandoObject> GetData()
{
Data.Clear();
ParentRecordID = 0;
ChildRecordID = 0;
for (var i = 1; i <= 60; i++)
{
Random ran = new Random();
DateTime start = new DateTime(1992, 06, 07);
int range = (DateTime.Today - start).Days;
DateTime startingDate = start.AddDays(ran.Next(range));
dynamic ParentRecord = new ExpandoObject();
ParentRecord.TaskID = ++ParentRecordID;
ParentRecord.TaskName = "Parent Task " + i;
ParentRecord.StartDate = startingDate;
ParentRecord.Progress = ParentRecordID % 2 == 0 ? "In Progress" : "Open";
ParentRecord.Priority = ParentRecordID % 2 == 0 ? "High" : "Low";
ParentRecord.Duration = ParentRecordID % 2 == 0 ? 32 : 76;
ParentRecord.ParentID = null;
Data.Add(ParentRecord);
AddChildRecords(ParentRecordID);
}
return Data;
}
public static void AddChildRecords(int ParentId)
{
for (var i = 1; i < 4; i++)
{
Random ran = new Random();
DateTime start = new DateTime(1992, 06, 07);
int range = (DateTime.Today - start).Days;
DateTime startingDate = start.AddDays(ran.Next(range));
dynamic ChildRecord = new ExpandoObject();

```

```

ChildRecord.TaskID = ++ParentRecordID;
ChildRecord.TaskName = "Child Task " + ++ChildRecordID;
ChildRecord.StartDate = startingDate;
ChildRecord.Progress = ParentRecordID % 3 == 0 ? "Validated" : "Closed";
ChildRecord.Priority = ParentRecordID % 3 == 0 ? "Low" : "Critical";
ChildRecord.Duration = ParentRecordID % 3 == 0 ? 64 : 98;
ChildRecord.ParentID = ParentId;
Data.Add(ChildRecord);
}
}
}

```

Herewith the list of reserved properties and the purpose used in TreeGrid are provided. It is recommended to avoid these reserved properties for Internal purpose(To get rid of conflicts).

Reserved keywords | Purpose

childRecords | Specifies the childRecords of a parentData

hasChildRecords | Specifies whether the record contains child records

hasFilteredChildRecords | Specifies whether the record contains filtered child records

expanded | Specifies whether the child records are expanded

parentRecord | Specifies the parentItem of childRecords

index | Specifies the index of current record

level | Specifies the hierarchy level of record

filterLevel | Specifies the hierarchy level of filtered record

parentIdMapping | Specifies the parentId

uniqueID | Specifies the unique ID of a record

parentUniqueID | Specifies the parent Unique ID of a record

checkboxState | Specifies the checkbox state of a record

Remote Service binding

To bind remote data to Tree Grid component, assign service data as an instance of **SfDataManager** to the [DataSource](#) property. To interact with remote data source, provide the endpoint **url** and define the [HasChildMapping](#) property of tree grid.

The [HasChildMapping](#) property maps the field name in data source, that denotes whether current record holds any child records. This is useful internally to show expand icon while binding child data on demand.

The Tree Grid provides **Load on Demand** support for rendering remote data. The Load on demand is considered in Tree Grid for the following actions.

- Expanding root nodes.
- Navigating pages, with paging enabled in Tree Grid.

When load on demand is enabled, all the root nodes are rendered in collapsed state at initial load.

When load on demand support is enabled in Tree Grid with paging, the current or active page's root node alone will be rendered in collapsed state. On expanding the root node, the child nodes will be loaded from the remote server.

When a root node is expanded, its child nodes are rendered and are cached locally, such that on consecutive expand/collapse actions on root node, the child nodes are loaded from the cache instead from the remote server.

Similarly, if the user navigates to a new page, the root nodes of that specific page, will be rendered with request to the remote server.

ASPX-CS

```
@using Syncfusion.Blazor.TreeGrid;
@using Syncfusion.Blazor.Data;
<SfTreeGrid IdMapping="TaskID" TValue="BusinessObject"
ParentIdMapping="ParentItem" HasChildMapping="isParent" AllowPaging="true"
TreeColumnIndex="1">
<SfDataManager Url="https://ej2services.syncfusion.com/production/web-
services/api/SelfReferenceData" CrossDomain="true"
Adaptor="Syncfusion.Blazor.Adaptors.WebApiAdaptor"></SfDataManager>
<TreeGridColumns>
<TreeGridColumn Field="TaskID" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public class BusinessObject
{
public int TaskID { get; set; }
public string TaskName { get; set; }
public DateTime? StartDate { get; set; }
public DateTime? EndDate { get; set; }
public String Progress { get; set; }
public String Priority { get; set; }
public double? Duration { get; set; }
public int? ParentID { get; set; }
public bool? isParent { get; set; }
public bool? Approved { get; set; }
public int? ParentItem { get; set; }
}
}
```

* By default, **SfDataManager** uses **ODataAdaptor** for remote data-binding.

* Based on the RESTful web services, set the corresponding adaptor to SfDataManager. Refer [here](#) for more details.

* Filtering and searching server-side data operations are not supported in load on demand.

Offline mode

On remote data binding, all tree grid actions such as paging, loading child on-demand, will be processed on server-side. To avoid postback, set the tree grid to load all data on initialization and make the actions process in client-side. To enable this behavior, use the *offline* property of **SfDataManager**.

ASPX-CS

```
@using Syncfusion.Blazor.TreeGrid;
@using Syncfusion.Blazor.Data;
<SfTreeGrid IdMapping="TaskID" TValue="BusinessObject"
ParentIdMapping="ParentItem" HasChildMapping="isParent" AllowPaging="true"
TreeColumnIndex="1">
<SfDataManager Url="https://ej2services.syncfusion.com/production/web-
services/api/SelfReferenceData" Offline="true" CrossDomain="true"
Adaptor="Syncfusion.Blazor.Adaptors.WebApiAdaptor"></SfDataManager>
<TreeGridColumns>
<TreeGridColumn Field="TaskID" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
```

LoadChildOnDemand

Tree Grid provides option to load the child records also during the initial rendering itself for remote data binding by setting the **LoadChildOnDemand** as **true**.

When the **LoadChildOnDemand** is enabled parent records are rendered in expanded state.

The following code example describes the behavior of the **LoadChildOnDemand** feature of Tree Grid.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="grid" TValue="SelfReferenceData" LoadChildOnDemand="true"
HasChildMapping="isParent" Height="315" IdMapping="TaskID"
ParentIdMapping="ParentID" TreeColumnIndex="1" AllowPaging="true">
<SfDataManager Url="api/Default" Adaptor="Adaptors.WebApiAdaptor"
CrossDomain="true"></SfDataManager>
<TreeGridPageSettings PageSize="2"></TreeGridPageSettings>
<TreeGridColumns>
<TreeGridColumn Field="TaskID" HeaderText="Task ID" IsPrimaryKey="true"
Width="80" TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="145"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="TextAlign.Right"></TreeGridColumn>
```

```
<TreeGridColumn Field="Progress" HeaderText="Progress"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="100"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
```

C#

```
namespace TreeGridComponent.Data {
public class SelfReferenceData
{
public static List<SelfReferenceData> tree = new List<SelfReferenceData>();
public int? TaskID { get; set; }
public string TaskName { get; set; }
public DateTime StartDate { get; set; }
public DateTime EndDate { get; set; }
public string Progress { get; set; }
public string Priority { get; set; }
public int Duration { get; set; }
public int? ParentID { get; set; }
public bool? isParent { get; set; }
public SelfReferenceData() { }
public static List<SelfReferenceData> GetTree()
{
tree.Clear();
if (tree.Count == 0)
{
int root = -1;
for (var t = 1; t <= 10; t++)
{
Random ran = new Random();
string math = (ran.Next() % 3) == 0 ? "High" : (ran.Next() % 2) == 0 ?
"Release Breaker" : "Critical";
string progr = (ran.Next() % 3) == 0 ? "Started" : (ran.Next() % 2) == 0 ?
"Open" : "In Progress";
root++;
int rootItem = tree.Count + root + 1;
tree.Add(new SelfReferenceData() { TaskID = rootItem, TaskName = "Parent
Task " + rootItem.ToString(), StartDate = new DateTime(1992, 06, 07),
EndDate = new DateTime(1994, 08, 25), isParent = true, Progress = progr,
Priority = math, Duration = ran.Next(1, 50) });
int parent = tree.Count;
for (var c = 0; c < 3; c++)
{
root++;
string val = ((parent + c + 1) % 3 == 0) ? "Low" : "Critical";
int parn = parent + c + 1;
progr = (ran.Next() % 3) == 0 ? "In Progress" : (ran.Next() % 2) == 0 ?
"Open" : "Validated";
int iD = tree.Count + root + 1;
tree.Add(new SelfReferenceData() { TaskID = iD, TaskName = "Child Task " +
iD.ToString(), StartDate = new DateTime(1992, 06, 07), EndDate = new
DateTime(1994, 08, 25), isParent = (((parent + c + 1) % 3) == 0), ParentID =
rootItem, Progress = progr, Priority = val, Duration = ran.Next(1, 50) });
if (((parent + c + 1) % 3) == 0)

```

```

{
    int immParent = tree.Count;
    for (var s = 0; s <= 1; s++)
    {
        root++;
        string Prior = (immParent % 2 == 0) ? "Validated" : "Normal";
        tree.Add(new SelfReferenceData() { TaskID = tree.Count + root + 1, TaskName
        = "Sub Task " + (tree.Count + root + 1).ToString(), StartDate = new
        DateTime(1992, 06, 07), EndDate = new DateTime(1994, 08, 25), isParent =
        false, ParentID = iD, Progress = (immParent % 2 == 0) ? "On Progress" :
        "Closed", Priority = Prior, Duration = ran.Next(1, 50) });
    }
}
}
}
}
return tree;
}
}

```

C#

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Primitives;
using Syncfusion.Blazor;
using Syncfusion.Blazor.Data;
using WebAPI.Data;
namespace WebAPI.Controller
{
    [Route("api/[controller]")]
    [ApiController]
    public class DefaultController : ControllerBase
    {
        public static List<SelfReferenceData> FlatData = new
        List<SelfReferenceData>();
        // GET: api/Default
        [HttpGet]
        public async Task<object> Get(int? code)
        {
            var queryString = Request.Query;
            FlatData.Clear();
            if (SelfReferenceData.tree.Count == 0)
                SelfReferenceData.GetTree();
            List<SelfReferenceData> data = SelfReferenceData.tree.ToList();
            bool isFiltered = false;
            if (queryString.Keys.Contains("$filter"))
            {
                StringValues filter;
                isFiltered = true;
                queryString.TryGetValue("$filter", out filter);
            }
        }
    }
}

```



```

string[] filterQuery = null;
if (filter[0].IndexOf('(') != -1 && filter[0].IndexOf(')') != -1)
{
    filterQuery = filter[0].Split('(', ')')[1].Split(" eq ");
}
else
{
    filterQuery = filter[0].Split(" eq ");
}
var field = filterQuery[0];
var value = filterQuery[1];
if (field == "ParentID" && value == "null")
{
    data = data.Where(p => p.ParentID == null).ToList();
}
}
if (queryString.Keys.Contains("$orderby"))
{
    StringValues srt;
    queryString.TryGetValue("$orderby", out srt);
    srt = srt.ToString().Replace("desc", "descending");
    IQueryable<SelfReferenceData> data1 = SortingExtend.Sort(data.AsQueryable(), srt);
    data = data1.ToList();
}
int count = data.Count;
if (queryString.Keys.Contains("$inlinecount"))
{
    StringValues Skip;
    StringValues Take;
    int skip = (queryString.TryGetValue("$skip", out Skip)) ?
    Convert.ToInt32(Skip[0]) : 0;
    int top = (queryString.TryGetValue("$top", out Take)) ?
    Convert.ToInt32(Take[0]) : data.Count();
    FlatData = data.Skip(skip).Take(top).ToList();
    var GroupData = SelfReferenceData.tree.ToList().GroupBy(rec => rec.ParentID)
    .Where(g => g.Key != null).ToDictionary(g => g.Key?.ToString(), g =>
    g.ToList());
    foreach (var Record in FlatData.ToList())
    {
        if (GroupData.ContainsKey(Record.TaskID.ToString()))
        {
            var ChildGroup = GroupData[Record.TaskID.ToString()];
            if (ChildGroup?.Count > 0)
            AppendChildren(ChildGroup, Record, GroupData);    /////    appending the child
            records for the respective parent records
        }
    }
    return new { Items = FlatData, FlatData.Count };
}
else
{
    return SelfReferenceData.GetTree();
}
}

```

```

private void AppendChildren(List<SelfReferenceData> ChildRecords,
SelfReferenceData ParentItem, Dictionary<string, List<SelfReferenceData>>
GroupData)
{
    var queryString = Request.Query;
    string TaskId = ParentItem.TaskID.ToString();
    if (queryString.Keys.Contains("$orderby"))
    {
        StringValues srt;
        queryString.TryGetValue("$orderby", out srt);
        srt = srt.ToString().Replace("desc", "descending");
        List<SelfReferenceData> SortedChildRecords =
        SortingExtend.Sort(ChildRecords.AsQueryable(), srt).ToList();
        var index = FlatData.IndexOf(ParentItem);
        foreach (var Child in SortedChildRecords)
        {
            string ParentId = Child.ParentID.ToString();
            if (TaskId == ParentId)
            {
                if (FlatData.IndexOf(Child) == -1)
                ((IList)FlatData).Insert(++index, Child);
                if (GroupData.ContainsKey(Child.TaskID.ToString()))
                {
                    var DeepChildRecords = GroupData[Child.TaskID.ToString()];
                    if (DeepChildRecords?.Count > 0)
                    AppendChildren(DeepChildRecords, Child, GroupData);
                }
            }
        }
    }
    else
    {
        var index = FlatData.IndexOf(ParentItem);
        foreach (var Child in ChildRecords)
        {
            string ParentId = Child.ParentID.ToString();
            if (TaskId == ParentId)
            {
                if (FlatData.IndexOf(Child) == -1)
                ((IList)FlatData).Insert(++index, Child);
                if (GroupData.ContainsKey(Child.TaskID.ToString()))
                {
                    var DeepChildRecords = GroupData[Child.TaskID.ToString()];
                    if (DeepChildRecords?.Count > 0)
                    AppendChildren(DeepChildRecords, Child, GroupData);
                }
            }
        }
    }
}

```

<!-- Custom Adaptor

You can create your own adaptor by extending the built-in adaptors. The following demonstrates custom adaptor approach and how to add a serial number for the records by overriding the built-in response processing using the **processResponse** method of the **ODataAdaptor**.

-->

Sending additional parameters to the Rest Web Services

To add a custom parameter to the data request, use the **addParams** method of **Query**. Assign the **Query** object with additional parameters to the tree grid [Query](#) property.

ASPX-CS

```
@using Syncfusion.Blazor.TreeGrid;
@using Syncfusion.Blazor.Data;
<SfTreeGrid IdMapping="TaskID" TValue="BusinessObject"
ParentIdMapping="ParentItem" HasChildMapping="isParent" Query=@TreeGridQuery
AllowPaging="true" TreeColumnIndex="1">
<SfDataManager Url="https://ej2services.syncfusion.com/production/web-
services/api/SelfReferenceData" CrossDomain="true"
Adaptor="Syncfusion.Blazor.Adaptors.WebApiAdaptor"></SfDataManager>
<TreeGridColumns>
<TreeGridColumn Field="TaskID" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public string ParamValue = "true";
public Query TreeGridQuery { get; set; }
protected override void OnInitialized()
{
TreeGridQuery = new Query().AddParams("ej2treegrid", ParamValue);
}
public class BusinessObject
{
public int TaskID { get; set; }
public string TaskName { get; set; }
public DateTime? StartDate { get; set; }
public DateTime? EndDate { get; set; }
public String Progress { get; set; }
public String Priority { get; set; }
public double? Duration { get; set; }
public int? ParentID { get; set; }
public bool? isParent { get; set; }
public bool? Approved { get; set; }
public int? ParentItem { get; set; }
}
}
```

<!--Handling HTTP error

During server interaction from the tree grid, some server-side exceptions may occur, and you can acquire those error messages or exception details

in the client-side using the [ActionFailure](#) event.

The argument passed to the [ActionFailure](#) event contains the error details returned from the server.

The [ActionFailure](#) event will be triggered not only for the server errors, but

also when there is an exception while processing the tree grid actions.

-->

Entity Framework

Follow the below steps to consume data from the **Entity Framework** in the Tree Grid component.

Create DbContext class

The first step is to create a DbContext class called **TasksContext** to connect to a Microsoft SQL Server database.

CSHARP

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.EntityFrameworkCore;
namespace TreeGridWebApiEFSample.Shared.DataAccess
{
    public class TasksContext: DbContext
    {
        public virtual DbSet<Shared.Models.Task> Tasks { get; set; }
        protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
        {
            if (!optionsBuilder.IsConfigured)
            {
                optionsBuilder.UseSqlServer(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=D:\Documentation\TreeGridWebA
piEFSample\TreeGridWebApiEFSample\Shared\App_Data\TreeGridDB.mdf;Integrated
Security=True;Connect Timeout=30");
            }
        }
    }
}
```

Create data access layer to perform data operation

Now, create a class named **TasksDataAccessLayer**, which act as a data access layer for retrieving the records and also insert, update and delete the records from the database table.

CSHARP

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
```

```

using TreeGridWebApiEFSample.Shared.Models;
using Microsoft.EntityFrameworkCore;
namespace TreeGridWebApiEFSample.Shared.DataAccess
{
    public class TasksDataAccessLayer
    {
        TasksContext treedb = new TasksContext();
        //To Get all Task details
        public IEnumerable<Shared.Models.Task> GetAllRecords()
        {
            try
            {
                return treedb.Tasks.ToList();
            }
            catch
            {
                throw;
            }
        }
    }
}

```

Creating Web API Controller

A Web API Controller has to be created which allows Tree Grid directly to consume data from the Entity framework.

CSHARP

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using TreeGridWebApiEFSample.Shared.Models;
using TreeGridWebApiEFSample.Shared.DataAccess;
using Microsoft.Extensions.Primitives;
using System.Web;
using Microsoft.AspNetCore.Http;
using Newtonsoft.Json.Linq;
namespace TreeGridWebApiEFSample.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class TreeGridController : ControllerBase
    {
        TasksDataAccessLayer db = new TasksDataAccessLayer();
        // GET: api/<TreeGridController>
        [HttpGet]
        public object Get()
        {
            var queryString = Request.Query;
            IQueryable<TreeGridWebApiEFSample.Shared.Models.Task> data1 =
            db.GetAllRecords().AsQueryable();
            if (queryString.Keys.Contains("$filter") &&
            !queryString.Keys.Contains("$top"))
            {

```

```

StringValues filter;
queryString.TryGetValue("$filter", out filter);
int fltr = Int32.Parse(filter[0].ToString().Split("eq")[1]);
data1 = data1.Where(f => f.ParentID == fltr).AsQueryable();
return new { Items = data1, Count = data1.Count() };
}
if (queryString.Keys.Contains("$select"))
{
    data1 = (from ord in data1
    select new TreeGridWebApiEFSample.Shared.Models.Task
    {
        ParentID = ord.ParentID
    }
    );
    return data1;
}
data1 = data1.Where(p => p.ParentID == null);
var count = data1.Count();
if (queryString.Keys.Contains("$inlinecount"))
{
    StringValues Skip;
    StringValues Take;
    int skip = (queryString.TryGetValue("$skip", out Skip)) ?
    Convert.ToInt32(Skip[0]) : 0;
    int top = (queryString.TryGetValue("$top", out Take)) ?
    Convert.ToInt32(Take[0]) : data1.Count();
    return new { Items = data1.Skip(skip).Take(top), Count = count };
}
else
{
    return data1;
}
}
}
}
}

```

Configure Tree Grid component using Web API adaptor

Now, the Tree Grid can be configured using the '**SfDataManager**' to interact with the created Web API and consume the data appropriately. To interact with web API, use WebApiAdaptor.

ASPX-CS

```

@using Syncfusion.Blazor.Grids
@using Syncfusion.Blazor.Data
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.TreeGrid
@using Shared.Models
<SfTreeGrid TValue="Shared.Models.Task" @ref="treeGrid" IdMapping="TaskID"
AllowPaging="true"
ParentIdMapping="ParentID" HasChildMapping="IsParent"
TreeColumnIndex="0">
<SfDataManager Url="api/TreeGrid" Adaptor="Adaptors.WebApiAdaptor"
CrossDomain="true"></SfDataManager>
<TreeGridColumn>
<TreeGridColumn Field="TaskID" HeaderText="Task ID" Width="80"
IsPrimaryKey="true" Type=ColumnType.Number></TreeGridColumn>

```

```

<TreeGridColumn Field="TaskName" HeaderText="Task Name" Width="160"
Type=ColumnType.String></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="160"
Type=ColumnType.Number></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="160"
Type=ColumnType.Number></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid<Shared.Models.Task> treeGrid { get; set; }
}

```

To perform Tree Grid CRUD operation using Entity Framework. You can refer [here](#).

You can find the fully working sample [here](#).

Custom Binding in Blazor TreeGrid Component

The [SfDataManager](#) has custom adaptor support which allows to perform manual operations on the data. This can be utilized for implementing custom data binding and editing operations in the Tree Grid component.

For implementing custom data binding in the Tree Grid, the **DataAdaptor** class is used. This abstract class acts as a base class for the custom adaptor.

The **DataAdaptor** abstract class has both synchronous and asynchronous method signatures which can be overridden in the custom adaptor. Following are the method signatures present in this class,

CSHARP

```

public abstract class DataAdaptor
{
    /// <summary>
    /// Performs data Read operation synchronously.
    /// </summary>
    public virtual object Read(DataManagerRequest dataManagerRequest, string key
= null)
    /// <summary>
    /// Performs data Read operation asynchronously.
    /// </summary>
    public virtual Task<object> ReadAsync(DataManagerRequest dataManagerRequest,
string key = null)
    /// <summary>
    /// Performs Insert operation synchronously.
    /// </summary>
    public virtual object Insert(DataManager dataManager, object data, string
key)
    /// <summary>
    /// Performs Insert operation asynchronously.
    /// </summary>
    public virtual Task<object> InsertAsync(DataManager dataManager, object
data, string key)
    /// <summary>
    /// Performs Remove operation synchronously.
    /// </summary>
    public virtual object Remove(DataManager dataManager, object data, string
keyField, string key)
}

```

```

/// <summary>
/// Performs Remove operation asynchronously..
/// </summary>
public virtual Task<object> RemoveAsync(DataManager dataManager, object
data, string keyField, string key)
/// <summary>
/// Performs Update operation synchronously.
/// </summary>
public virtual object Update (DataManager dataManager, object data, string
keyField, string key)
/// <summary>
/// Performs Update operation asynchronously.
/// </summary>
public virtual Task<object> UpdateAsync(DataManager dataManager, object
data, string keyField, string key)
/// <summary>
/// Performs Batch CRUD operations synchronously.
/// </summary>
public virtual object BatchUpdate(DataManager dataManager, object
changedRecords, object addedRecords, object deletedRecords, string keyField,
string key, int? dropIndex)
/// <summary>
/// Performs Batch CRUD operations asynchronously.
/// </summary>
public virtual Task<object> BatchUpdateAsync(DataManager dataManager, object
changedRecords, object addedRecords, object deletedRecords, string keyField,
string key, int? dropIndex)
}

```

Data binding

The custom data binding can be performed in the Tree Grid component by providing the custom adaptor class and overriding the **Read** or **ReadAsync** method of the **DataAdaptor** abstract class.

The following sample code demonstrates implementing custom data binding using custom adaptor,

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
@using Syncfusion.Blazor.Data;
@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor;
<SfTreeGrid TValue="SelfReferenceData" AllowFiltering="true"
HasChildMapping="isParent" IdMapping="TaskID" ParentIdMapping="ParentID"
TreeColumnIndex="1" AllowPaging="true" AllowSorting="true">
<SfDataManager AdaptorInstance="@typeof(CustomAdaptor) "
Adaptor="Adaptors.CustomAdaptor"></SfDataManager>
<TreeGridPageSettings PageSize="3"></TreeGridPageSettings>
<TreeGridFilterSettings
Type="Syncfusion.Blazor.TreeGrid.FilterType.FilterBar"></TreeGridFilterSetti
ngs>
<TreeGridColumn>
<TreeGridColumn Field="TaskID" HeaderText="Task ID" IsPrimaryKey="true"
Width="80" TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="145"></TreeGridColumn>

```



```

<TreeGridColumn Field="StartDate" HeaderText="Start Date" Format="d"
Type=ColumnType.Date Width="88"
TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="100"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public static List<SelfReferenceData> TreeData { get; set; }
protected override void OnInitialized()
{
TreeData = SelfReferenceData.GetTree().ToList();
}
// Implementing custom adaptor by extending the DataAdaptor class
public class CustomAdaptor : DataAdaptor
{
// Performs data Read operation
public override object Read(DataManagerRequest dm, string key = null)
{
IEnumerable<SelfReferenceData> DataSource = TreeData;
if (dm.Search != null && dm.Search.Count > 0)
{
// Searching
DataSource = DataOperations.PerformSearching(DataSource, dm.Search);
}
if (dm.Sorted != null && dm.Sorted.Count > 0)
{
// Sorting
DataSource = DataOperations.PerformSorting(DataSource, dm.Sorted);
}
if (dm.Where != null && dm.Where.Count > 0)
{
// Filtering
DataSource = DataOperations.PerformFiltering(DataSource, dm.Where,
dm.Where[0].Operator);
}
int count = DataSource.Cast<SelfReferenceData>().Count();
if (dm.Skip != 0)
{
//Paging
DataSource = DataOperations.PerformSkip(DataSource, dm.Skip);
}
if (dm.Take != 0)
{
DataSource = DataOperations.PerformTake(DataSource, dm.Take);
}
return dm.RequiresCounts ? new DataResult() { Result = DataSource, Count =
count } : (object)DataSource;
}
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class SelfReferenceData
{
public List<SelfReferenceData> tree = new List<SelfReferenceData>();
public int? TaskID { get; set; }
public string TaskName { get; set; }
public DateTime StartDate { get; set; }
public DateTime EndDate { get; set; }
public String Progress { get; set; }
public String Priority { get; set; }
public int Duration { get; set; }
public int? ParentID { get; set; }
public bool? isParent { get; set; }
public SelfReferenceData() { }
public List<SelfReferenceData> GetTree()
{
if (tree.Count == 0)
{
int root = -1;
for (var t = 1; t <= 8; t++)
{
Random ran = new Random();
string math = (ran.Next() % 3) == 0 ? "High" : (ran.Next() % 2) == 0 ?
"Release Breaker" : "Critical";
string progr = (ran.Next() % 3) == 0 ? "Started" : (ran.Next() % 2) == 0 ?
"Open" : "In Progress";
root++;
int rootItem = tree.Count + root + 1;
tree.Add(new SelfReferenceData() { TaskID = rootItem, TaskName = "Parent
Task " + rootItem.ToString(), StartDate = new DateTime(1992, 06, 07),
EndDate = new DateTime(1994, 08, 25), isParent = true, ParentID = null,
Progress = progr, Priority = math, Duration = ran.Next(1, 50) });
int parent = tree.Count;
for (var c = 0; c < 3; c++)
{
root++;
string val = ((parent + c + 1) % 3 == 0) ? "Low" : "Critical";
int parn = parent + c + 1;
progr = (ran.Next() % 3) == 0 ? "In Progress" : (ran.Next() % 2) == 0 ?
"Open" : "Validated";
int iD = tree.Count + root + 1;
tree.Add(new SelfReferenceData() { TaskID = iD, TaskName = "Child Task " +
iD.ToString(), StartDate = new DateTime(1992, 06, 07), EndDate = new
DateTime(1994, 08, 25), isParent = (((parent + c + 1) % 3) == 0), ParentID =
rootItem, Progress = progr, Priority = val, Duration = ran.Next(1, 50) });
if (((parent + c + 1) % 3) == 0)
{
int immParent = tree.Count;
for (var s = 0; s <= 1; s++)
{
root++;
string Prior = (immParent % 2 == 0) ? "Validated" : "Normal";
tree.Add(new SelfReferenceData() { TaskID = tree.Count + root + 1, TaskName
= "Sub Task " + (tree.Count + root + 1).ToString(), StartDate = new
DateTime(1992, 06, 07), EndDate = new DateTime(1994, 08, 25), isParent =

```

```

false, ParentID = iD, Progress = (immParent % 2 == 0) ? "On Progress" :
"Closed", Priority = Prior, Duration = ran.Next(1, 50) });
}
}
}
}
}
return tree;
}
}
}

```

If the **DataManagerRequest.RequiresCounts** value is **true**, then the Read/ReadAsync return value must be of **DataRow** with properties **Result** whose value is a collection of records and **Count** whose value is the total number of records. If the **DataManagerRequest.RequiresCounts** is **false**, then simply send the collection of records.

If the Read/ReadAsync method is not overridden in the custom adaptor then it will be handled by the default read handler.

Inject service into Custom Adaptor

If you want to inject some of your service into Custom Adaptor and use the service, then you can achieve your requirement by using below way.

Initially, the CustomAdaptor class must be added as AddScoped in **Startup.cs** file.

CSHARP

```

public void ConfigureServices(IServiceCollection services)
{
    ...
    services.AddSingleton<TaskDataAccessLayer>();
    services.AddScoped<CustomAdaptor>();
    services.AddScoped<ServiceClass>();
}

```

The following sample code demonstrates injecting service into Custom Adaptor,

ASPX-CS

```

@using Syncfusion.Blazor.Data
@using Syncfusion.Blazor.Grids
@using Syncfusion.Blazor
@using Syncfusion.Blazor.TreeGrid
<SfTreeGrid TValue="SelfReferenceData" AllowFiltering="true"
HasChildMapping="isParent" IdMapping="TaskID" ParentIdMapping="ParentID"
TreeColumnIndex="1" AllowPaging="true" AllowSorting="true">
<SfDataManager AdaptorInstance="@typeof(CustomAdaptor)"
Adaptor="Adaptors.CustomAdaptor"></SfDataManager>
<TreeGridPageSettings PageSize="3"></TreeGridPageSettings>
<TreeGridFilterSettings
Type="Syncfusion.Blazor.TreeGrid.FilterType.FilterBar"></TreeGridFilterSetti
ngs>
<TreeGridColumn>
<TreeGridColumn Field="TaskID" HeaderText="Task ID" IsPrimaryKey="true"
Width="80" TextAlign="TextAlign.Right"></TreeGridColumn>

```

```

<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="145"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText="Start Date" Format="d"
Type=ColumnType.Date Width="88"
TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="100"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public class CustomAdaptor : DataAdaptor
{
//here you can inject your service
[Inject]
public TaskDataAccessLayer context { get; set; } = new
TaskDataAccessLayer();
// Performs data Read operation
public override object Read(DataManagerRequest dm, string key = null)
{
IEnumerable<SelfReferenceData> DataSource = context.GetTree();
if (dm.Search != null && dm.Search.Count > 0)
{
// Searching
DataSource = DataOperations.PerformSearching(DataSource, dm.Search);
}
if (dm.Sorted != null && dm.Sorted.Count > 0)
{
// Sorting
DataSource = DataOperations.PerformSorting(DataSource, dm.Sorted);
}
if (dm.Where != null && dm.Where.Count > 0)
{
// Filtering
DataSource = DataOperations.PerformFiltering(DataSource, dm.Where,
dm.Where[0].Operator);
}
int count = DataSource.Cast<SelfReferenceData>().Count();
if (dm.Skip != 0)
{
//Paging
DataSource = DataOperations.PerformSkip(DataSource, dm.Skip);
}
if (dm.Take != 0)
{
DataSource = DataOperations.PerformTake(DataSource, dm.Take);
}
return dm.RequiresCounts ? new DataResult() { Result = DataSource, Count =
count } : (object)DataSource;
}
}
}

```

Custom adaptor as Component

Custom Adaptor can be created as a component when `DataAdaptor` is extended from `OwningComponentBase`. Custom Adaptor can be created from any of the two versions of the class, `DataAdaptor` and `DataAdaptor<T>`.

Ensure to register your service in **Startup.cs** file.

CSHARP

```
public void ConfigureServices(IServiceCollection services)
{
    ...
    services.AddScoped<SelfReferenceData>();
}
```

The following sample code demonstrates creating Custom Adaptor as a component,

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.Data;
@using Syncfusion.Blazor.TreeGrid;
@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor;
<SfTreeGrid TValue="SelfReferenceData" AllowFiltering="true"
HasChildMapping="isParent" IdMapping="TaskID" ParentIdMapping="ParentID"
TreeColumnIndex="1" AllowPaging="true" AllowSorting="true" Toolbar="@ (new
List<string>() { "Add", "Edit", "Delete", "Update", "Cancel", "Search" })">
<SfDataManager Adaptor="Adaptors.CustomAdaptor">
<CustomAdaptorComponent></CustomAdaptorComponent>
</SfDataManager>
<TreeGridPageSettings PageSize="3"></TreeGridPageSettings>
<TreeGridEditSettings AllowEditing="true" AllowAdding="true"
AllowDeleting="true"
Mode="Syncfusion.Blazor.TreeGrid.EditMode.Row"></TreeGridEditSettings>
<TreeGridFilterSettings
Type="Syncfusion.Blazor.TreeGrid.FilterType.FilterBar"></TreeGridFilterSetti
ngs>
<TreeGridColumns>
<TreeGridColumn Field="TaskID" HeaderText="Task ID" IsPrimaryKey="true"
Width="80" TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="145"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText="Start Date" Format="d"
Type=ColumnType.Date Width="88"
TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="100"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
```

C#

```

namespace TreeGridComponent.Data {
public class SelfReferenceData
{
public List<SelfReferenceData> tree = new List<SelfReferenceData>();
public int? TaskID { get; set; }
public string TaskName { get; set; }
public DateTime StartDate { get; set; }
public DateTime EndDate { get; set; }
public String Progress { get; set; }
public String Priority { get; set; }
public int Duration { get; set; }
public int? ParentID { get; set; }
public bool? isParent { get; set; }
public SelfReferenceData() { }
public List<SelfReferenceData> GetTree()
{
if (tree.Count == 0)
{
int root = -1;
for (var t = 1; t <= 8; t++)
{
Random ran = new Random();
string math = (ran.Next() % 3) == 0 ? "High" : (ran.Next() % 2) == 0 ?
"Release Breaker" : "Critical";
string progr = (ran.Next() % 3) == 0 ? "Started" : (ran.Next() % 2) == 0 ?
"Open" : "In Progress";
root++;
int rootItem = tree.Count + root + 1;
tree.Add(new SelfReferenceData() { TaskID = rootItem, TaskName = "Parent
Task " + rootItem.ToString(), StartDate = new DateTime(1992, 06, 07),
EndDate = new DateTime(1994, 08, 25), isParent = true, ParentID = null,
Progress = progr, Priority = math, Duration = ran.Next(1, 50) });
int parent = tree.Count;
for (var c = 0; c < 3; c++)
{
root++;
string val = ((parent + c + 1) % 3 == 0) ? "Low" : "Critical";
int parn = parent + c + 1;
progr = (ran.Next() % 3) == 0 ? "In Progress" : (ran.Next() % 2) == 0 ?
"Open" : "Validated";
int iD = tree.Count + root + 1;
tree.Add(new SelfReferenceData() { TaskID = iD, TaskName = "Child Task " +
iD.ToString(), StartDate = new DateTime(1992, 06, 07), EndDate = new
DateTime(1994, 08, 25), isParent = (((parent + c + 1) % 3) == 0), ParentID =
rootItem, Progress = progr, Priority = val, Duration = ran.Next(1, 50) });
if (((parent + c + 1) % 3) == 0)
{
int immParent = tree.Count;
for (var s = 0; s <= 1; s++)
{
root++;
string Prior = (immParent % 2 == 0) ? "Validated" : "Normal";
tree.Add(new SelfReferenceData() { TaskID = tree.Count + root + 1, TaskName
= "Sub Task " + (tree.Count + root + 1).ToString(), StartDate = new
DateTime(1992, 06, 07), EndDate = new DateTime(1994, 08, 25), isParent =

```

```

false, ParentID = iD, Progress = (immParent % 2 == 0) ? "On Progress" :
"Closed", Priority = Prior, Duration = ran.Next(1, 50) });
}
}
}
}
}
return tree;
}
}
}
}

```

The following sample code demonstrates `DataAdaptor` extended from `OwningComponentBase`. This provides the possibility to request multiple services.

CSHARP

```

[CustomAdaptorComponent.razor]
@using Syncfusion.Blazor;
@using Syncfusion.Blazor.Data;
@using TreeGridComponent.Data;
@using Newtonsoft.Json
@inherits DataAdaptor
<CascadingValue Value="@this">
@ChildContent
</CascadingValue>
@code {
    [Parameter]
    [JsonIgnore]
    public RenderFragment ChildContent { get; set; }
    SelfReferenceData TreeData;
    public override object Read(DataManagerRequest dm, string key = null)
    {
        TreeData =
            (SelfReferenceData)ScopedServices.GetService(typeof(SelfReferenceData));
        IEnumerable<SelfReferenceData> DataSource =
            (IEnumerable<SelfReferenceData>)TreeData.GetTree().Take(30);
        if (dm.Search != null && dm.Search.Count > 0)
        {
            // Searching
            DataSource = DataOperations.PerformSearching(DataSource, dm.Search);
        }
        if (dm.Sorted != null && dm.Sorted.Count > 0)
        {
            // Sorting
            DataSource = DataOperations.PerformSorting(DataSource, dm.Sorted);
        }
        if (dm.Where != null && dm.Where.Count > 0)
        {
            // Filtering
            DataSource = DataOperations.PerformFiltering(DataSource, dm.Where,
                dm.Where[0].Operator);
        }
        int count = DataSource.Cast<SelfReferenceData>().Count();
        if (dm.Skip != 0)
        {

```

```
//Paging
DataSource = DataOperations.PerformSkip(DataSource, dm.Skip);
}
if (dm.Take != 0)
{
DataSource = DataOperations.PerformTake(DataSource, dm.Take);
}
return dm.RequiresCounts ? new DataResult() { Result = DataSource, Count =
count } : (object)DataSource;
}
}
```

CRUD operation

The CRUD operations for the custom bounded data in the Tree Grid component can be implemented by overriding the following CRUD methods of the **DataAdaptor** abstract class,

- **Insert/InsertAsync**
- **Remove/RemoveAsync**
- **Update/UpdateAsync**
- **BatchUpdate/BatchUpdateAsync**

While using batch editing in tree grid, use BatchUpdate/BatchUpdateAsync method to handle the corresponding CRUD operation

The following sample code demonstrates implementing CRUD operations for the custom bounded data,

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
@using Syncfusion.Blazor.Data;
@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor;
<SfTreeGrid TValue="SelfReferenceData" AllowFiltering="true"
HasChildMapping="isParent" IdMapping="TaskID" ParentIdMapping="ParentID"
TreeColumnIndex="1"
AllowPaging="true" AllowSorting="true" Toolbar="@ (new List<string>() {
"Add", "Edit", "Delete", "Update", "Cancel", "Search" })">
<SfDataManager AdaptorInstance="@typeof(CustomAdaptor)"
Adaptor="Adaptors.CustomAdaptor"></SfDataManager>
<TreeGridPageSettings PageSize="3"></TreeGridPageSettings>
<TreeGridEditSettings AllowEditing="true" AllowAdding="true"
AllowDeleting="true"
Mode="Syncfusion.Blazor.TreeGrid.EditMode.Row"></TreeGridEditSettings>
<TreeGridFilterSettings
Type="Syncfusion.Blazor.TreeGrid.FilterType.FilterBar"></TreeGridFilterSettings>
<TreeGridColumn>
<TreeGridColumn Field="TaskID" HeaderText="Task ID" IsPrimaryKey="true"
Width="80" TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="145"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText="Start Date" Format="d"
Type=ColumnType.Date Width="88"
TextAlign="TextAlign.Right"></TreeGridColumn>
```



```

<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="100"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public static List<SelfReferenceData> TreeData { get; set; }
protected override void OnInitialized()
{
TreeData = SelfReferenceData.GetTree().ToList();
}
// Implementing custom adaptor by extending the DataAdaptor class
public class CustomAdaptor : DataAdaptor
{
// Performs data Read operation
public override object Read(DataManagerRequest dm, string key = null)
{
IEnumerable<SelfReferenceData> DataSource = TreeData;
if (dm.Search != null && dm.Search.Count > 0)
{
// Searching
DataSource = DataOperations.PerformSearching(DataSource, dm.Search);
}
if (dm.Sorted != null && dm.Sorted.Count > 0)
{
// Sorting
DataSource = DataOperations.PerformSorting(DataSource, dm.Sorted);
}
if (dm.Where != null && dm.Where.Count > 0)
{
// Filtering
DataSource = DataOperations.PerformFiltering(DataSource, dm.Where,
dm.Where[0].Operator);
}
int count = DataSource.Cast<SelfReferenceData>().Count();
if (dm.Skip != 0)
{
//Paging
DataSource = DataOperations.PerformSkip(DataSource, dm.Skip);
}
if (dm.Take != 0)
{
DataSource = DataOperations.PerformTake(DataSource, dm.Take);
}
return dm.RequiresCounts ? new DataResult() { Result = DataSource, Count =
count } : (object)DataSource;
}
// Performs Insert operation
public override object Insert(DataManager dm, object value, string key)
{
TreeData.Insert(0, value as SelfReferenceData);
return value;
}
// Performs Remove operation

```

```

public override object Remove(DataManager dm, object value, string keyField,
string key)
{
    TreeData.Remove(TreeData.Where(or => or.TaskID ==
int.Parse(value.ToString())).FirstOrDefault());
    return value;
}
// Performs Update operation
public override object Update(DataManager dm, object value, string keyField,
string key)
{
    var data = TreeData.Where(or => or.TaskID == (value as
SelfReferenceData).TaskID).FirstOrDefault();
    if (data != null)
    {
        data.TaskID = (value as SelfReferenceData).TaskID;
        data.TaskName = (value as SelfReferenceData).TaskName;
        data.StartDate = (value as SelfReferenceData).StartDate;
        data.Duration = (value as SelfReferenceData).Duration;
        data.Priority = (value as SelfReferenceData).Priority;
        data.Progress = (value as SelfReferenceData).Progress;
    }
    return value;
}
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class SelfReferenceData
{
    public List<SelfReferenceData> tree = new List<SelfReferenceData>();
    public int? TaskID { get; set; }
    public string TaskName { get; set; }
    public DateTime StartDate { get; set; }
    public DateTime EndDate { get; set; }
    public string Progress { get; set; }
    public string Priority { get; set; }
    public int Duration { get; set; }
    public int? ParentID { get; set; }
    public bool? isParent { get; set; }
    public SelfReferenceData() { }
    public List<SelfReferenceData> GetTree()
    {
        if (tree.Count == 0)
        {
            int root = -1;
            for (var t = 1; t <= 8; t++)
            {
                Random ran = new Random();
                string math = (ran.Next() % 3) == 0 ? "High" : (ran.Next() % 2) == 0 ?
"Release Breaker" : "Critical";
                string progr = (ran.Next() % 3) == 0 ? "Started" : (ran.Next() % 2) == 0 ?
"Open" : "In Progress";
                root++;
            }
        }
    }
}
}

```

```

int rootItem = tree.Count + root + 1;
tree.Add(new SelfReferenceData() { TaskID = rootItem, TaskName = "Parent
Task " + rootItem.ToString(), StartDate = new DateTime(1992, 06, 07),
EndDate = new DateTime(1994, 08, 25), isParent = true, ParentID = null,
Progress = progr, Priority = math, Duration = ran.Next(1, 50) });
int parent = tree.Count;
for (var c = 0; c < 3; c++)
{
    root++;
    string val = ((parent + c + 1) % 3 == 0) ? "Low" : "Critical";
    int parn = parent + c + 1;
    progr = (ran.Next() % 3) == 0 ? "In Progress" : (ran.Next() % 2) == 0 ?
    "Open" : "Validated";
    int iD = tree.Count + root + 1;
    tree.Add(new SelfReferenceData() { TaskID = iD, TaskName = "Child Task " +
    iD.ToString(), StartDate = new DateTime(1992, 06, 07), EndDate = new
    DateTime(1994, 08, 25), isParent = (((parent + c + 1) % 3) == 0), ParentID =
    rootItem, Progress = progr, Priority = val, Duration = ran.Next(1, 50) });
    if (((parent + c + 1) % 3) == 0)
    {
        int immParent = tree.Count;
        for (var s = 0; s <= 1; s++)
        {
            root++;
            string Prior = (immParent % 2 == 0) ? "Validated" : "Normal";
            tree.Add(new SelfReferenceData() { TaskID = tree.Count + root + 1, TaskName
            = "Sub Task " + (tree.Count + root + 1).ToString(), StartDate = new
            DateTime(1992, 06, 07), EndDate = new DateTime(1994, 08, 25), isParent =
            false, ParentID = iD, Progress = (immParent % 2 == 0) ? "On Progress" :
            "Closed", Priority = Prior, Duration = ran.Next(1, 50) });
        }
    }
}
return tree;
}
}
}
}
}

```

Columns in Blazor TreeGrid Component

The column definitions are used as the datasource schema in the Tree Grid. This plays a vital role in rendering column values in the required format. The tree grid operations such as sorting, filtering and searching etc. are performed based on the column definitions. The [Field](#) property of [TreeGridColumnns](#) tag helper is necessary to map the data source values in Tree Grid columns.

If the column [Field](#) is not specified in the dataSource, the column values will be empty. `
`[TreeColumnIndex](#) property denotes the column that is used to expand and collapse child rows.

Complex data binding

The complex data binding in the Tree Grid can be achieved by using the dot(.) operator in the column.field. In the below examples **Task.TaskName** and **Task.Duration** are complex data.

ASPX-CS

```

@using Syncfusion.Blazor.TreeGrid
<SfTreeGrid DataSource="@TreeData" AllowPaging="true" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Task.TaskName" HeaderText="Task Name" Width="160">
</TreeGridColumn>
<TreeGridColumn Field="Task.Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority" Width="80">
</TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
public class BusinessObject
{
public int TaskId { get; set; }
public TaskDetails Task { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
}
public class TaskDetails
{
public string TaskName { get; set; }
public int Duration { get; set; }
}
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
TreeData.Add(new BusinessObject() { Task = new TaskDetails() { TaskName =
"Parent Task 1", Duration = 50000 }, TaskId = 1, Progress = 70, ParentId =
null, Priority = "High" });
TreeData.Add(new BusinessObject() { Task = new TaskDetails() { TaskName =
"Child task 1", Duration = 400000 }, TaskId = 2, Progress = 80, ParentId =
1, Priority = "Normal" });
TreeData.Add(new BusinessObject() { Task = new TaskDetails() { TaskName =
"Child Task 2", Duration = 500000 }, TaskId = 3, Progress = 65, ParentId =
1, Priority = "Critical" });
TreeData.Add(new BusinessObject() { Task = new TaskDetails() { TaskName =
"Parent Task 2", Duration = 50000 }, TaskId = 4, Progress = 70, ParentId =
null, Priority = "High" });
TreeData.Add(new BusinessObject() { Task = new TaskDetails() { TaskName =
"Child task 1", Duration = 400000 }, TaskId = 5, Progress = 80, ParentId =
4, Priority = "Normal" });
TreeData.Add(new BusinessObject() { Task = new TaskDetails() { TaskName =
"Child Task 2", Duration = 500000 }, TaskId = 6, Progress = 65, ParentId =
4, Priority = "Critical" });
}
}

```

Task ID	Task Name	Duration	Progress	Priority
1	▼ Parent Task 1	50000	70	High
2	Child task 1	400000	80	Normal
3	Child Task 2	500000	65	Critical
4	▼ Parent Task 2	50000	70	High
5	Child task 1	400000	80	Normal
6	Child Task 2	500000	65	Critical

<<

<

1

>

>>

1 of 1 pages (2 items)

Expando data binding

Tree Grid supports Complex Data Binding with ExpandoObject. In the below examples **Task.TaskName** and **Task.Duration** are complex data with ExpandoObject.

ASPX-CS

```
@using Syncfusion.Blazor.TreeGrid;
@using System.Dynamic;
<SfTreeGrid DataSource="@TreeData" AllowPaging="true" IdMapping="TaskID"
ParentIdMapping="ParentID" TreeColumnIndex="1">
<TreeGridColumn>
<TreeGridColumn Field="TaskID" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Task.TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Task.Duration" HeaderText="Duration" Width="160"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code {
public List<ExpandoObject> TreeData { get; set; }
protected override void OnInitialized()
{
this.TreeData = GetData().ToList();
}
public static List<ExpandoObject> Data = new List<ExpandoObject>();
public static int ParentRecordID { get; set; }
public static int ChildRecordID { get; set; }
public static List<ExpandoObject> GetData()
{
Data.Clear();
ParentRecordID = 0;
ChildRecordID = 0;
for (var i = 1; i <= 2; i++)
{
```

```

dynamic ParentRecord = new ExpandoObject();
dynamic Task = new ExpandoObject();
Task.TaskName = "Parent Task " + i;
Task.Duration = ParentRecordID % 2 == 0 ? 32 : 76;
ParentRecord.TaskID = ++ParentRecordID;
ParentRecord.Task = Task;
ParentRecord.Progress = ParentRecordID % 2 == 0 ? "In Progress" : "Open";
ParentRecord.Priority = ParentRecordID % 2 == 0 ? "High" : "Low";
ParentRecord.ParentID = null;
Data.Add(ParentRecord);
AddChildRecords(ParentRecordID);
}
return Data;
}
public static void AddChildRecords(int ParentId)
{
    for (var i = 1; i < 3; i++)
    {
        dynamic ChildRecord = new ExpandoObject();
        dynamic Task = new ExpandoObject();
        Task.TaskName = "Child Task " + ++ChildRecordID;
        Task.Duration = ParentRecordID % 3 == 0 ? 64 : 98;
        ChildRecord.TaskID = ++ParentRecordID;
        ChildRecord.Task = Task;
        ChildRecord.Progress = ParentRecordID % 3 == 0 ? "Validated" : "Closed";
        ChildRecord.Priority = ParentRecordID % 3 == 0 ? "Low" : "Critical";
        ChildRecord.ParentID = ParentId;
        Data.Add(ChildRecord);
    }
}
}

```

Task ID	Task Name	Duration	Progress	Priority
1	▼ Parent Task 1	32	Open	Low
2	Child Task 1	98	Closed	Critical
3	Child Task 2	98	Validated	Low
4	▼ Parent Task 2	76	In Progress	High
5	Child Task 3	98	Closed	Critical
6	Child Task 4	98	Validated	Low

<<
 <
 1
 >
 >>

1 of 1 pages (2 items)

Header template

To know about **Header Template** in Blazor tree grid Component, you can check this video.

```
{% youtube
```

```
"youtube:https://www.youtube.com/watch?v=PnM11O-BPVU"%}
```

The header element can be customized by using the [HeaderTemplate](#) property.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid Height="400" DataSource="@TreeGridData" IdMapping="EmployeeID"
ParentIdMapping="ParentId" TreeColumnIndex="0">
<TreeGridColumns>
<TreeGridColumn Field="Name" HeaderText="Name" Width="160">
<HeaderTemplate>
<div>
<span class="e-icon-userlogin e-icons employee"></span> Name
</div>
</HeaderTemplate>
</TreeGridColumn>
<TreeGridColumn Field="Designation" HeaderText="Designation"
Width="120"></TreeGridColumn>
<TreeGridColumn Field="EmpID" HeaderText="Progress" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Country" HeaderText="Priority"
Width="100"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
<style>
@@font-face {
font-family: 'ej2-headertemplate';
src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj1vTFIAAAEoAAAAVmNtYXDS2c5qAAABjAAAAEBnbHlmQmN
FrQAAAdQAAANoaGVhZBRdbkIAAADQAAAAANmhoZWEIUQQEAAAArAAAACRobXR4DAAAAAAAAAYAAAAA
MbG9jYQCOAbQAAAHMAAACG1heHABHgENAAABCAAAACBuYW1lohGZJQAABTWAAAKpcG9zdA2o3w0
AAAFoAAAAQAABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAAwABAAAAQAATBXY9l8
PPPUACwQAAAAAANillKkAAAAA2KWUqQAAAAAD9APzAAAACAACAAAAAAAAAAAAEAAAADAQEAEQAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAABQAAAAoKcZAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLpYAQAAAAAXAQAAAAAAAAABAAAAAAAAABAAAAQA AAAA
EAAAAAAAAAgAAAAAMAAAAUAMAAQAAABQABAAsAAAAABgAEAAEAucC6WD//wAA5wLpYP//AAAAAA
BAAYABgAAAAIAAQAAAAAAjgG0ABEAAAAAA8kD8wADAAACwAPABMAFWAbAB8AIwAnACsALwAzADc
AOwBPAGsAACUVIzUjFSM1IxUjNSMVIzUjFSM1JRujNSMVIzUjFSM1IxUjNSMVIzU1FSM1IxUjNSM
VIzUjFSM1IxUjNQMVHwYhPwCRITcjDwghNS8HIzUjFSE1IwN2U1NTU1RTU1NTAUxTU1NTVFNTU1M
C7FNTU1NUU1NTU1QCAwUGBgIA0QICACHBQQBAVxsp30ICACHAgUDAQED1AECBAUHBwgIfVP+YFO
zU1NTU1NTU1NTU6dUVFVRUVFVRUplNTU1NTU1NTU1P+NgQIBwcGBAMCAQIEBQcHAWgCdPoBAGQ
FAwCHCIF8CQgHBgUEAgFTU1MABAAAAAAD9APeADQAbQCuAQAAAAEfCDc1Lwc1PwIPBy8HHww3HwQ
PATMVPwclLx0jDwMfAgUdAR8GBTUzLxQjDx0BFxUFEDsBPxA1Nyc1LxIPEhUCCg8OGxcVExANCgM
BAQMDCCQDAGECAXESEhMTEhUUFRTFBISEhEHCwYHCAkKCw0NDw8SuQQAwIBAgRxlgsKCQcGBAM
BAGMDAwUFBQcGBwgICQkKCgsKDasMDQwNDQ4NDg45BQUDAQEAA/1eAgUGBwkKCwHjeggKDhEUFxs
1FRMSEA4NCwoJBwcJBjwODg0ODQ0MDQwLDAoLCgoJCQgIBwYHBQUFAwMDAGEBQECAGYICg0ODxI
SFBUXFwwMDA0MDQwMFxcVFBISDw4MCwgGAgIBAQICAwJCw0PERITFRUXDAwMDA0NDAMMDAsXFRQ
TEQ8ODQoIBgICAVQEAWgJCgsMCwwbEBAREREZE8VDAwKCgoIBwYFAwIBAQIDBQYHCAoUFQwLCws
LCgoJCACGMwsWFhUVHB3QAQIEBgICgueDg4ODg0NDA0MCwsLCwoKCQgJBwgGBwUFBAQDAwECCw8
NDxETCrIlawsKCAgGBAIB0AoLCwoLCQgNCAkLDA0NDg4ODg4ZFAlBAwMEBAUGBgYIBwkICQoKCws
LDAwMDA0ODQ4ODgH7DQwMDBgWFRQTERAODAoIBgICAQECAGYICgwoEBETFBWUgAwMDA0MDQwMCxc
WFRMSEA8NDakHawIBAQEBQECAGYICgwoEBETFBWUgAwMDQAAAAASAN4AAQAAAAAAAAABAAAAQA
AAAAAAQASAAEAQAAAAAAAgAHABMAAQAAAAAAAwASABoAAQAAAAAABAASACwAAQAAAAAABQALAD4
AAQAAAAAABgASAEkAAQAAAAAACgAsAFsAAQAAAAAACwASAIcAAwABBAkAAAAACAjKAAwABBAkAAQA
```

```

kAJJsAAwABBAkAAgAOAL8AAwABBAkAAwAkAM0AAwABBAkABAakAPEAAwABBAkABQAWARUAAwABBAk
ABgAkASSAAwABBAkACgBYAU8AAwABBAkACwAkAacgZWoyLWhlYWRLcnRlbXBsYXRlUmVndWxhcmV
qMiloZWFKZXJ0ZW1wbGF0ZWVqMiloZWFKZXJ0ZW1wbGF0ZVZlcnNpb24gMS4wZWoyLWhlYWRLcnR
lbXBsYXRlRm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z1c2lubiBNZXRYbyBTdHVkaW93d3cuc3l
uY2Z1c2lubi5jb20AIABlAGoAMgAtAGgAZQBhAGQAZQByAHQAZQBtAHAAbABhAHQAZQBSAGUAZwB
lAGWAYQByAGUAagAyAC0AaABlAGEAZABlAHlAdABlAG0AcABsAGEAdABlAGUAagAyAC0AaABlAGE
AZABlAHlAdABlAG0AcABsAGEAdABlAFYAZQByAHMAaQBvAG4AIAAxAC4AMABlAGoAMgAtAGgAZQB
hAGQAZQByAHQAZQBtAHAAbABhAHQAZQBGAG8AbgB0ACAAZwBlAG4AZQByAGEAdABlAGQAIABlAHM
AaQBUAGcAIABTahkAbgBjAGYAdQBzAGkAbwBuACAATQBlAHQAcgBvACAAUwB0AHUAZABpAG8AdwB
3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAaGAAAAAAAKAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAADAQIBAwEEAAtCVF9DYWxlbmRhcghlbXBsb3llZQAQ)
format('truetype');
font-weight: normal;
font-style: normal;
}
.e-grid .e-icon-userlogin::before {
font-family: 'ej2-headertemplate';
width: 15px !important;
content: '\e702';
}
.e-grid .e-icon-calender::before {
font-family: 'ej2-headertemplate';
width: 15px !important;
content: '\e960';
}
</style>
@code{
public Employee model = new Employee();
public IEnumerable<Employee> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = Employee.GetTemplateData();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class Employee
{
public string Name { get; set; }
public string Designation { get; set; }
public string EmpID { get; set; }
public string Contact { get; set; }
public int? ParentId { get; set; }
public TreeData Treedata { get; set; }
public static List<Employee> GetTemplateData()
{
List<Employee> DataCollection = new List<Employee>();
DataCollection.Add(new Employee { Name = "Robert King",Designation = "Chief
Executive Officer",EmpID = "EMP001",Country = "USA",ParentId = null,Treedata
= new TreeData() { ID = 21}});
DataCollection.Add(new Employee { Name = "David william",Designation = "Vice
President",EmpID = "EMP004",Country = "USA",ParentId = 1,Treedata = new
TreeData() { ID = 21 }});
}
}


```



```

DataCollection.Add(new Employee { Name = "Nancy Davolio", Designation =
"Marketing Executive", EmpID = "EMP035", Country = "USA", ParentId = 1, Treedata
= new TreeData() { ID = 21 } });
DataCollection.Add(new Employee { Name = "Andrew Fuller", Designation =
"Sales Representative", EmpID = "EMP045", Country = "UK", ParentId = 1, Treedata
= new TreeData() { ID = 21 } });
DataCollection.Add(new Employee { Name = "Anne Dodsworth", FullName =
"AnneDodsworth", Designation = "Sales Representative", EmployeeID = 5, EmpID =
"EMP091", Country = "USA", ParentId = null, Treedata = new TreeData() { ID = 21
} });
DataCollection.Add(new Employee { Name = "Michael Suyama", FullName =
"MichaelSuyama", Designation = "Sales Representative", EmployeeID = 6, EmpID =
"EMP110", Country = "UK", ParentId = 5, Treedata = new TreeData() { ID = 21
} });
return DataCollection;
}
}
}

```

 Name	Designation	Progress	Priority
▼ Robert King	Chief Executive ...	EMP001	USA
David william	Vice President	EMP004	USA
Nancy Davolio	Marketing Execu...	EMP035	USA
Andrew Fuller	Sales Represent...	EMP045	UK
▼ Anne Dodsworth	Sales Represent...	EMP091	USA
Michael Suyama	Sales Represent...	EMP110	UK
Janet Leverling	Sales Coordinator	EMP131	UK

For Templated Tree Grid component, [ModelType](#) property of Tree Grid should be defined.

Header text

By default, column header title is displayed from column [Field](#) value. To override the default header title, define the [HeaderText](#) value.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>

```

```

<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
}
public static List<BusinessObject> GetSelfDataSource()
{
List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", Duration = 6, Progress = 80, Priority = "Low", ParentId = 1 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5});
}
}

```

```
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5});
return BusinessObjectCollection;
}
}
}
```

Task ID	Task Name	Duration	Progress	Priority
1	▼ Parent Task 1	10	70	Critical
2	▼ Child task 1	4	80	Low
3	▼ Child Task 2	5	65	Critical
4	Child task 3	6	77	High
5	▼ Parent Task 2	10	70	Critical
6	Child task 1	4	80	Critical
7	Child Task 2	5	65	Low
8	Child task 3	6	77	High
9	Child task 4	6	77	Low

If both the [Field](#) and [HeaderText](#) are not defined in the column, the column renders with **empty** header text.

Format

To format cell values based on specific culture, use the [Format](#) property of the [TreeGridColumn](#) tag helper. The Tree Grid uses **Internalization** library to format the number values.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="OrderID"
ParentIdMapping="ParentId" TreeColumnIndex="1" >
<TreeGridColumn>
<TreeGridColumn Field="OrderID" HeaderText="Order ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="OrderName" HeaderText="Order Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Price" HeaderText="Price" Width="100" Format="C2"
Type="Syncfusion.Blazor.Grids.ColumnType.Number"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
</TreeGridColumn>
```

```

</SfTreeGrid>
@code{
public List<TreeDataFormat> TreeGridData { get; set; }
protected override void OnInitialized()
{
    this.TreeGridData = TreeDataFormat.GetDataFormat().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeDataFormat
{
    public TreeDataFormat() { }
    public int OrderID { get; set; }
    public string OrderName { get; set; }
    public double Price { get; set; }
    public int? ParentId { get; set; }
    public static List<TreeDataFormat> GetDataFormat()
    {
        List<TreeDataFormat> data = new List<TreeDataFormat>()
        {
            new TreeDataFormat() { OrderID= 1,OrderName= "Order 1",ParentId =
            null,Price=133.66 },
            new TreeDataFormat() { OrderID= 11, ParentId = 1, OrderName= "Mackerel",
            Price= 28.20},
            new TreeDataFormat() { OrderID= 12, ParentId = 1, OrderName= "Mackerel",
            Price= 25.92},
            new TreeDataFormat() { OrderID= 13, ParentId = 1, OrderName= "Mackerel",
            Price= 52.68},
            new TreeDataFormat() { OrderID= 14, ParentId = 1, OrderName= "Mackerel",
            Price= 11.25},
            new TreeDataFormat() { OrderID= 15, ParentId = 1, OrderName= "Mackerel",
            Price= 15.61},
            new TreeDataFormat() { OrderID= 2,OrderName= "Order 2",ParentId =
            null,Price=212.33},
            new TreeDataFormat() { OrderID= 21, ParentId = 2, OrderName=
            "Tilapias",Price= 55.50},
            new TreeDataFormat() { OrderID= 22, ParentId = 2, OrderName= "White Shrimp",
            Price= 41.70},
            new TreeDataFormat() { OrderID= 23, ParentId = 2, OrderName= "Fresh
            Cheese",Price= 39.20},
            new TreeDataFormat() { OrderID= 24, ParentId = 2, OrderName= "Blue Veined
            Cheese",Price= 38.76},
            new TreeDataFormat() { OrderID= 25, ParentId = 2, OrderName= "Butter",
            Price= 37.17}};
        return data;
    }
}
}

```

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 100. 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.

<!-- Please refer to the link to know more about [Number formatting](#). -->

Date formatting

The date values can be formatted using built-in date format string.

For built-in date format specify the [Format](#) property as string (Example: d).

<!-- Please refer to the link to know more about [Date formatting](#). -->

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid IdMapping="OrderID" ParentIdMapping="ParentId"
DataSource="@TreeGridData" TreeColumnIndex="1" >
<TreeGridColumn>
<TreeGridColumn Field="OrderID" HeaderText="Order ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="OrderName" HeaderText="Order Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="OrderDate" HeaderText="Order Date" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date" Width="160"></TreeGridColumn>
<TreeGridColumn Field="Price" HeaderText="Price" Width="100" Format="C2"
Type="Syncfusion.Blazor.Grids.ColumnType.Number"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
public List<TreeDataFormat> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeDataFormat.GetDataFormat().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeDataFormat
{
public TreeDataFormat() { }
public int OrderID { get; set; }
public string OrderName { get; set; }
}
}

```

```
public double Price { get; set; }
public DateTime? OrderDate { get; set; }
public int? ParentId { get; set; }
public static List<TreeDataFormat> GetDataFormat()
{
    List<TreeDataFormat> data = new List<TreeDataFormat>()
    {
        new TreeDataFormat() { OrderID= 1, OrderName= "Order 1", OrderDate = new
        DateTime(1963, 2, 15), ParentId = null, Price=133.66 },
        new TreeDataFormat() { OrderID= 11, ParentId = 1, OrderName=
        "Mackerel", OrderDate = new DateTime(1966, 3, 19), Price= 28.20},
        new TreeDataFormat() { OrderID= 12, ParentId = 1, OrderName=
        "Mackerel", OrderDate = new DateTime(1963, 2, 15), Price= 25.92},
        new TreeDataFormat() { OrderID= 13, ParentId = 1, OrderName= "Mackerel",
        OrderDate = new DateTime(1966, 3, 19), Price= 52.68},
        new TreeDataFormat() { OrderID= 14, ParentId = 1, OrderName=
        "Mackerel", OrderDate = new DateTime(1963, 2, 15), Price= 11.25},
        new TreeDataFormat() { OrderID= 15, ParentId = 1, OrderName= "Mackerel",
        OrderDate = new DateTime(1980, 9, 20), Price= 15.61},
        new TreeDataFormat() { OrderID= 2, OrderName= "Order 2", ParentId =
        null, OrderDate = new DateTime(1980, 9, 20), Price=212.33},
        new TreeDataFormat() { OrderID= 21, ParentId = 2, OrderName=
        "Tilapias", OrderDate = new DateTime(1966, 3, 19), Price= 55.50},
        new TreeDataFormat() { OrderID= 22, ParentId = 2, OrderName= "White
        Shrimp", OrderDate = new DateTime(1963, 2, 15), Price= 41.70},
        new TreeDataFormat() { OrderID= 23, ParentId = 2, OrderName= "Fresh
        Cheese", OrderDate = new DateTime(1980, 9, 20), Price= 39.20},
        new TreeDataFormat() { OrderID= 24, ParentId = 2, OrderName= "Blue Veined
        Cheese", OrderDate = new DateTime(1966, 3, 19), Price= 38.76},
        new TreeDataFormat() { OrderID= 25, ParentId = 2, OrderName=
        "Butter", OrderDate = new DateTime(1966, 3, 19), Price= 37.17}};
    return data;
}
}
```

Order ID	Order Name	Order Date	Price
1	▼ Order 1	3/2/2017	\$133.66
11	Mackerel	3/2/2017	\$28.20
12	Mackerel	3/5/2017	\$25.92
13	Mackerel	3/10/2017	\$52.68
14	Mackerel	3/8/2017	\$11.25
15	Mackerel	3/9/2017	\$15.61
2	▼ Order 2	3/5/2017	\$212.33
21	Tilapias	3/5/2017	\$55.50
22	White Shrimp	5/7/2017	\$41.70

AutoFit specific columns

The **AutoFitColumns** method resizes the column to fit the widest cell's content without wrapping. A specific column can be autofitted at initial rendering by invoking the [AutoFitColumns](#) method in [DataBound](#) event.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridEvents DataBound="OnDataBound"
TValue="TreeData.BusinessObject"></TreeGridEvents>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid<TreeData.BusinessObject> TreeGrid;
public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
```

```
{
    this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
private void OnDataBound(object e)
{
    this.TreeGrid.AutoFitColumns(new List<string>() { "TaskName" });
}
}
```

C#

```
namespace TreeGridComponent.Data {
    public class TreeData
    {
        public class BusinessObject
        {
            public int TaskId { get; set; }
            public string TaskName { get; set; }
            public int? Duration { get; set; }
            public int? Progress { get; set; }
            public string Priority { get; set; }
            public int? ParentId { get; set; }
        }
        public static List<BusinessObject> GetSelfDataSource()
        {
            List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
                "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
                null });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
                "Child task 1", Duration = 4, Progress = 80, Priority = "Low", ParentId = 1 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
                "Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
            });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
                "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
                "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
                null });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
                "Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
                5 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
                "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
                "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
                "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
            return BusinessObjectCollection;
        }
    }
}
```


Task ID	Task Name	Duration	Progress	Priority
1	▼ Parent Task 1	10	70	Critical
2	▼ Child task 1	4	80	Low
3	▼ Child Task 2	5	65	Critical
4	Child task 3	6	77	High
5	▼ Parent Task 2	10	70	Critical
6	Child task 1	4	80	Critical
7	Child Task 2	5	65	Low
8	Child task 3	6	77	High
9	Child task 4	6	77	Low

All the columns can be autofitted by invoking the **AutoFitColumns** method without column names.

Reorder

Reordering can be done by drag and drop of a particular column header from one index to another index within the treegrid. To enable reordering, set the [AllowReordering](#) to true.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid IdMapping="TaskId" ParentIdMapping="ParentId"
DataSource="@TreeGridData" TreeColumnIndex="1" AllowReordering="true">
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="90"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText=" Start Date"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date" Width="90"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="80"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
```

```
}
}
```

C#

```
namespace TreeGridComponent.Data {
public class TreeData
{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public DateTime? StartDate { get; set; }
public int? Duration { get; set; }
public int? ParentId { get; set; }
}
public static List<BusinessObject> GetSelfDataSource()
{
List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", StartDate = new DateTime(2017, 10, 23), Duration =
10, ParentId = null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", StartDate = new DateTime(2017, 10, 23), Duration = 6, ParentId
= 1 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", StartDate = new DateTime(2017, 10, 24), Duration = 5, ParentId
= 2 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", StartDate = new DateTime(2017, 10, 25), Duration = 6, ParentId
= 3 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Parent Task 2", StartDate = new DateTime(2017, 10, 23), Duration =
10, ParentId = null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child task 1", StartDate = new DateTime(2017, 10, 23), Duration = 4, ParentId
= 5 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Child Task 2", StartDate = new DateTime(2017, 10, 24), Duration = 5, ParentId
= 5 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child task 3", StartDate = new DateTime(2017, 10, 25), Duration = 6, ParentId
= 5 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child task 4", StartDate = new DateTime(2017, 10, 25), Duration = 6, ParentId
= 5 });
return BusinessObjectCollection;
}
}
}
```

Reordering a particular column can be disabled by setting the [AllowReordering](#) property of [TreeGridColumns](#) tag helper to false.

Reorder multiple columns

Multiple columns can be reordered at a time by using the **ReorderColumns** method.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<button id="reorderMultipleCols" @onclick="click">Reorder</button>
<SfTreeGrid @ref="TreeGrid" IdMapping="TaskId" ParentIdMapping="ParentId"
DataSource="@TreeGridData" TreeColumnIndex="1" AllowReordering="true">
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="90"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText=" Start Date"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date" Width="90"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="80"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
SfTreeGrid<TreeData.BusinessObject> TreeGrid;
public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
private void click()
{
this.TreeGrid.ReorderColumns(new List<string>() { "TaskId", "Duration" },
"Progress");
}
}
```

C#

```
namespace TreeGridComponent.Data {
public class TreeData
{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public DateTime? StartDate { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public int? ParentId { get; set; }
}
public static List<BusinessObject> GetSelfDataSource()
{
}
```

```

List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", StartDate = new DateTime(2017, 10, 23), Duration =
10, Progress = 70, ParentId = null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", StartDate = new DateTime(2017, 10, 23), Duration = 6, Progress
= 80, ParentId = 1 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", StartDate = new DateTime(2017, 10, 24), Duration = 5, Progress
= 65, ParentId = 2 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", StartDate = new DateTime(2017, 10, 25), Duration = 6, Progress
= 77, ParentId = 3 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Parent Task 2", StartDate = new DateTime(2017, 10, 23), Duration =
10, Progress = 70, ParentId = null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child task 1", StartDate = new DateTime(2017, 10, 23), Duration = 4, Progress
= 80, ParentId = 5 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Child Task 2", StartDate = new DateTime(2017, 10, 24), Duration = 5, Progress
= 65, ParentId = 5 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child task 3", StartDate = new DateTime(2017, 10, 25), Duration = 6, Progress
= 77, ParentId = 5 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child task 4", StartDate = new DateTime(2017, 10, 25), Duration = 6, Progress
= 77, ParentId = 5 });
return BusinessObjectCollection;
}
}
}

```

Lock columns

Columns can be locked by using the [LockColumn](#) property. The locked columns will be moved to the first position. Also this position can't be reordered.

In the below example, Duration column is locked and its reordering functionality is disabled.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid IdMapping="TaskId" ParentIdMapping="ParentId"
DataSource="@TreeGridData" TreeColumnIndex="1" AllowReordering="true">
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="90"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText="Start Date"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date" Width="90"></TreeGridColumn>
<TreeGridColumn Field="Duration" LockColumn="true" HeaderText="Duration"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="80"></TreeGridColumn>

```

```

<TreeGridColumn Field="Progress" HeaderText="Progress"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
    this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
    public class BusinessObject
    {
        public int TaskId { get; set; }
        public string TaskName { get; set; }
        public DateTime? StartDate { get; set; }
        public int? Duration { get; set; }
        public int? Progress { get; set; }
        public int? ParentId { get; set; }
    }
    public static List<BusinessObject> GetSelfDataSource()
    {
        List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
            "Parent Task 1", StartDate = new DateTime(2017, 10, 23), Duration =
            10, Progress = 70, ParentId = null });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
            "Child task 1", StartDate = new DateTime(2017, 10, 23), Duration = 4, Progress
            = 80, ParentId = 1 });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
            "Child Task 2", StartDate = new DateTime(2017, 10, 24), Duration = 5, Progress
            = 65, ParentId = 2 });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
            "Child task 3", StartDate = new DateTime(2017, 10, 25), Duration = 6, Progress
            = 77, ParentId = 3 });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
            "Parent Task 2", StartDate = new DateTime(2017, 10, 23), Duration =
            10, Progress = 70, ParentId = null });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
            "Child task 1", StartDate = new DateTime(2017, 10, 23), Duration = 4, Progress
            = 80, ParentId = 5 });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
            "Child Task 2", StartDate = new DateTime(2017, 10, 24), Duration = 5, Progress
            = 65, ParentId = 5 });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
            "Child task 3", StartDate = new DateTime(2017, 10, 25), Duration = 6, Progress
            = 77, ParentId = 5 });
    }
}

```

```

BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child task 4", StartDate = new DateTime(2017, 10, 25), Duration = 6, Progress
= 77, ParentId = 5});
return BusinessObjectCollection;
}
}
}

```

Duration	Task ID	Task Name	Start Date	Progress
10	▼ 1	Parent Task 1	10/23/2017	70
4	▼ 2	Child task 1	10/23/2017	80
5	▼ 3	Child Task 2	10/24/2017	65
6	4	Child task 3	10/25/2017	77
10	▼ 5	Parent Task 2	10/23/2017	70
4	6	Child task 1	10/23/2017	80
5	7	Child Task 2	10/24/2017	65
6	8	Child task 3	10/25/2017	77
6	9	Child task 4	10/25/2017	77

Column resizing

Column width can be resized by clicking and dragging the right edge of the column header. While dragging, the width of the respective column will be resized immediately. Each column can be auto resized by double-clicking the right edge of the column header to fit the width of that column based on the widest cell content. To enable column resize, set the [AllowResizing](#) property to true.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid IdMapping="TaskId" ParentIdMapping="ParentId"
DataSource="@TreeGridData" TreeColumnIndex="1" AllowResizing="true">
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="90"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText=" Start Date"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date" Width="90"></TreeGridColumn>

```

```

<TreeGridColumn Field="Duration" LockColumn="true" HeaderText="Duration"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public DateTime? StartDate { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public int? ParentId { get; set; }
}
public static List<BusinessObject> GetSelfDataSource()
{
List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", StartDate = new DateTime(2017, 10, 23), Duration =
10, Progress = 70, ParentId = null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", StartDate = new DateTime(2017, 10, 23), Duration = 4, Progress
= 80, ParentId = 1 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", StartDate = new DateTime(2017, 10, 24), Duration = 5, Progress
= 65, ParentId = 2 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", StartDate = new DateTime(2017, 10, 25), Duration = 6, Progress
= 77, ParentId = 3 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Parent Task 2", StartDate = new DateTime(2017, 10, 23), Duration =
10, Progress = 70, ParentId = null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child task 1", StartDate = new DateTime(2017, 10, 23), Duration = 4, Progress
= 80, ParentId = 5 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Child Task 2", StartDate = new DateTime(2017, 10, 24), Duration = 5, Progress
= 65, ParentId = 5 });
}
}

```

```

BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child task 3", StartDate = new DateTime(2017, 10, 25), Duration = 6, Progress
= 77, ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child task 4", StartDate = new DateTime(2017, 10, 25), Duration = 6, Progress
= 77, ParentId = 5});
return BusinessObjectCollection;
}
}
}

```

* Resizing for a particular column can be disabled by setting the [AllowResizing](#) property of the [TreeGridColumnns](#) tag helper to false.

* In RTL mode, click and drag the left edge of the header cell to resize the column.

Min and max width

Column resize can be restricted between minimum and maximum width by defining the [MinWidth](#) and [MaxWidth](#) properties of the [TreeGridColumnns](#) tag helper.

In the following sample, minimum and maximum width are defined for **Duration**, and **Task Name** columns.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid IdMapping="TaskId" ParentIdMapping="ParentId"
DataSource="@TreeGridData" TreeColumnIndex="1" AllowResizing="true">
<TreeGridColumnns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name" MinWidth="170"
MaxWidth="250" Width="180"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" MinWidth="50"
MaxWidth="150" TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Priority " HeaderText="Priority "
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="80"></TreeGridColumn>
</TreeGridColumnns>
</SfTreeGrid>
@code{
public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {

```



```

public class TreeData
{
    public class BusinessObject
    {
        public int TaskId { get; set; }
        public string TaskName { get; set; }
        public int? Duration { get; set; }
        public int? Progress { get; set; }
        public int? ParentId { get; set; }
        public string Priority { get; set; }
    }
    public static List<BusinessObject> GetSelfDataSource()
    {
        List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
        "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
        null });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
        "Child task 1", Duration = 6, Progress = 80, Priority = "Low", ParentId = 1 });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
        "Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
        });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
        "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
        "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
        null });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
        "Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
        5 });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
        "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
        "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
        "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
        return BusinessObjectCollection;
    }
}

```

Resize stacked column

Stacked columns can be resized by clicking and dragging the right edge of the stacked column header. While dragging, the width of the respective child columns will be resized at the same time. Resize for a particular stacked column can be disabled by setting the [AllowResizing](#) as **false** to its columns.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid IdMapping="ID" ParentIdMapping="ParentID"
DataSource="@TreeGridData" TreeColumnIndex="1" AllowResizing="true">
<TreeGridColumn>
<TreeGridColumn HeaderText="Order Details"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right">
</TreeGridColumn>

```

```

<TreeGridColumn Field="ID" Width="110" HeaderText="Order ID"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Name" Width="220" HeaderText="Order Name"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Left"></TreeGridColumn>
<TreeGridColumn Field="OrderDate" Width="120" HeaderText="Order Date"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date"></TreeGridColumn>
</TreeGridColumns>
</TreeGridColumn>
<TreeGridColumn HeaderText="Shipment Details"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center">
<TreeGridColumns>
<TreeGridColumn Field="ShipmentCategory" Width="170" HeaderText="Shipment
Category"></TreeGridColumn>
<TreeGridColumn Field="Units" Width="220" HeaderText="Units"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Left"></TreeGridColumn>
<TreeGridColumn Field="ShippedDate" Width="120" HeaderText="Shipment Date"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date"></TreeGridColumn>
</TreeGridColumns>
</TreeGridColumn>
<TreeGridColumn HeaderText="Price Details"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center">
<TreeGridColumns>
<TreeGridColumn Field="UnitPrice" Width="180" HeaderText="Price per unit"
Format="C2" Type="Syncfusion.Blazor.Grids.ColumnType.Number"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Price" Width="220" HeaderText="Price" Format="C"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Left"></TreeGridColumn>
<TreeGridColumn Field="OrderDate" Width="120" HeaderText="Total Price"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Type="Syncfusion.Blazor.Grids.ColumnType.Number"></TreeGridColumn>
</TreeGridColumns>
</TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public List<ShipmentData> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = ShipmentData.GetShipmentData().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class ShipmentData
{
public int ID { get; set; }
public string Name { get; set; }
public int Units { get; set; }
public string Category { get; set; }
public int UnitPrice { get; set; }
public int Price { get; set; }
public int? ParentID { get; set; }
}
}

```

```

public string ShipmentCategory { get; set; }
public DateTime ShippedDate { get; set; }
public DateTime OrderDate { get; set; }
public List<ShipmentData> Children { get; set; }
public static List<ShipmentData> GetShipmentData()
{
    List<ShipmentData> DataCollection = new List<ShipmentData>() {
        new ShipmentData() { ID = 1, Name = "Order 1", Category = "Seafood", Units =
            1395, UnitPrice = 47, Price = 65565, ParentID = null, OrderDate = new
            DateTime(2017, 3, 2), ShippedDate = new DateTime(2017, 9, 2), ShipmentCategory
            = "Seafood"},
        new ShipmentData() { ID = 11, Name = "Mackerel", Category = "Frozen
            Seafood", Units = 235, UnitPrice = 12, Price = 2820, ParentID = 1, OrderDate =
            new DateTime(2017, 3, 3), ShippedDate = new DateTime(2017, 10,
            3), ShipmentCategory = "Frozen Seafood"},
        new ShipmentData() { ID = 12, Name = "Yellowfin Tuna", Category = "Frozen
            Seafood", Units = 324, UnitPrice = 8, Price = 2592, ParentID = 1, OrderDate = new
            DateTime(2017, 3, 5), ShippedDate = new DateTime(2017, 10,
            5), ShipmentCategory = "Frozen Seafood"},
        new ShipmentData() { ID = 13, Name = "Herrings", Category = "Frozen
            Seafood", Units = 488, UnitPrice = 11, Price = 5368, ParentID = 1, OrderDate =
            new DateTime(2017, 8, 5), ShippedDate = new DateTime(2017, 5,
            15), ShipmentCategory = "Frozen Seafood"},
        new ShipmentData() { ID = 14, Name = "Preserved Olives", Category =
            "Edible", Units = 125, UnitPrice = 9, Price = 1125, ParentID = 1, OrderDate = new
            DateTime(2017, 6, 10), ShippedDate = new DateTime(2017, 6,
            17), ShipmentCategory = "Edible"},
        new ShipmentData() { ID = 15, Name = " Sweet corn Frozen ", Category =
            "Edible", Units = 223, UnitPrice = 7, Price = 1561, ParentID = 1, OrderDate = new
            DateTime(2017, 7, 12), ShippedDate = new DateTime(2017, 7,
            19), ShipmentCategory = "Edible"},
        new ShipmentData() { ID = 2, Name = "Order 2", Category = "Products", Units =
            1944, UnitPrice = 58, Price = 21233, ParentID = null, OrderDate = new
            DateTime(2017, 1, 10), ShippedDate = new DateTime(2017, 1,
            16), ShipmentCategory = "Seafood", Children = new List<ShipmentData>() },
        new ShipmentData() { ID = 21, Name = "Tilapias", Category = "Frozen
            Seafood", Units = 278, UnitPrice = 15, Price = 4170, ParentID = 2, OrderDate =
            new DateTime(2017, 2, 5), ShippedDate = new DateTime(2017, 2,
            12), ShipmentCategory = "Frozen Seafood"},
        new ShipmentData() { ID = 22, Name = "White Shrimp", Category = "Frozen
            Seafood", Units = 560, UnitPrice = 7, Price = 3920, ParentID = 2, OrderDate = new
            DateTime(2017, 5, 22), ShippedDate = new DateTime(2017, 5,
            29), ShipmentCategory = "Frozen Seafood"},
        new ShipmentData() { ID = 23, Name = "Fresh Cheese", Category = "Dairy", Units
            = 323, UnitPrice = 12, Price = 3876, ParentID = 2, OrderDate = new
            DateTime(2017, 6, 8), ShippedDate = new DateTime(2017, 6,
            15), ShipmentCategory = "Dairy"},
        new ShipmentData() { ID = 24, Name = "Blue Veined Cheese", Category =
            "Dairy", Units = 370, UnitPrice = 15, Price = 5550, ParentID = 2, OrderDate = new
            DateTime(2017, 7, 10), ShippedDate = new DateTime(2017, 7,
            17), ShipmentCategory = "Dairy"},
        new ShipmentData() { ID = 25, Name = "Butter", Category = "Dairy", Units =
            413, UnitPrice = 9, Price = 3717, ParentID = 2, OrderDate = new DateTime(2017,
            9, 18), ShippedDate = new DateTime(2017, 9, 25), ShipmentCategory = "Dairy"}
    };
    return DataCollection;
}

```

```
}
}
```

Touch interaction

When the right edge of the header cell is tapped, a floating handler will be visible over the right border of the column. To resize the column, tap and drag the floating handler as needed.

<!-- markdownlint-disable MD033 -->

<!-- markdownlint-enable MD033 -->

Column template

To know about **Column Template** in Blazor tree grid Component, you can check this video.

{% youtube

"youtube:https://www.youtube.com/watch?v=PnM11O-BPVU"%}

The column [Template](#) has options to display custom element instead of a field value in the column.







C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
@inject Microsoft.AspNetCore.Components.NavigationManager UriHelper
<SfTreeGrid Height="400" DataSource="@TreeGridData" IdMapping="EmployeeID"
ParentIdMapping="ParentId" TreeColumnIndex="0">
  <TreeGridColumn>
    <TreeGridColumn Field="Name" HeaderText="Name" Width="160"></TreeGridColumn>
    <TreeGridColumn HeaderText="Employee Image" Width="80">
      <Template>
        @{
          var employee = (context as Employee);
          <div class="image">
            
          </div>
        }
      </Template>
    </TreeGridColumn>
    <TreeGridColumn Field="DOB" HeaderText="DOB" Width="10"
      Type="Syncfusion.Blazor.Grids.ColumnType.Date"
      Format="yMd"></TreeGridColumn>
    <TreeGridColumn Field="Designation" HeaderText="Designation"
      Width="120"></TreeGridColumn>
    <TreeGridColumn Field="EmployeeID" HeaderText="Employee ID" Width="80"
      TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
    <TreeGridColumn Field="Country" HeaderText="Country"
      Width="100"></TreeGridColumn>
  </TreeGridColumn>
</SfTreeGrid>
@code{
  public Employee model = new Employee();
  public IEnumerable<Employee> TreeGridData { get; set; }
  protected override void OnInitialized()
```

```
{
    this.TreeGridData = Employee.GetTemplateData();
}
}
```

C#

```
namespace TreeGridComponent.Data {
    public class Employee
    {
        public string Name { get; set; }
        public string FullName { get; set; }
        public DateTime? DOB { get; set; }
        public string Designation { get; set; }
        public string EmpID { get; set; }
        public int? EmployeeID { get; set; }
        public string Country { get; set; }
        public string Address { get; set; }
        public string Contact { get; set; }
        public int? ParentId { get; set; }
        public TreeData Treedata { get; set; }
        public static List<Employee> GetTemplateData()
        {
            List<Employee> DataCollection = new List<Employee>();
            DataCollection.Add(new Employee { Name = "Robert King", FullName =
            "RobertKing", Designation = "Chief Executive Officer", EmployeeID = 1, EmpID =
            "EMP001", Address = "507 - 20th Ave. E.Apt. 2A, Seattle", Contact = "(206) 555-9857", Country = "USA", DOB = new DateTime(1963, 2, 15), ParentId =
            null, Treedata = new TreeData() { ID = 21 } });
            DataCollection.Add(new Employee { Name = "David william", FullName =
            "DavidWilliam", Designation = "Vice President", EmployeeID = 2, EmpID =
            "EMP004", Address = "722 Moss Bay Blvd., Kirkland", Contact = "(206) 555-3412", Country = "USA", DOB = new DateTime(1971, 5, 20), ParentId = 1, Treedata
            = new TreeData() { ID = 21 } });
            DataCollection.Add(new Employee { Name = "Nancy Davolio", FullName =
            "NancyDavolio", Designation = "Marketing Executive", EmployeeID = 3, EmpID =
            "EMP035", Address = "4110 Old Redmond Rd., Redmond", Contact = "(206) 555-8122", Country = "USA", DOB = new DateTime(1966, 3, 19), ParentId = 1, Treedata
            = new TreeData() { ID = 21 } });
            DataCollection.Add(new Employee { Name = "Andrew Fuller", FullName =
            "AndrewFuller", Designation = "Sales Representative", EmployeeID = 4, EmpID =
            "EMP045", Country = "UK", DOB = new DateTime(1980, 9, 20), ParentId =
            1, Treedata = new TreeData() { ID = 21 } });
            DataCollection.Add(new Employee { Name = "Anne Dodsworth", FullName =
            "AnneDodsworth", Designation = "Sales Representative", EmployeeID = 5, EmpID =
            "EMP091", Country = "USA", ParentId = null, Treedata = new TreeData() { ID = 21
            } });
            DataCollection.Add(new Employee { Name = "Michael Suyama", FullName =
            "MichaelSuyama", Designation = "Sales Representative", EmployeeID = 6, EmpID =
            "EMP110", Country = "UK", ParentId = 5, Treedata = new TreeData() { ID = 21
            } });
            return DataCollection;
        }
    }
}
```

Name	Employee Image	Designation	Employee ID	Country
▼ Robert King		Chief Executive Officer	1	USA
David William		Vice President	2	USA
Nancy Davolio		Marketing Executive	3	USA
Andrew Fuller		Sales Representative	4	UK
▼ Anne Dodsworth		Sales Representative	5	USA
Michael Suyama		Sales Representative	6	UK

* Tree Grid actions such as editing, filtering and sorting etc. will depend upon the column [Field](#). If the [Field](#) is not specified in the

template column, the tree grid actions cannot be performed.

* For Templated Tree Grid component, [ModelType](#) property of Tree Grid should be defined.

Using condition template

The template elements can be rendered based on condition.

In the following code, checkbox is rendered based on **Duration** field value.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
@using Syncfusion.Blazor.Buttons;
<SfTreeGrid DataSource="@TreeGridData" Id="TreeGrid" AllowPaging="true"
ParentIdMapping="ParentID" IdMapping="TaskID" TreeColumnIndex="2">
<TreeGridColumn>
<TreeGridColumn HeaderText="Template Column" Width="90">
<Template>
@{
var tempColumn= (context as SelfReferenceData);
@if (tempColumn.Duration > 1)
{
<SfCheckBox Checked="true"></SfCheckBox>
}
else
{
<SfCheckBox Checked="false"></SfCheckBox>
}
}
</Template>
```

```

</TreeGridColumn>
<TreeGridColumn Field="TaskID" HeaderText="Task ID" IsPrimaryKey="true"
Width="60"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="120"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText=" Start Date"
Type="Syncfusion.Blazor.Grids.ColumnType.Date" Format="yMd" Width="90"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public SelfReferenceData model = new SelfReferenceData();
public List<SelfReferenceData> TreeGridData { get; set; }
protected override void OnInitialized()
{
    this.TreeGridData = SelfReferenceData.GetTree().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class SelfReferenceData
{
    public static List<SelfReferenceData> tree = new List<SelfReferenceData>();
    [Key]
    public int TaskID { get; set; }
    public string TaskName { get; set; }
    public DateTime StartDate { get; set; }
    public string Progress { get; set; }
    public int Duration { get; set; }
    public int? ParentID { get; set; }
    public SelfReferenceData() { }
    public static List<SelfReferenceData> GetTree()
    {
        if (tree.Count == 0)
        {
            int root = -1;
            for (var t = 1; t <= 60; t++)
            {
                Random ran = new Random();
                string math = (ran.Next() % 3) == 0 ? "High" : (ran.Next() % 2) == 0 ?
                "Release Breaker" : "Critical";
                string progr = (ran.Next() % 3) == 0 ? "Started" : (ran.Next() % 2) == 0 ?
                "Open" : "In Progress";
                root++;
                int rootItem = tree.Count + root + 1;
                tree.Add(new SelfReferenceData() { TaskID = rootItem, TaskName = "Parent
                Task " + rootItem.ToString(), StartDate = new DateTime(1992, 06, 07),
                ParentID = null, Progress = progr, Priority = math, Duration = ran.Next(1,
                50) });
            }
        }
    }
}

```

```

int parent = tree.Count;
for (var c = 0; c < 10; c++)
{
    root++;
    string val = ((parent + c + 1) % 3 == 0) ? "Low" : "Critical";
    int parn = parent + c + 1;
    progr = (ran.Next() % 3) == 0 ? "In Progress" : (ran.Next() % 2) == 0 ?
    "Open" : "Validated";
    int iD = tree.Count + root + 1;
    tree.Add(new SelfReferenceData() { TaskID = iD, TaskName = "Child Task " +
    iD.ToString(), StartDate = new DateTime(1992, 06, 07), ParentID = rootItem,
    Progress = progr, Priority = val, Duration = ran.Next(1, 50) });
    if (((parent + c + 1) % 3) == 0)
    {
        int immParent = tree.Count;
        for (var s = 0; s < 3; s++)
        {
            root++;
            string Prior = (immParent % 2 == 0) ? "Validated" : "Normal";
            tree.Add(new SelfReferenceData() { TaskID = tree.Count + root + 1, TaskName
            = "Sub Task " + (tree.Count + root + 1).ToString(), StartDate = new
            DateTime(1992, 06, 07), ParentID = iD, Progress = (immParent % 2 == 0) ? "On
            Progress" : "Closed", Priority = Prior, Duration = ran.Next(1, 50) });
        }
    }
}
return tree;
}
}
}

```

For Templated Tree Grid component, [ModelType](#) property of Tree Grid should be defined.

Column type

Column type can be specified using the [Type](#) property of [TreeGridColumn](#) tag helper. 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**).

Tree Grid column supports the following types:

- string
- number
- boolean
- date
- datetime

If the [Type](#) is not defined, it will be determined from the first record of the [DataSource](#).

Checkbox column

To render checkboxes in the existing column, set the [ShowCheckbox](#) property of the [TreeGridColumn](#) as **true**.

It is also possible to select the rows hierarchically using checkboxes in the Tree Grid by enabling [AutoCheckHierarchy](#) property. When we check on any parent record checkbox, the child record checkboxes will get checked.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid IdMapping="TaskId" ParentIdMapping="ParentId"
DataSource="@TreeGridData" TreeColumnIndex="1" AutoCheckHierarchy="true">
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name" ShowCheckbox="true"
Width="90"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText=" Start Date"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date" Width="90"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="80"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}
```

C#

```
namespace TreeGridComponent.Data {
public class TreeData
{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public DateTime? StartDate { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public int? ParentId { get; set; }
}
public static List<BusinessObject> GetSelfDataSource()
{
List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
```

```
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =  
"Parent Task 1", StartDate = new DateTime(2017, 10, 23), Duration =  
10, Progress = 70, ParentId = null });  
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =  
"Child task 1", StartDate = new DateTime(2017, 10, 23), Duration = 4, Progress  
= 80, ParentId = 1 });  
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =  
"Child Task 2", StartDate = new DateTime(2017, 10, 24), Duration = 5, Progress  
= 65, ParentId = 2 });  
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =  
"Child task 3", StartDate = new DateTime(2017, 10, 25), Duration = 6, Progress  
= 77, ParentId = 3 });  
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =  
"Parent Task 2", StartDate = new DateTime(2017, 10, 23), Duration =  
10, Progress = 70, ParentId = null });  
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =  
"Child task 1", StartDate = new DateTime(2017, 10, 23), Duration = 4, Progress  
= 80, ParentId = 5});  
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =  
"Child Task 2", StartDate = new DateTime(2017, 10, 24), Duration = 5, Progress  
= 65, ParentId = 5});  
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =  
"Child task 3", StartDate = new DateTime(2017, 10, 25), Duration = 6, Progress  
= 77, ParentId = 5});  
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =  
"Child task 4", StartDate = new DateTime(2017, 10, 25), Duration = 6, Progress  
= 77, ParentId = 5});  
return BusinessObjectCollection;  
}  
}  
}
```

Task ID	<input type="checkbox"/> Task Name	Start Date	Duration	Progress
1	▼ <input type="checkbox"/> Parent Task 1	10/23/2017	10	70
2	▼ <input type="checkbox"/> Child task 1	10/23/2017	4	80
3	▼ <input type="checkbox"/> Child Task 2	10/24/2017	5	65
4	<input type="checkbox"/> Child task 3	10/25/2017	6	77
5	▼ <input type="checkbox"/> Parent Task 2	10/23/2017	10	70
6	<input type="checkbox"/> Child task 1	10/23/2017	4	80
7	<input type="checkbox"/> Child Task 2	10/24/2017	5	65
8	<input type="checkbox"/> Child task 3	10/25/2017	6	77
9	<input type="checkbox"/> Child task 4	10/25/2017	6	77

Responsive columns

The column visibility can be toggled based on the media queries which are defined at the [HideAtMedia](#).

The [HideAtMedia](#) accepts valid [Media Queries](#).

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid IdMapping="TaskId" ParentIdMapping="ParentId"
DataSource="@TreeGridData" TreeColumnIndex="1" AutoCheckHierarchy="true">
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
HideAtMedia="max-width: 700px"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name" ShowCheckbox="true"
Width="90"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText="Start Date"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date" Width="90"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" HideAtMedia="max-
width: 500px" TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="80"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
```

```
}

```

C#

```
namespace TreeGridComponent.Data {
public class TreeData
{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public DateTime? StartDate { get; set; }
public int? Duration { get; set; }
public int? ParentId { get; set; }
}
public static List<BusinessObject> GetSelfDataSource()
{
List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", StartDate = new DateTime(2017, 10, 23), Duration =
10, ParentId = null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", StartDate = new DateTime(2017, 10, 23), Duration = 6, ParentId
= 1 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", StartDate = new DateTime(2017, 10, 24), Duration = 5, ParentId
= 2 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", StartDate = new DateTime(2017, 10, 25), Duration = 6, ParentId
= 3 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Parent Task 2", StartDate = new DateTime(2017, 10, 23), Duration =
10, ParentId = null});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child task 1", StartDate = new DateTime(2017, 10, 23), Duration = 4, ParentId
= 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Child Task 2", StartDate = new DateTime(2017, 10, 24), Duration = 5, ParentId
= 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child task 3", StartDate = new DateTime(2017, 10, 25), Duration = 6, ParentId
= 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child task 4", StartDate = new DateTime(2017, 10, 25), Duration = 6, ParentId
= 5});
return BusinessObjectCollection;
}
}
}
```

<input type="checkbox"/> Task Name	Start Date
▼ <input type="checkbox"/> Parent Task 1	10/23/2017
▼ <input type="checkbox"/> Child task 1	10/23/2017
▼ <input type="checkbox"/> Child Task 2	10/24/2017
<input type="checkbox"/> Child task 3	10/25/2017
▼ <input type="checkbox"/> Parent Task 2	10/23/2017
<input type="checkbox"/> Child task 1	10/23/2017
<input type="checkbox"/> Child Task 2	10/24/2017
<input type="checkbox"/> Child task 3	10/25/2017
<input type="checkbox"/> Child task 4	10/25/2017

Controlling treegrid actions

The tree grid action can be enabled or disabled for a particular column by setting the [AllowFiltering](#), and [AllowSorting](#) properties of [TreeGridColumn](#) tag helper.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid IdMapping="TaskId" ParentIdMapping="ParentId"
DataSource="@TreeGridData" TreeColumnIndex="1" AllowFiltering="true"
AllowSorting="true">
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
AllowSorting="false" AllowFiltering="false"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="90"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText=" Start Date"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date" Width="90"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="80"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
```

```
}
}
```

C#

```
namespace TreeGridComponent.Data {
public class TreeData
{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public DateTime? StartDate { get; set; }
public int? Duration { get; set; }
public int? ParentId { get; set; }
}
public static List<BusinessObject> GetSelfDataSource()
{
List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", StartDate = new DateTime(2017, 10, 23), Duration =
10, ParentId = null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", StartDate = new DateTime(2017, 10, 23), Duration = 6, ParentId
= 1 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", StartDate = new DateTime(2017, 10, 24), Duration = 5, ParentId
= 2 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", StartDate = new DateTime(2017, 10, 25), Duration = 6, ParentId
= 3 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Parent Task 2", StartDate = new DateTime(2017, 10, 23), Duration =
10, ParentId = null});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child task 1", StartDate = new DateTime(2017, 10, 23), Duration = 4, ParentId
= 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Child Task 2", StartDate = new DateTime(2017, 10, 24), Duration = 5, ParentId
= 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child task 3", StartDate = new DateTime(2017, 10, 25), Duration = 6, ParentId
= 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child task 4", StartDate = new DateTime(2017, 10, 25), Duration = 6, ParentId
= 5});
return BusinessObjectCollection;
}
}
}
```

Show or Hide Columns by external button

The tree grid columns can be shown or hidden dynamically using the external buttons by invoking the [ShowColumns](#) or [HideColumns](#) method.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<button id="hide" @onclick="HideColumns">Hide Column</button>
<button id="show" @onclick="ShowColumns">Show Column</button>
<SfTreeGrid @ref="TreeGrid" IdMapping="TaskId" ParentIdMapping="ParentId"
DataSource="@TreeGridData" TreeColumnIndex="1" AllowFiltering="true"
AllowSorting="true">
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
AllowSorting="false" AllowFiltering="false"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="90"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText=" Start Date"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date" Width="90">
</TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="80"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
SfTreeGrid<TreeData.BusinessObject> TreeGrid;
public List<TreeData.BusinessObject> TreeGridData { get; set; }
public string[] ColumnItems = new string[] { "Task ID", "Duration" };
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
private void HideColumns()
{
this.TreeGrid.HideColumns(ColumnItems); //hide by HeaderText
}
private void ShowColumns()
{
this.TreeGrid.ShowColumns(ColumnItems); //show by HeaderText
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public DateTime? StartDate { get; set; }
public int? Duration { get; set; }
public int? ParentId { get; set; }
}
public static List<BusinessObject> GetSelfDataSource()

```

```

{
    List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
    "Parent Task 1", StartDate = new DateTime(2017, 10, 23), Duration =
    10, ParentId = null });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
    "Child task 1", StartDate = new DateTime(2017, 10, 23), Duration = 6, ParentId
    = 1 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
    "Child Task 2", StartDate = new DateTime(2017, 10, 24), Duration = 5, ParentId
    = 2 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
    "Child task 3", StartDate = new DateTime(2017, 10, 25), Duration = 6, ParentId
    = 3 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
    "Parent Task 2", StartDate = new DateTime(2017, 10, 23), Duration =
    10, ParentId = null });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
    "Child task 1", StartDate = new DateTime(2017, 10, 23), Duration = 4, ParentId
    = 5 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
    "Child Task 2", StartDate = new DateTime(2017, 10, 24), Duration = 5, ParentId
    = 5 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
    "Child task 3", StartDate = new DateTime(2017, 10, 25), Duration = 6, ParentId
    = 5 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
    "Child task 4", StartDate = new DateTime(2017, 10, 25), Duration = 6, ParentId
    = 5 });
    return BusinessObjectCollection;
}
}
}

```

How to render boolean values as checkbox

To render boolean values as checkbox in columns, set the [DisplayAsCheckBox](#) property as **true**.

C#

```

@using TreeGridComponent.Data
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid IdMapping="TaskId" ParentIdMapping="ParentId"
DataSource="@TreeGridData" TreeColumnIndex="1">
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Approved" HeaderText="Approved"
DisplayAsCheckBox="true" Width="80"></TreeGridColumn>

```



```

</TreeGridColumn>
</SfTreeGrid>
@code{
    public List<TreeData.BusinessObject> TreeGridData { get; set; }
    protected override void OnInitialized()
    {
        this.TreeGridData = TreeData.GetSelfDataSource().ToList();
    }
}

```

C#

```

namespace TreeGridComponent.Data {
    public class TreeData
    {
        public class BusinessObject
        {
            public int TaskId { get; set; }
            public string TaskName { get; set; }
            public int? Duration { get; set; }
            public int? Progress { get; set; }
            public bool Approved { get; set; }
            public int? ParentId { get; set; }
            public string Priority { get; set; }
        }
        public static List<BusinessObject> GetSelfDataSource()
        {
            List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
                "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
                null, Approved=true });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
                "Child task 1", Duration = 10, Progress = 80, Priority = "Low", ParentId = 1,
                Approved = false });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
                "Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId =
                2, Approved = false });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
                "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 ,
                Approved=true, Approved = false });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
                "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
                null, Approved=true });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
                "Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
                5, Approved = false });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
                "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5,
                Approved=true });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
                "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5,
                Approved = false });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
                "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5,
                Approved=true });
            return BusinessObjectCollection;
        }
    }
}

```

```
}
}
}
```

Task ID	Task Name	Duration	Progress	Approved
1	▼ Parent Task 1	10	70	<input type="checkbox"/>
2	▼ Child task 1	4	80	<input type="checkbox"/>
3	▼ Child Task 2	5	65	<input type="checkbox"/>
4	Child task 3	6	77	<input type="checkbox"/>
5	▼ Parent Task 2	10	70	<input type="checkbox"/>
6	Child task 1	4	80	<input type="checkbox"/>
7	Child Task 2	5	65	<input type="checkbox"/>
8	Child task 3	6	77	<input type="checkbox"/>
9	Child task 4	6	77	<input type="checkbox"/>

Rows in Blazor TreeGrid Component

The row represents record details fetched from the data source.

Customize rows

The appearance of a row can be customized by using the [RowDataBound](#) event. The [RowDataBound](#) event triggers for every row. In the event handler, the **args** is achieved which contains the details of the row.

C#

```
@using TreeGridComponent.Data
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" ParentIdMapping="ParentId"
IdMapping="TaskId" TreeColumnIndex="1">
<TreeGridEvents RowDataBound="OnRowDataBound"
TValue="TreeData"></TreeGridEvents>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="90"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
```

```

<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="100"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
<style>
.equal-5 {
background-color: #336c12;
}
.above-6 {
background-color: #7b2b1d;
}
</style>
@code{
public List<TreeData> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
private void OnRowDataBound(RowDataBoundEventArgs<TreeData> args)
{
if (args.Data.Duration == 5)
{
args.Row.AddClass(new string[] { "equal-5" });
}
else if (args.Data.Duration > 6)
{
args.Row.AddClass(new string[] { "above-6" });
}
}
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public DateTime? StartDate { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<TreeData> GetSelfDataSource()
{
List<TreeData> TreeDataCollection = new List<TreeData>();
TreeDataCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent Task
1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
TreeDataCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child task
1", Duration = 5, Progress = 80, Priority = "Low", ParentId = 1 });
TreeDataCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child Task
2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
TreeDataCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child task
3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
TreeDataCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent Task
2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
}
}
}

```

```

TreeDataCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
return TreeDataCollection;
}
}
}

```

Task ID	Task Name	Duration	Progress	Priority
1	▼ Parent Task 1	10	70	Critical
2	▼ Child task 1	50	80	Low
3	▼ Child Task 2	5	65	Critical
4	Child task 3	6	77	High
5	▼ Parent Task 2	10	70	Critical
6	Child task 1	4	80	Critical
7	Child Task 2	5	65	Low
8	Child task 3	6	77	High
9	Child task 4	6	77	Low

Styling alternate rows

The tree grid's alternative rows' background color can be changed by overriding the **.e-altrow** class.

CSS

```

.e-treegrid .e-altrow {
background-color: #fafafa;
}

```

Please refer to the following example.

C#

```

@using TreeGridComponent.Data
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" ParentIdMapping="ParentId"
IdMapping="TaskId" TreeColumnIndex="1">
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>

```

```

<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}
<style>
.e-treegrid .e-altrow {
background-color: #fafafa;
}
</style>

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public int? ParentId { get; set; }
}
public static List<BusinessObject> GetSelfDataSource()
{
List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", Duration = 10, Progress = 70, ParentId = null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", Duration = 6, Progress = 80, ParentId = 1 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", Duration = 5, Progress = 65, ParentId = 2 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", Duration = 6, Progress = 77, ParentId = 3 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Parent Task 2", Duration = 10, Progress = 70, ParentId = null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child task 1", Duration = 4, Progress = 80, ParentId = 5 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Child Task 2", Duration = 5, Progress = 65, ParentId = 5 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child task 3", Duration = 6, Progress = 77, ParentId = 5 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child task 4", Duration = 6, Progress = 77, ParentId = 5 });
return BusinessObjectCollection;
}
}

```

```

}
}
}

```

Task ID	Task Name	Duration	Progress
1	▼ Parent Task 1	10	70
2	▼ Child task 1	4	80
3	▼ Child Task 2	5	65
4	Child task 3	6	77
5	▼ Parent Task 2	10	70
6	Child task 1	4	80
7	Child Task 2	5	65
8	Child task 3	6	77
9	Child task 4	6	77

Row height

The row height of tree grid rows can be customized through the [RowHeight](#) property. The [RowHeight](#) property is used to change the row height of the entire tree grid rows.

In the below example, the **RowHeight** is set as *60px*.

C#

```

@using TreeGridComponent.Data
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" ParentIdMapping="ParentId"
IdMapping="TaskId" RowHeight="60" TreeColumnIndex="1">
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()

```

```
{
    this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
```

C#

```
namespace TreeGridComponent.Data {
    public class TreeData
    {
        public class BusinessObject
        {
            public int TaskId { get; set; }
            public string TaskName { get; set; }
            public int? Duration { get; set; }
            public int? Progress { get; set; }
            public int? ParentId { get; set; }
        }
        public static List<BusinessObject> GetSelfDataSource()
        {
            List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
                "Parent Task 1", Duration = 10, Progress = 70, ParentId = null });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
                "Child task 1", Duration = 6, Progress = 80, ParentId = 1 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
                "Child Task 2", Duration = 5, Progress = 65, ParentId = 2 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
                "Child task 3", Duration = 6, Progress = 77, ParentId = 3 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
                "Parent Task 2", Duration = 10, Progress = 70, ParentId = null });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
                "Child task 1", Duration = 4, Progress = 80, ParentId = 5 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
                "Child Task 2", Duration = 5, Progress = 65, ParentId = 5 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
                "Child task 3", Duration = 6, Progress = 77, ParentId = 5 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
                "Child task 4", Duration = 6, Progress = 77, ParentId = 5 });
            return BusinessObjectCollection;
        }
    }
}
```

Task ID	Task Name	Duration	Progress
1	▼ Parent Task 1	10	70
2	▼ Child task 1	4	80
3	▼ Child Task 2	5	65
4	Child task 3	6	77
5	▼ Parent Task 2	10	70
6	Child task 1	4	80
7	Child Task 2	5	65

Row template

To know about Row Template in Blazor tree grid Component, you can check this video.

{% youtube

"youtube:https://www.youtube.com/watch?v=cB4NjwFya_U"%}

The **RowTemplate** has an option to customize the look and behavior of the tree grid rows. The [RowTemplate](#) property accepts either the **template** string or HTML elements.

C#

```
@using TreeGridComponent.Data
@using Syncfusion.Blazor.TreeGrid;
@inject Microsoft.AspNetCore.Components.NavigationManager UriHelper
<SfTreeGrid Height="335" DataSource="@TreeData" IdMapping="EmployeeID"
ParentIdMapping="ParentId" TreeColumnIndex="0" RowHeight="83"
GridLines="Syncfusion.Blazor.Grids.GridLine.Vertical">
<TreeGridTemplates>
<RowTemplate>
<td style='padding-left:18px; border-bottom: 0.5px solid #e0e0e0;'>
<RowTemplateTreeColumn>
@{
var cntxt = context as Employee;
<div>@(cntxt.EmpID)</div>
}
</RowTemplateTreeColumn>
</td>
<td style='padding: 10px 0px 0px 20px; border-bottom: 0.5px solid #e0e0e0;'>
<div style="font-size:14px;">
@((context as Employee).FullName)
</div>
```



```

</td>
<td style="border-bottom: 0.5px solid #e0e0e0;">
<div>
<div style="position:relative;display:inline-block;">

</div>
<div style="display:inline-block;">
<div style="padding:5px;">@((context as Employee).Address)</div>
<div style="padding:5px;">@((context as Employee).Country)</div>
<div style="padding:5px;font-size:12px;">@((context as
Employee).Contact)</div>
</div>
</div>
</td>
<td style="padding-left: 20px; border-bottom: 0.5px solid #e0e0e0;">
<div>@((context as Employee).Designation)</div>
</td>
</RowTemplate>
</TreeGridTemplates>
<TreeGridColumn>
<TreeGridColumn Field="EmpID" HeaderText="Employee ID"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Name" HeaderText="Employee Name"></TreeGridColumn>
<TreeGridColumn Field="Address" HeaderText="Employee Details" Width="340"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Designation"
HeaderText="Designation"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
public Employee model = new Employee();
public IEnumerable<Employee> TreeData { get; set; }
protected override void OnInitialized()
{
this.TreeData = Employee.GetTemplateData();
}
}

```

C#

```






namespace TreeGridComponent.Data {
public class Employee
{
public string Name { get; set; }
public string FullName { get; set; }
public DateTime? DOB { get; set; }
public string Designation { get; set; }
public string EmpID { get; set; }
public int? EmployeeID { get; set; }
public string Country { get; set; }
public string Address { get; set; }
public string Contact { get; set; }
public int? ParentId { get; set; }
public TreeData Treedata { get; set; }
public static List<Employee> GetTemplateData()

```

```

{
    List<Employee> DataCollection = new List<Employee>();
    DataCollection.Add(new Employee { Name = "Robert King",FullName =
    "RobertKing",Designation = "Chief Executive Officer",EmployeeID = 1,EmpID =
    "EMP001",Address = "507 - 20th Ave. E.Apt. 2A, Seattle",Contact = "(206)
    555-9857",Country = "USA",DOB = new DateTime(1963, 2, 15),ParentId =
    null,Treedata = new TreeData() { ID = 21}}});
    DataCollection.Add(new Employee { Name = "David william",FullName =
    "DavidWilliam",Designation = "Vice President",EmployeeID = 2,EmpID =
    "EMP004",Address = "722 Moss Bay Blvd., Kirkland",Contact = "(206) 555-
    3412",Country = "USA",DOB = new DateTime(1971, 5, 20),ParentId = 1,Treedata
    = new TreeData() { ID = 21 }}});
    DataCollection.Add(new Employee { Name = "Nancy Davolio",FullName =
    "NancyDavolio",Designation = "Marketing Executive",EmployeeID = 3,EmpID =
    "EMP035",Address = "4110 Old Redmond Rd., Redmond",Contact = "(206) 555-
    8122",Country = "USA",DOB = new DateTime(1966, 3, 19),ParentId = 1,Treedata
    = new TreeData() { ID = 21 }}});
    DataCollection.Add(new Employee { Name = "Andrew Fuller",FullName =
    "AndrewFuller",Designation = "Sales Representative",EmployeeID = 4,EmpID =
    "EMP045",Country = "UK",DOB = new DateTime(1980, 9, 20),ParentId =
    1,Treedata = new TreeData() { ID = 21 }}});
    return DataCollection;
}
}
}

```

Employee ID	Employee Name	Employee Details	Designation
EMP001	RobertKing	507 - 20th Ave. E.Apt. 2A, Seattle  USA (206) 555-9857	Chief Executive Officer
EMP004	DavidWilliam	722 Moss Bay Blvd., Kirkland  USA (206) 555-3412	Vice President
EMP035	NancyDavolio	4110 Old Redmond Rd., Redmond  USA (206) 555-8122	Marketing Executive
EMP045	AndrewFuller	14 Garrett Hill, London  UK (71) 555-4848	Sales Representative
EMP091	AnneDodsworth	4726 - 11th Ave. N.E., Seattle  USA (206) 555-1189	Sales Representative

Row template with formatting

If [RowTemplate](#) is used, the value cannot be formatted inside the template using the [Format](#) property. In that case, a function should be defined globally to format the value and invoke it inside the template.

C#

```

@using TreeGridComponent.Data
@using Syncfusion.Blazor.TreeGrid;
@inject Microsoft.AspNetCore.Components.NavigationManager UriHelper
<SfTreeGrid ID="TreeGrid" Height="335" DataSource="@TreeData"
IdMapping="EmployeeID" ParentIdMapping="ParentId" TreeColumnIndex="0"
RowHeight="83" GridLines="Syncfusion.Blazor.Grids.GridLine.Vertical">
<TreeGridTemplates>
<RowTemplate>
<td style='padding-left:18px; border-bottom: 0.5px solid #e0e0e0;'>
<RowTemplateTreeColumn>
@{
var cntxt = context as Employee;
<div>@(cntxt.EmpID)</div>
}
</RowTemplateTreeColumn>
</td>
<td style='padding: 10px 0px 0px 20px; border-bottom: 0.5px solid #e0e0e0;'>
<div style="font-size:14px;">
@((context as Employee).FullName)
</div>
</td>
<td style="border-bottom: 0.5px solid #e0e0e0;">
<div>
<div style="position:relative;display:inline-block;">

</div>
<div style="display:inline-block;">
<div style="padding:5px;">@((context as Employee).Address)</div>
<div style="padding:5px;">@((context as Employee).Country)</div>
<div style="padding:5px;font-size:12px;">@((context as
Employee).Contact)</div>
</div>
</div>
</td>
<td style='padding-left: 20px; border-bottom: 0.5px solid #e0e0e0;'>
<div>@Format((context as Employee).DOB)</div>
</td>
</RowTemplate>
</TreeGridTemplates>
<TreeGridColumn>
<TreeGridColumn Field="EmpID" HeaderText="Employee ID"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Name" HeaderText="Employee Name"></TreeGridColumn>
<TreeGridColumn Field="Address" HeaderText="Employee Details" Width="340"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="DOB" HeaderText="DOB"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
public Employee model = new Employee();
public IEnumerable<Employee> TreeData { get; set; }
protected override void OnInitialized()
{
this.TreeData = Employee.GetTemplateData();
}
}

```

```

}
public object Format(DateTime? DOB) {
return DOB.Value.ToLocalTime();
}
}






```

C#

```

namespace TreeGridComponent.Data {
public class Employee
{
public string Name { get; set; }
public string FullName { get; set; }
public DateTime? DOB { get; set; }
public string Designation { get; set; }
public string EmpID { get; set; }
public int? EmployeeID { get; set; }
public string Country { get; set; }
public string Address { get; set; }
public string Contact { get; set; }
public int? ParentId { get; set; }
public TreeData Treedata { get; set; }
public static List<Employee> GetTemplateData()
{
List<Employee> DataCollection = new List<Employee>();
DataCollection.Add(new Employee { Name = "Robert King",FullName =
"RobertKing",Designation = "Chief Executive Officer",EmployeeID = 1,EmpID =
"EMP001",Address = "507 - 20th Ave. E.Apt. 2A, Seattle",Contact = "(206) 555-9857",Country = "USA",DOB = new DateTime(1963, 2, 15),ParentId =
null,Treedata = new TreeData() { ID = 21 } });
DataCollection.Add(new Employee { Name = "David william",FullName =
"DavidWilliam",Designation = "Vice President",EmployeeID = 2,EmpID =
"EMP004",Address = "722 Moss Bay Blvd., Kirkland",Contact = "(206) 555-3412",Country = "USA",DOB = new DateTime(1971, 5, 20),ParentId = 1,Treedata =
new TreeData() { ID = 21 } });
DataCollection.Add(new Employee { Name = "Nancy Davolio",FullName =
"NancyDavolio",Designation = "Marketing Executive",EmployeeID = 3,EmpID =
"EMP035",Address = "4110 Old Redmond Rd., Redmond",Contact = "(206) 555-8122",Country = "USA",DOB = new DateTime(1966, 3, 19),ParentId = 1,Treedata =
new TreeData() { ID = 21 } });
DataCollection.Add(new Employee { Name = "Andrew Fuller",FullName =
"AndrewFuller",Designation = "Sales Representative",EmployeeID = 4,EmpID =
"EMP045",Country = "UK",DOB = new DateTime(1980, 9, 20),ParentId =
1,Treedata = new TreeData() { ID = 21 } });
return DataCollection;
}
}
}

```

Employee ID	Employee Name	Employee Details		DOB
▼ EMP001	RobertKing		507 - 20th Ave. E Apt. 2A, Seattle USA (206) 555-9857	2/15/1963 12:00:00 AM
EMP004	DavidWilliam		722 Moss Bay Blvd., Kirkland USA (206) 555-3412	5/20/1971 12:00:00 AM
EMP035	NancyDavalio		4110 Old Redmond Rd., Redmond USA (206) 555-8122	3/19/1966 12:00:00 AM
EMP045	AndrewFuller		14 Garrett Hill, London UK (71) 555-4848	9/20/1980 12:00:00 AM
▼ EMP091	AnneDodsworth		4726 - 11th Ave. N.E., Seattle USA (206) 555-1189	1/1/1901 12:08:50 AM

Limitations

Row template feature is not compatible with all the features which are available in the tree grid and it has limited features support. Here the features which are compatible with row template feature are listed out.

- Filtering
- Paging
- Sorting
- Scrolling
- Searching
- Rtl
- Context Menu
- State Persistence

Detail template

The detail template provides additional information about a particular row. By expanding the parent row the child rows are expanded along with their detail template. The [DetailTemplate](#) property accepts either the template string or HTML elements.

C#

```
@using TreeGridComponent.Data
@using Syncfusion.Blazor.TreeGrid;
@inject Microsoft.AspNetCore.Components.NavigationManager UriHelper
<SfTreeGrid Height="400" DataSource="@TreeData" IdMapping="EmployeeID"
ParentIdMapping="ParentId" TreeColumnIndex="0">
<TreeGridTemplates>
```

```

<DetailTemplate>
<div style="position: relative; display: inline-block; float: left; font-
weight: bold; width: 10%;padding:5px 4px 2px 27px;;">

</div>
<div style="padding-left: 10px; display: inline-block; width: 66%; text-
wrap: normal;font-size:13px;font-family:'Segoe UI';">
<div class="e-description" style="word-wrap: break-word;">
<b>@((context as Employee).Name)</b> was lives at @((context as
Employee).Address), @((context as Employee).Country). @((context as
Employee).Name) holds a position of <b>@((context as
Employee).Designation)</b> in our WA department, (Seattle USA).
</div>
<div class="e-description" style="word-wrap: break-word;margin-top:5px;">
<b style="margin-right:10px;">Contact:</b>@((context as Employee).Contact)
</div>
</div>
</DetailTemplate>
</TreeGridTemplates>
<TreeGridColumn>
<TreeGridColumn Field="Name" HeaderText="Name" Width="160"></TreeGridColumn>
<TreeGridColumn Field="DOB" HeaderText="DOB" Width="10"
Type="Syncfusion.Blazor.Grids.ColumnType.Date"
Format="yMd"></TreeGridColumn>
<TreeGridColumn Field="Designation" HeaderText="Designation"
Width="120"></TreeGridColumn>
<TreeGridColumn Field="EmpID" HeaderText="Employee ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Country" HeaderText="Country"
Width="100"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
public Employee model = new Employee();
public IEnumerable<Employee> TreeData { get; set; }
protected override void OnInitialized()
{
this.TreeData = Employee.GetTemplateData();
}
}

```





C#

```

namespace TreeGridComponent.Data {
public class Employee
{
public string Name { get; set; }
public string FullName { get; set; }
public DateTime? DOB { get; set; }
public string Designation { get; set; }
public string EmpID { get; set; }
public int? EmployeeID { get; set; }
public string Country { get; set; }
public string Address { get; set; }
public string Contact { get; set; }
}
}

```

```
public int? ParentId { get; set; }
public TreeData Treedata { get; set; }
public static List<Employee> GetTemplateData()
{
    List<Employee> DataCollection = new List<Employee>();
    DataCollection.Add(new Employee { Name = "Robert King", FullName =
    "RobertKing", Designation = "Chief Executive Officer", EmployeeID = 1, EmpID =
    "EMP001", Address = "507 - 20th Ave. E.Apt. 2A, Seattle", Contact = "(206)
    555-9857", Country = "USA", DOB = new DateTime(1963, 2, 15), ParentId =
    null, Treedata = new TreeData() { ID = 21 } });
    DataCollection.Add(new Employee { Name = "David william", FullName =
    "DavidWilliam", Designation = "Vice President", EmployeeID = 2, EmpID =
    "EMP004", Address = "722 Moss Bay Blvd., Kirkland", Contact = "(206) 555-
    3412", Country = "USA", DOB = new DateTime(1971, 5, 20), ParentId = 1, Treedata
    = new TreeData() { ID = 21 } });
    DataCollection.Add(new Employee { Name = "Nancy Davolio", FullName =
    "NancyDavolio", Designation = "Marketing Executive", EmployeeID = 3, EmpID =
    "EMP035", Address = "4110 Old Redmond Rd., Redmond", Contact = "(206) 555-
    8122", Country = "USA", DOB = new DateTime(1966, 3, 19), ParentId = 1, Treedata
    = new TreeData() { ID = 21 } });
    DataCollection.Add(new Employee { Name = "Andrew Fuller", FullName =
    "AndrewFuller", Designation = "Sales Representative", EmployeeID = 4, EmpID =
    "EMP045", Country = "UK", DOB = new DateTime(1980, 9, 20), ParentId =
    1, Treedata = new TreeData() { ID = 21 } });
    return DataCollection;
}
}
```

Name	Designation	Employee ID	Country
▼ Robert King	Chief Executive Officer	EMP001	USA
 Robert King was lives at 507 - 20th Ave. E.Apt. 2A, Seattle, USA. Robert King holds a position of Chief Executive Officer in our WA department, (Seattle USA). Contact: (206) 555-9857			
David william	Vice President	EMP004	USA
 David william was lives at 722 Moss Bay Blvd., Kirkland, USA. David william holds a position of Vice President in our WA department, (Seattle USA). Contact: (206) 555-3412			
Nancy Davolio	Marketing Executive	EMP035	USA
 Nancy Davolio was lives at 4110 Old Redmond Rd., Redmond, USA. Nancy Davolio holds a position of Marketing Executive in our WA department, (Seattle USA). Contact: (206) 555-8122			
Andrew Fuller	Sales Representative	EMP045	UK
 Andrew Fuller was lives at 14 Garrett Hill, London, UK. Andrew Fuller holds a position of Sales Representative in our WA department, (Seattle USA). Contact: (71) 555-4848			
▼ Anne Dodsworth	Sales Representative	EMP091	USA

<!-- Customize row height for particular row

Grid row height for particular row can be customized using the [RowDataBound](#)

event by setting the [RowHeight](#) in arguments for each row based on the requirement.

In the below example, the row height for the row with Task ID as 3 is set as 90px using the [RowDataBound](#) event.

CSHARP

```
@using TreeGridComponent.Data
@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.Data;
<SfTreeGrid ChildMapping="Children" TreeColumnIndex="1"
RowDataBound="@onRowDataBound">
<SfDataManager Json="@TreeGridData"
Adaptor="Syncfusion.Blazor.Adaptors.JsonAdaptor"></SfDataManager>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
</TreeGridColumn>
```



```

</SfTreeGrid>
@code{
public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetDefaultData().ToList();
}
private void onRowDataBound(RowDataBoundEventArgs args)
{
int val =
int.Parse(args.Data.GetType().GetProperty("TaskId").GetValue(args.Data,
null).ToString());
if (val == 3)
{
args.RowHeight = 90;
}
}
}
}

```

-->

[See Also](#)

- [TreeGridTemplates component](#)

Drag and drop

The Tree Grid rows can be reordered, dropped to another Tree Grid or custom control by enabling the [AllowRowDragAndDrop](#) to true.

Drag and drop within TreeGrid

The Tree Grid row drag and drop allows to drag and drop Tree Grid rows on the same Tree Grid using drag icon. To enable row drag and drop, set the [AllowRowDragAndDrop](#) to true. It provides the way to drop the row above, below or child to the target row with respective to the target row position.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" AllowRowDragAndDrop="true"
TreeColumnIndex="1" IdMapping="TaskId" ParentIdMapping="ParentId"
AllowPaging="true">
<TreeGridPageSettings PageSize="2"></TreeGridPageSettings>
<TreeGridSelectionSettings
Type="Syncfusion.Blazor.Grids.SelectionType.Multiple"></TreeGridSelectionSet
tings>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
IsPrimaryKey="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="240"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText="Start Date" Format="d"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>

```

```

<TreeGridColumn Field="Progress" HeaderText="Progress"
Width="100"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public List<WrapData> TreeGridData { get; set; }
protected override void OnInitialized()
{
TreeGridData = WrapData.GetWrapData();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class WrapData
{
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 string Progress { get; set; }
public string Priority { get; set; }
public bool Approved { get; set; }
public int Resources { get; set; }
public int? ParentId { get; set; }
public static List<WrapData> GetWrapData()
{
List<WrapData> BusinessObjectCollection = new List<WrapData>();
BusinessObjectCollection.Add(new WrapData()
{
TaskId = 1,
TaskName = "Planning",
StartDate = new DateTime(2017, 03, 02),
EndDate = new DateTime(2017, 07, 03),
Progress = "Open",
Duration = 5,
Priority = "Normal",
Resources = 6,
Approved = false,
ParentId = null
});
BusinessObjectCollection.Add(new WrapData()
{
TaskId = 2,
TaskName = "Plan timeline",
StartDate = new DateTime(2017, 03, 04),
EndDate = new DateTime(2017, 07, 05),
Progress = "Inprogress",
Duration = 5,
Resources = 4,
Priority = "Normal",
Approved = false,
ParentId = 1
});
}
}

```

```
BusinessObjectCollection.Add(new WrapData()  
{  
    TaskId = 3,  
    TaskName = "Plan budget",  
    StartDate = new DateTime(2017, 03, 06),  
    EndDate = new DateTime(2017, 07, 07),  
    Duration = 5,  
    Progress = "Started",  
    Approved = true,  
    Resources = 6,  
    Priority = "Low",  
    ParentId = 1  
});  
BusinessObjectCollection.Add(new WrapData()  
{  
    TaskId = 4,  
    TaskName = "Allocate resources",  
    StartDate = new DateTime(2017, 03, 08),  
    EndDate = new DateTime(2017, 07, 09),  
    Duration = 5,  
    Progress = "Open",  
    Priority = "Critical",  
    ParentId = 1,  
    Resources = 3,  
    Approved = false  
});  
BusinessObjectCollection.Add(new WrapData()  
{  
    TaskId = 5,  
    TaskName = "Planning complete",  
    StartDate = new DateTime(2017, 07, 10),  
    EndDate = new DateTime(2017, 07, 11),  
    Duration = 1,  
    Progress = "Open",  
    Priority = "Low",  
    Resources = 5,  
    ParentId = 1,  
    Approved = true  
});  
BusinessObjectCollection.Add(new WrapData()  
{  
    TaskId = 6,  
    TaskName = "Design",  
    StartDate = new DateTime(2017, 10, 12),  
    EndDate = new DateTime(2017, 02, 13),  
    Progress = "Inprogress",  
    Duration = 3,  
    Priority = "High",  
    Resources = 4,  
    Approved = false,  
    ParentId = null  
});  
BusinessObjectCollection.Add(new WrapData()  
{  
    TaskId = 7,  
    TaskName = "Software Specification",  
    StartDate = new DateTime(2017, 10, 14),
```

```
EndDate = new DateTime(2017, 02, 15),
Duration = 3,
Progress = "Started",
Resources = 3,
Priority = "Normal",
ParentId = 6,
Approved = false
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 8,
    TaskName = "Develop prototype",
    StartDate = new DateTime(2017, 10, 16),
    EndDate = new DateTime(2017, 02, 17),
    Duration = 3,
    Progress = "Inprogress",
    Resources = 2,
    Priority = "Critical",
    ParentId = 6,
    Approved = false
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 9,
    TaskName = "Get approval from customer",
    StartDate = new DateTime(2017, 02, 18),
    EndDate = new DateTime(2017, 02, 19),
    Duration = 2,
    Progress = "Inprogress",
    Resources = 3,
    Priority = "Low",
    Approved = true,
    ParentId = 6
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 10,
    TaskName = "Design complete",
    StartDate = new DateTime(2017, 02, 20),
    EndDate = new DateTime(2017, 02, 21),
    Duration = 1,
    Progress = "Inprogress",
    Resources = 6,
    Priority = "Normal",
    ParentId = 6,
    Approved = true
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 12,
    TaskName = "Implementation Phase",
    StartDate = new DateTime(2017, 02, 22),
    EndDate = new DateTime(2017, 02, 23),
    Priority = "Normal",
    Approved = false,
    Duration = 11,
    Resources = 5,
```

```
Progress = "Started",
ParentId = null
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 13,
    TaskName = "Phase 1",
    StartDate = new DateTime(2017, 02, 24),
    EndDate = new DateTime(2017, 02, 25),
    Priority = "High",
    Approved = false,
    Duration = 11,
    Progress = "Open",
    Resources = 4,
    ParentId = 12
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 14,
    TaskName = "Implementation Module 1",
    StartDate = new DateTime(2017, 02, 26),
    EndDate = new DateTime(2017, 02, 27),
    Priority = "Normal",
    Duration = 11,
    Progress = "Started",
    Resources = 3,
    Approved = false,
    ParentId = 13
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 15,
    TaskName = "Development Task 1",
    StartDate = new DateTime(2017, 06, 18),
    EndDate = new DateTime(2017, 06, 19),
    Duration = 3,
    Progress = "Inprogress",
    Priority = "High",
    Resources = 2,
    ParentId = 14,
    Approved = false
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 16,
    TaskName = "Development Task 2",
    StartDate = new DateTime(2017, 02, 13),
    EndDate = new DateTime(2017, 03, 01),
    Duration = 3,
    Progress = "Closed",
    Priority = "Low",
    Resources = 5,
    ParentId = 14,
    Approved = true
});
BusinessObjectCollection.Add(new WrapData()
{
```

```
TaskId = 17,
TaskName = "Testing",
StartDate = new DateTime(2017, 03, 02),
EndDate = new DateTime(2017, 03, 03),
Duration = 2,
Progress = "Closed",
Priority = "Normal",
ParentId = 14,
Resources = 1,
Approved = true
});
BusinessObjectCollection.Add(new WrapData()
{
TaskId = 18,
TaskName = "Bug fix",
StartDate = new DateTime(2017, 03, 04),
EndDate = new DateTime(2017, 03, 05),
Duration = 2,
Progress = "Validated",
Priority = "Critical",
ParentId = 14,
Resources = 6,
Approved = false
});
BusinessObjectCollection.Add(new WrapData()
{
TaskId = 19,
TaskName = "Customer review meeting",
StartDate = new DateTime(2017, 03, 06),
EndDate = new DateTime(2017, 03, 07),
Duration = 2,
Progress = "Open",
Priority = "High",
ParentId = 14,
Resources = 6,
Approved = false
});
BusinessObjectCollection.Add(new WrapData()
{
TaskId = 20,
TaskName = "Phase 1 complete",
StartDate = new DateTime(2017, 04, 27),
EndDate = new DateTime(2017, 07, 27),
Duration = 2,
Progress = "Closed",
Priority = "Low",
ParentId = 14,
Resources = 5,
Approved = true
});
BusinessObjectCollection.Add(new WrapData()
{
TaskId = 21,
TaskName = "Phase 2",
StartDate = new DateTime(2017, 07, 17),
EndDate = new DateTime(2017, 09, 28),
Priority = "High",
```

```
Approved = false,
Progress = "Open",
ParentId = 12,
Resources = 3,
Duration = 12,
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 22,
    TaskName = "Implementation Module 2",
    StartDate = new DateTime(2017, 01, 17),
    EndDate = new DateTime(2017, 02, 28),
    Priority = "Critical",
    Approved = false,
    Progress = "Inprogress",
    ParentId = 21,
    Resources = 3,
    Duration = 12
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 23,
    TaskName = "Development Task 1",
    StartDate = new DateTime(2017, 08, 17),
    EndDate = new DateTime(2017, 09, 20),
    Duration = 4,
    Progress = "Closed",
    Priority = "Normal",
    ParentId = 22,
    Resources = 2,
    Approved = true,
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 24,
    TaskName = "Development Task 2",
    StartDate = new DateTime(2017, 04, 17),
    EndDate = new DateTime(2017, 03, 20),
    Duration = 4,
    Progress = "Closed",
    Priority = "Critical",
    ParentId = 22,
    Resources = 5,
    Approved = true,
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 25,
    TaskName = "Testing",
    StartDate = new DateTime(2017, 01, 21),
    EndDate = new DateTime(2017, 01, 24),
    Duration = 2,
    Progress = "Open",
    Priority = "High",
    ParentId = 22,
    Resources = 3,
    Approved = false,
```

```
});  
BusinessObjectCollection.Add(new WrapData()  
{  
    TaskId = 26,  
    TaskName = "Bug fix",  
    StartDate = new DateTime(2017, 03, 25),  
    EndDate = new DateTime(2017, 08, 26),  
    Duration = 2,  
    Progress = "Validated",  
    Priority = "Low",  
    Approved = false,  
    Resources = 6,  
    ParentId = 22  
});  
BusinessObjectCollection.Add(new WrapData()  
{  
    TaskId = 27,  
    TaskName = "Customer review meeting",  
    StartDate = new DateTime(2017, 07, 27),  
    EndDate = new DateTime(2017, 06, 28),  
    Duration = 2,  
    Progress = "Inprogress",  
    Priority = "Critical",  
    ParentId = 22,  
    Resources = 4,  
    Approved = true,  
});  
BusinessObjectCollection.Add(new WrapData()  
{  
    TaskId = 28,  
    TaskName = "Phase 2 complete",  
    StartDate = new DateTime(2017, 07, 19),  
    EndDate = new DateTime(2017, 05, 28),  
    Duration = 2,  
    Priority = "Normal",  
    Progress = "Open",  
    ParentId = 22,  
    Resources = 3,  
    Approved = false,  
});  
BusinessObjectCollection.Add(new WrapData()  
{  
    TaskId = 29,  
    TaskName = "Phase 3",  
    StartDate = new DateTime(2017, 07, 17),  
    EndDate = new DateTime(2017, 02, 12),  
    Priority = "Normal",  
    Approved = false,  
    Duration = 11,  
    Progress = "Inprogress",  
    Resources = 4,  
    ParentId = 12  
});  
BusinessObjectCollection.Add(new WrapData()  
{  
    TaskId = 30,  
    TaskName = "Implementation Module 3",
```



```
StartDate = new DateTime(2017, 08, 17),
EndDate = new DateTime(2017, 09, 27),
Priority = "High",
Approved = false,
Duration = 11,
Resources = 5,
Progress = "Validated",
ParentId = 29,
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 31,
    TaskName = "Development Task 1",
    StartDate = new DateTime(2017, 11, 17),
    EndDate = new DateTime(2017, 12, 19),
    Duration = 3,
    Progress = "Closed",
    Priority = "Low",
    Approved = true,
    Resources = 3,
    ParentId = 30
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 32,
    TaskName = "Development Task 2",
    StartDate = new DateTime(2017, 12, 17),
    EndDate = new DateTime(2017, 02, 19),
    Duration = 3,
    Progress = "Closed",
    Priority = "Normal",
    Approved = false,
    Resources = 2,
    ParentId = 30
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 33,
    TaskName = "Testing",
    StartDate = new DateTime(2017, 01, 01),
    EndDate = new DateTime(2017, 07, 21),
    Duration = 2,
    Progress = "Closed",
    Priority = "Critical",
    ParentId = 30,
    Resources = 4,
    Approved = true,
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 34,
    TaskName = "Bug fix",
    StartDate = new DateTime(2017, 01, 24),
    EndDate = new DateTime(2017, 01, 25),
    Duration = 2,
    Progress = "Open",
    Priority = "High",
```

```

Approved = false,
Resources = 3,
ParentId = 30
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 35,
    TaskName = "Customer review meeting",
    StartDate = new DateTime(2017, 12, 26),
    EndDate = new DateTime(2017, 12, 27),
    Duration = 2,
    Progress = "Inprogress",
    Priority = "Normal",
    ParentId = 30,
    Resources = 6,
    Approved = true,
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 36,
    TaskName = "Phase 3 complete",
    StartDate = new DateTime(2017, 05, 27),
    EndDate = new DateTime(2017, 05, 27),
    Duration = 2,
    Priority = "Critical",
    Progress = "Open",
    Resources = 5,
    ParentId = 30,
    Approved = false,
});
return BusinessObjectCollection;
}
}
}

```

Task ID	Task Name	Start Date	Duration	Progress
1	Planning	3/2/2017	5	Open
2	Plan timeline	3/4/2017	5	Inprogress
3	4 Allocate resources	3/8/2017	5	Started
4	Allocate resources	3/8/2017	5	Open
5	Planning complete	7/10/2017	1	Open
6	Design	10/12/2017	3	Inprogress
7	Software Specification	10/14/2017	3	Started
8	Develop prototype	10/16/2017	3	Inprogress
9	Get approval from customer	2/18/2017	2	Inprogress
10	Design complete	2/20/2017	1	Inprogress

1 of 2 pages (3 items)

Selection feature must be enabled for row drag and drop.

For multiple row selection, the type property must be set to multiple.

Drag and drop to another TreeGrid

To drag and drop between two Tree Grid, enable the [AllowRowDragAndDrop](#) property and specify the target Tree Grid ID in [TargetID](#) property of [RowDropSettings](#).

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<div id='container'>
<div>
<div style="float: left; width:49%" id="Grid">
<SfTreeGrid ID="Grid" DataSource="@TreeGridData" AllowRowDragAndDrop="true"
AllowSelection="true" AllowPaging="true" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridSelectionSettings
Type="Syncfusion.Blazor.Grids.SelectionType.Multiple"></TreeGridSelectionSet
tings>
<TreeGridRowDropSettings TargetID="DestGrid"></TreeGridRowDropSettings>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
</div>
<div style="float: right; width:49%">
<SfTreeGrid ID="DestGrid" DataSource="@SecondGrid"
AllowRowDragAndDrop="true" AllowSelection="true" AllowPaging="true"
IdMapping="TaskId" ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridSelectionSettings
Type="Syncfusion.Blazor.Grids.SelectionType.Multiple"></TreeGridSelectionSet
tings>
<TreeGridRowDropSettings TargetID="Grid"></TreeGridRowDropSettings>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
</div>
</div>
@code{
public List<TreeData.BusinessObject> TreeGridData { get; set; }
public List<TreeData.BusinessObject> SecondGrid { get; set; } = new
List<TreeData.BusinessObject>();
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
```

```
}
```

C#

```
namespace TreeGridComponent.Data {
public class WrapData
{
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 string Progress { get; set; }
public string Priority { get; set; }
public bool Approved { get; set; }
public int Resources { get; set; }
public int? ParentId { get; set; }
public static List<WrapData> GetWrapData()
{
List<WrapData> BusinessObjectCollection = new List<WrapData>();
BusinessObjectCollection.Add(new WrapData()
{
TaskId = 1,
TaskName = "Planning",
StartDate = new DateTime(2017, 03, 02),
EndDate = new DateTime(2017, 07, 03),
Progress = "Open",
Duration = 5,
Priority = "Normal",
Resources = 6,
Approved = false,
ParentId = null
});
BusinessObjectCollection.Add(new WrapData()
{
TaskId = 2,
TaskName = "Plan timeline",
StartDate = new DateTime(2017, 03, 04),
EndDate = new DateTime(2017, 07, 05),
Progress = "Inprogress",
Duration = 5,
Resources = 4,
Priority = "Normal",
Approved = false,
ParentId = 1
});
BusinessObjectCollection.Add(new WrapData()
{
TaskId = 3,
TaskName = "Plan budget",
StartDate = new DateTime(2017, 03, 06),
EndDate = new DateTime(2017, 07, 07),
Duration = 5,
Progress = "Started",
Approved = true,
Resources = 6,
```

```
Priority = "Low",
ParentId = 1
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 4,
    TaskName = "Allocate resources",
    StartDate = new DateTime(2017, 03, 08),
    EndDate = new DateTime(2017, 07, 09),
    Duration = 5,
    Progress = "Open",
    Priority = "Critical",
    ParentId = 1,
    Resources = 3,
    Approved = false
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 5,
    TaskName = "Planning complete",
    StartDate = new DateTime(2017, 07, 10),
    EndDate = new DateTime(2017, 07, 11),
    Duration = 1,
    Progress = "Open",
    Priority = "Low",
    Resources = 5,
    ParentId = 1,
    Approved = true
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 6,
    TaskName = "Design",
    StartDate = new DateTime(2017, 10, 12),
    EndDate = new DateTime(2017, 02, 13),
    Progress = "Inprogress",
    Duration = 3,
    Priority = "High",
    Resources = 4,
    Approved = false,
    ParentId = null
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 7,
    TaskName = "Software Specification",
    StartDate = new DateTime(2017, 10, 14),
    EndDate = new DateTime(2017, 02, 15),
    Duration = 3,
    Progress = "Started",
    Resources = 3,
    Priority = "Normal",
    ParentId = 6,
    Approved = false
});
BusinessObjectCollection.Add(new WrapData()
{
```

```
TaskId = 8,
TaskName = "Develop prototype",
StartDate = new DateTime(2017, 10, 16),
EndDate = new DateTime(2017, 02, 17),
Duration = 3,
Progress = "Inprogress",
Resources = 2,
Priority = "Critical",
ParentId = 6,
Approved = false
});
BusinessObjectCollection.Add(new WrapData()
{
TaskId = 9,
TaskName = "Get approval from customer",
StartDate = new DateTime(2017, 02, 18),
EndDate = new DateTime(2017, 02, 19),
Duration = 2,
Progress = "Inprogress",
Resources = 3,
Priority = "Low",
Approved = true,
ParentId = 6
});
BusinessObjectCollection.Add(new WrapData()
{
TaskId = 10,
TaskName = "Design complete",
StartDate = new DateTime(2017, 02, 20),
EndDate = new DateTime(2017, 02, 21),
Duration = 1,
Progress = "Inprogress",
Resources = 6,
Priority = "Normal",
ParentId = 6,
Approved = true
});
BusinessObjectCollection.Add(new WrapData()
{
TaskId = 12,
TaskName = "Implementation Phase",
StartDate = new DateTime(2017, 02, 22),
EndDate = new DateTime(2017, 02, 23),
Priority = "Normal",
Approved = false,
Duration = 11,
Resources = 5,
Progress = "Started",
ParentId = null
});
BusinessObjectCollection.Add(new WrapData()
{
TaskId = 13,
TaskName = "Phase 1",
StartDate = new DateTime(2017, 02, 24),
EndDate = new DateTime(2017, 02, 25),
Priority = "High",
```

```
Approved = false,
Duration = 11,
Progress = "Open",
Resources = 4,
ParentId = 12
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 14,
    TaskName = "Implementation Module 1",
    StartDate = new DateTime(2017, 02, 26),
    EndDate = new DateTime(2017, 02, 27),
    Priority = "Normal",
    Duration = 11,
    Progress = "Started",
    Resources = 3,
    Approved = false,
    ParentId = 13
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 15,
    TaskName = "Development Task 1",
    StartDate = new DateTime(2017, 06, 18),
    EndDate = new DateTime(2017, 06, 19),
    Duration = 3,
    Progress = "Inprogress",
    Priority = "High",
    Resources = 2,
    ParentId = 14,
    Approved = false
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 16,
    TaskName = "Development Task 2",
    StartDate = new DateTime(2017, 02, 13),
    EndDate = new DateTime(2017, 03, 01),
    Duration = 3,
    Progress = "Closed",
    Priority = "Low",
    Resources = 5,
    ParentId = 14,
    Approved = true
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 17,
    TaskName = "Testing",
    StartDate = new DateTime(2017, 03, 02),
    EndDate = new DateTime(2017, 03, 03),
    Duration = 2,
    Progress = "Closed",
    Priority = "Normal",
    ParentId = 14,
    Resources = 1,
    Approved = true
});
```

```
});  
BusinessObjectCollection.Add(new WrapData()  
{  
    TaskId = 18,  
    TaskName = "Bug fix",  
    StartDate = new DateTime(2017, 03, 04),  
    EndDate = new DateTime(2017, 03, 05),  
    Duration = 2,  
    Progress = "Validated",  
    Priority = "Critical",  
    ParentId = 14,  
    Resources = 6,  
    Approved = false  
});  
BusinessObjectCollection.Add(new WrapData()  
{  
    TaskId = 19,  
    TaskName = "Customer review meeting",  
    StartDate = new DateTime(2017, 03, 06),  
    EndDate = new DateTime(2017, 03, 07),  
    Duration = 2,  
    Progress = "Open",  
    Priority = "High",  
    ParentId = 14,  
    Resources = 6,  
    Approved = false  
});  
BusinessObjectCollection.Add(new WrapData()  
{  
    TaskId = 20,  
    TaskName = "Phase 1 complete",  
    StartDate = new DateTime(2017, 04, 27),  
    EndDate = new DateTime(2017, 07, 27),  
    Duration = 2,  
    Progress = "Closed",  
    Priority = "Low",  
    ParentId = 14,  
    Resources = 5,  
    Approved = true  
});  
BusinessObjectCollection.Add(new WrapData()  
{  
    TaskId = 21,  
    TaskName = "Phase 2",  
    StartDate = new DateTime(2017, 07, 17),  
    EndDate = new DateTime(2017, 09, 28),  
    Priority = "High",  
    Approved = false,  
    Progress = "Open",  
    ParentId = 12,  
    Resources = 3,  
    Duration = 12,  
});  
BusinessObjectCollection.Add(new WrapData()  
{  
    TaskId = 22,  
    TaskName = "Implementation Module 2",
```



```
StartDate = new DateTime(2017, 01, 17),
EndDate = new DateTime(2017, 02, 28),
Priority = "Critical",
Approved = false,
Progress = "Inprogress",
ParentId = 21,
Resources = 3,
Duration = 12
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 23,
    TaskName = "Development Task 1",
    StartDate = new DateTime(2017, 08, 17),
    EndDate = new DateTime(2017, 09, 20),
    Duration = 4,
    Progress = "Closed",
    Priority = "Normal",
    ParentId = 22,
    Resources = 2,
    Approved = true,
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 24,
    TaskName = "Development Task 2",
    StartDate = new DateTime(2017, 04, 17),
    EndDate = new DateTime(2017, 03, 20),
    Duration = 4,
    Progress = "Closed",
    Priority = "Critical",
    ParentId = 22,
    Resources = 5,
    Approved = true,
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 25,
    TaskName = "Testing",
    StartDate = new DateTime(2017, 01, 21),
    EndDate = new DateTime(2017, 01, 24),
    Duration = 2,
    Progress = "Open",
    Priority = "High",
    ParentId = 22,
    Resources = 3,
    Approved = false,
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 26,
    TaskName = "Bug fix",
    StartDate = new DateTime(2017, 03, 25),
    EndDate = new DateTime(2017, 08, 26),
    Duration = 2,
    Progress = "Validated",
    Priority = "Low",
```

```
Approved = false,
Resources = 6,
ParentId = 22
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 27,
    TaskName = "Customer review meeting",
    StartDate = new DateTime(2017, 07, 27),
    EndDate = new DateTime(2017, 06, 28),
    Duration = 2,
    Progress = "Inprogress",
    Priority = "Critical",
    ParentId = 22,
    Resources = 4,
    Approved = true,
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 28,
    TaskName = "Phase 2 complete",
    StartDate = new DateTime(2017, 07, 19),
    EndDate = new DateTime(2017, 05, 28),
    Duration = 2,
    Priority = "Normal",
    Progress = "Open",
    ParentId = 22,
    Resources = 3,
    Approved = false,
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 29,
    TaskName = "Phase 3",
    StartDate = new DateTime(2017, 07, 17),
    EndDate = new DateTime(2017, 02, 12),
    Priority = "Normal",
    Approved = false,
    Duration = 11,
    Progress = "Inprogress",
    Resources = 4,
    ParentId = 12
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 30,
    TaskName = "Implementation Module 3",
    StartDate = new DateTime(2017, 08, 17),
    EndDate = new DateTime(2017, 09, 27),
    Priority = "High",
    Approved = false,
    Duration = 11,
    Resources = 5,
    Progress = "Validated",
    ParentId = 29,
});
BusinessObjectCollection.Add(new WrapData()
```

```
{
    TaskId = 31,
    TaskName = "Development Task 1",
    StartDate = new DateTime(2017, 11, 17),
    EndDate = new DateTime(2017, 12, 19),
    Duration = 3,
    Progress = "Closed",
    Priority = "Low",
    Approved = true,
    Resources = 3,
    ParentId = 30
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 32,
    TaskName = "Development Task 2",
    StartDate = new DateTime(2017, 12, 17),
    EndDate = new DateTime(2017, 02, 19),
    Duration = 3,
    Progress = "Closed",
    Priority = "Normal",
    Approved = false,
    Resources = 2,
    ParentId = 30
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 33,
    TaskName = "Testing",
    StartDate = new DateTime(2017, 01, 01),
    EndDate = new DateTime(2017, 07, 21),
    Duration = 2,
    Progress = "Closed",
    Priority = "Critical",
    ParentId = 30,
    Resources = 4,
    Approved = true,
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 34,
    TaskName = "Bug fix",
    StartDate = new DateTime(2017, 01, 24),
    EndDate = new DateTime(2017, 01, 25),
    Duration = 2,
    Progress = "Open",
    Priority = "High",
    Approved = false,
    Resources = 3,
    ParentId = 30
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 35,
    TaskName = "Customer review meeting",
    StartDate = new DateTime(2017, 12, 26),
    EndDate = new DateTime(2017, 12, 27),
```

```

Duration = 2,
Progress = "Inprogress",
Priority = "Normal",
ParentId = 30,
Resources = 6,
Approved = true,
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 36,
    TaskName = "Phase 3 complete",
    StartDate = new DateTime(2017, 05, 27),
    EndDate = new DateTime(2017, 05, 27),
    Duration = 2,
    Priority = "Critical",
    Progress = "Open",
    Resources = 5,
    ParentId = 30,
    Approved = false,
});
return BusinessObjectCollection;
}
}
}

```

Task ID	Task Name	Progress
1	▼ Planning	Open
2	Plan timeline	Inprogress
3	Plan budget	Started
4	Allocate resources	Open
5	Planning complete	Open
6	▼ Design	Inprogress
7	Software Specification	Started
8	Develop prototype	Inprogress
9	Get approval from cu...	Inprogress
10	Design complete	Inprogress

« < 1 2 > » 1 of 2 pages (3 items)

Task ID	Task Name	Progress
3	Plan budget	Started

« < > » 0 of 0 pages (0 items)

Drag and drop events

The following events are triggered while drag and drop the tree grid rows.

[OnRowDragStart](#) -Triggers when starts to drag the tree grid row.

[RowDropped](#) - Triggers when a drag element is dropped on the target element.

Templates in Blazor TreeGrid Component

Blazor has templated components which accepts one or more UI segments as input that can be rendered as part of the component during component rendering. Tree Grid is a templated razor

component, that allows to customize various part of the UI using template parameters. It allows to render custom components or content based on own logic.

The available template options in tree grid are as follows,

- [Column template](#) - Used to customize cell content.
- [Header template](#) - Used to customize header cell content.
- [Row template](#) - Used to customize row content.
- [Detail template](#) - Used to customize the detail cell content.

Template ModelType

To use templates, the tree grid must be bound with named model. This can be done by specifying the model type using the `ModelType` property of the tree grid component as follows.

C#

```
@using TreeGridComponent.Data
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid ModelType="@model" Height="400" DataSource="@TreeData"
IdMapping="EmployeeID" ParentIdMapping="ParentId" TreeColumnIndex="0">
<TreeGridColumns>
<TreeGridColumn Field="Name" HeaderText="Name" Width="160">
<HeaderTemplate>
<div class="rating">
<span class="star"></span> DOB
</div>
</HeaderTemplate>
</TreeGridColumn>
<TreeGridColumn Field="Designation" HeaderText="Designation"
Width="120"></TreeGridColumn>
<TreeGridColumn Field="EmpID" HeaderText="Progress" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Country" HeaderText="Priority"
Width="100"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
<style>
.rating .star:before {
content: '★';
}
</style>
@code{
public Employee model = new Employee();
public IEnumerable<Employee> TreeData { get; set; }
protected override void OnInitialized()
{
this.TreeData = Employee.GetTemplateData();
}
}
```

C#

```
namespace TreeGridComponent.Data {
public class Employee
{
```

```

public string Name { get; set; }
public string FullName { get; set; }
public DateTime? DOB { get; set; }
public string Designation { get; set; }
public string EmpID { get; set; }
public int? EmployeeID { get; set; }
public string Country { get; set; }
public string Address { get; set; }
public string Contact { get; set; }
public int? ParentId { get; set; }
public TreeData Treedata { get; set; }
public static List<Employee> GetTemplateData()
{
    List<Employee> DataCollection = new List<Employee>();
    DataCollection.Add(new Employee { Name = "Robert King", FullName =
    "RobertKing", Designation = "Chief Executive Officer", EmployeeID = 1, EmpID =
    "EMP001", Country = "USA", DOB = new DateTime(1963, 2, 15), ParentId =
    null, Treedata = new TreeData() { ID = 21 } });
    DataCollection.Add(new Employee { Name = "David william", FullName =
    "DavidWilliam", Designation = "Vice President", EmployeeID = 2, EmpID =
    "EMP004", Country = "USA", DOB = new DateTime(1971, 5, 20), ParentId =
    1, Treedata = new TreeData() { ID = 21 } });
    DataCollection.Add(new Employee { Name = "Nancy Davolio", FullName =
    "NancyDavolio", Designation = "Marketing Executive", EmployeeID = 3, EmpID =
    "EMP035", Country = "USA", DOB = new DateTime(1966, 3, 19), ParentId =
    1, Treedata = new TreeData() { ID = 21 } });
    DataCollection.Add(new Employee { Name = "Andrew Fuller", FullName =
    "AndrewFuller", Designation = "Sales Representative", EmployeeID = 4, EmpID =
    "EMP045", Country = "UK", DOB = new DateTime(1980, 9, 20), ParentId =
    1, Treedata = new TreeData() { ID = 21 } });
    DataCollection.Add(new Employee { Name = "Anne Dodsworth", FullName =
    "AnneDodsworth", Designation = "Sales Representative", EmployeeID = 5, EmpID =
    "EMP091", Country = "USA", ParentId = null, Treedata = new TreeData() { ID = 21
    } });
    DataCollection.Add(new Employee { Name = "Michael Suyama", FullName =
    "MichaelSuyama", Designation = "Sales Representative", EmployeeID = 6, EmpID =
    "EMP110", Country = "UK", ParentId = 5, Treedata = new TreeData() { ID = 21
    } });
    return DataCollection;
}
}
}

```

★ DOB	Designation	Progress	Priority
▼ Robert King	Chief Executive O...	EMP001	USA
David william	Vice President	EMP004	USA
Nancy Davolio	Marketing Execut...	EMP035	USA
Andrew Fuller	Sales Representa...	EMP045	UK
▼ Anne Dodsworth	Sales Representa...	EMP091	USA
Michael Suyama	Sales Representa...	EMP110	UK
Janet Leverling	Sales Coordinator	EMP131	UK

Template Context

Most of the templates used by tree grid are of type `RenderFragment<T>` and they will be passed with parameters. The parameters passed can be accessed to the templates using implicit parameter named `context`. This implicit parameter name can also be changed using the `Context` attribute.

For example, the data of the column template can be accessed using `context` as follows.

C#

```
@using TreeGridComponent.Data
@using Syncfusion.Blazor.TreeGrid;
@inject Microsoft.AspNetCore.Components.NavigationManager UriHelper
<SfTreeGrid ModelType="@model" Height="400" DataSource="@TreeData"
IdMapping="EmployeeID" ParentIdMapping="ParentId" TreeColumnIndex="0">
<TreeGridColumn>
<TreeGridColumn Field="Name" HeaderText="Name" Width="160"></TreeGridColumn>
<TreeGridColumn HeaderText="Employee Image" Width="80">
<Template>
@{
var employee = (context as Employee);
<div class="image">

</div>
}
</Template>
</TreeGridColumn>
<TreeGridColumn Field="DOB" HeaderText="DOB" Width="10"
Type="Syncfusion.Blazor.Grids.ColumnType.Date"
Format="yMd"></TreeGridColumn>
<TreeGridColumn Field="Designation" HeaderText="Designation"
Width="120"></TreeGridColumn>
<TreeGridColumn Field="EmpID" HeaderText="Progress" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
```

```

<TreeGridColumn Field="Country" HeaderText="Priority"
Width="100"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public Employee model = new Employee();
public IEnumerable<Employee> TreeData { get; set; }
protected override void OnInitialized()
{
    this.TreeData = Employee.GetTemplateData();
}
}







```

C#

```

namespace TreeGridComponent.Data {
public class Employee
{
    public string Name { get; set; }
    public string FullName { get; set; }
    public DateTime? DOB { get; set; }
    public string Designation { get; set; }
    public string EmpID { get; set; }
    public int? EmployeeID { get; set; }
    public string Country { get; set; }
    public string Address { get; set; }
    public string Contact { get; set; }
    public int? ParentId { get; set; }
    public TreeData Treedata { get; set; }
    public static List<Employee> GetTemplateData()
    {
        List<Employee> DataCollection = new List<Employee>();
        DataCollection.Add(new Employee { Name = "Robert King",FullName =
"RobertKing",Designation = "Chief Executive Officer",EmployeeID = 1,EmpID =
"EMP001",Address = "507 - 20th Ave. E.Apt. 2A, Seattle",Contact = "(206)
555-9857",Country = "USA",DOB = new DateTime(1963, 2, 15),ParentId =
null,Treedata = new TreeData() { ID = 21}});
        DataCollection.Add(new Employee { Name = "David william",FullName =
"DavidWilliam",Designation = "Vice President",EmployeeID = 2,EmpID =
"EMP004",Address = "722 Moss Bay Blvd., Kirkland",Contact = "(206) 555-
3412",Country = "USA",DOB = new DateTime(1971, 5, 20),ParentId = 1,Treedata
= new TreeData() { ID = 21 }});
        DataCollection.Add(new Employee { Name = "Nancy Davolio",FullName =
"NancyDavolio",Designation = "Marketing Executive",EmployeeID = 3,EmpID =
"EMP035",Address = "4110 Old Redmond Rd., Redmond",Contact = "(206) 555-
8122",Country = "USA",DOB = new DateTime(1966, 3, 19),ParentId = 1,Treedata
= new TreeData() { ID = 21 }});
        DataCollection.Add(new Employee { Name = "Andrew Fuller",FullName =
"AndrewFuller",Designation = "Sales Representative",EmployeeID = 4,EmpID =
"EMP045",Country = "UK",DOB = new DateTime(1980, 9, 20),ParentId =
1,Treedata = new TreeData() { ID = 21 }});
        return DataCollection;
    }
}
}

```


Name	Employee Image	Designation	Employee ID	Country
▼ Robert King		Chief Executive Officer	1	USA
David william		Vice President	2	USA
Nancy Davolio		Marketing Executive	3	USA
Andrew Fuller		Sales Representative	4	UK
▼ Anne Dodsworth		Sales Representative	5	USA
Michael Suyama		Sales Representative	6	UK

TreeGridTemplates component

If a component contains any **RenderFragment** type property then it does not allow any child components other than the render fragment property, which is [by design in Blazor](#).

This prevents us from directly specifying templates such as **RowTemplate** and **DetailTemplate** as descendant of the Tree Grid component. Hence the templates such as **RowTemplate** and **DetailTemplate** should be wrapped around a component named **TreeGridTemplates** as follows.

C#

```
@using TreeGridComponent.Data
@using Syncfusion.Blazor.TreeGrid;
@inject Microsoft.AspNetCore.Components.NavigationManager UriHelper
<SfTreeGrid ModelType="@model" Height="335" DataSource="@TreeData"
IdMapping="EmployeeID" ParentIdMapping="ParentId" TreeColumnIndex="0"
RowHeight="83" GridLines="@GridLine.Vertical">
<TreeGridTemplates>
<RowTemplate>
@{
var employee = (context as Employee);
<td style='padding-left:18px; border-bottom: 0.5px solid #e0e0e0;'>
<div>@employee.EmpID</div>
</td>
<td style='padding: 10px 0px 0px 20px; border-bottom: 0.5px solid #e0e0e0;'>
<div style="font-size:14px;">
@employee.FullName
</div>
</td>
<td style="border-bottom: 0.5px solid #e0e0e0;">
<div>
<div style="position:relative;display:inline-block;">
```

```


</div>
<div style="display:inline-block;">
<div style="padding:5px;">@employee.Address</div>
<div style="padding:5px;">@employee.Country</div>
<div style="padding:5px;font-size:12px;">@employee.Contact</div>
</div>
</div>
</td>
<td style='padding-left: 20px; border-bottom: 0.5px solid #e0e0e0;'>
<div>@employee.Designation</div>
</td>
}
</RowTemplate>
</TreeGridTemplates>
<TreeGridColumns>
<TreeGridColumn Field="EmpID" HeaderText="Employee ID"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Name" HeaderText="Employee Name"></TreeGridColumn>
<TreeGridColumn Field="Address" HeaderText="Employee Details" Width="340"
TextAlign="@TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Designation"
HeaderText="Designation"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public Employee model = new Employee();
public IEnumerable<Employee> TreeData { get; set; }
protected override void OnInitialized()
{
this.TreeData = Employee.GetTemplateData();
}
}

```

C#

```






namespace TreeGridComponent.Data {
public class Employee
{
public string Name { get; set; }
public DateTime? DOB { get; set; }
public string Designation { get; set; }
public string EmpID { get; set; }
public string Country { get; set; }
public int? ParentId { get; set; }
public TreeData Treedata { get; set; }
public static List<Employee> GetTemplateData()
{
List<Employee> DataCollection = new List<Employee>();
DataCollection.Add(new Employee { Name = "Robert King", Designation = "Chief
Executive Officer", EmpID = "EMP001", Country = "USA", DOB = new DateTime(1963,
2, 15), ParentId = null, Treedata = new TreeData() { ID = 21 } });
DataCollection.Add(new Employee { Name = "David william", Designation = "Vice
President", EmpID = "EMP004", Country = "USA", DOB = new DateTime(1971, 5,
20), ParentId = 1, Treedata = new TreeData() { ID = 21 } });
}
}

```

```

DataCollection.Add(new Employee { Name = "Nancy Davolio", Designation =
"Marketing Executive", EmpID = "EMP035", Country = "USA", DOB = new
DateTime(1966, 3, 19), ParentId = 1, Treedata = new TreeData() { ID = 21 } });
DataCollection.Add(new Employee { Name = "Andrew Fuller", Designation =
"Sales Representative", EmpID = "EMP045", Country = "UK", DOB = new
DateTime(1980, 9, 20), ParentId = 1, Treedata = new TreeData() { ID = 21 } });
return DataCollection;
}
}
}

```

Employee ID	Employee Name	Employee Details	Designation
▼ EMP001	RobertKing	507 - 20th Ave. E.Apt. 2A, Seattle  USA (206) 555-9857	Chief Executive Officer
EMP004	DavidWilliam	722 Moss Bay Blvd., Kirkland  USA (206) 555-3412	Vice President
EMP035	NancyDavolio	4110 Old Redmond Rd., Redmond  USA (206) 555-8122	Marketing Executive
EMP045	AndrewFuller	14 Garrett Hill, London  UK (71) 555-4848	Sales Representative
▼ EMP091	AnneDodsworth	4726 - 11th Ave. N.E., Seattle  USA (206) 555-1189	Sales Representative

Cell in Blazor TreeGrid Component

Displaying the HTML content

The HTML tags can be displayed in the Tree Grid header and content by enabling the [DisableHtmlEncode](#) property.

C#

```

@using TreeGridComponent.Data
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid IdMapping="TaskId" DataSource="@TreeGridData"
ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="<span> Task ID </span>"
Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="<span> Task Name </span>"
Width="160"></TreeGridColumn>

```

```

<TreeGridColumn Field="Duration" HeaderText="<span> Duration </span>"
DisableHtmlEncode="false" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="<span> Progress </span>"
DisableHtmlEncode="false" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public DateTime? StartDate { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public int? ParentId { get; set; }
}
public static List<BusinessObject> GetSelfDataSource()
{
List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", StartDate = new DateTime(2017, 10, 23), Duration =
10, Progress = 70, ParentId = null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", StartDate = new DateTime(2017, 10, 23), Duration = 4, Progress
= 80, ParentId = 1 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", StartDate = new DateTime(2017, 10, 24), Duration = 5, Progress
= 65, ParentId = 2 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", StartDate = new DateTime(2017, 10, 25), Duration = 6, Progress
= 77, ParentId = 3 });
return BusinessObjectCollection;
}
}
}

```

 Task ID 	 Task Name 	Duration	Progress
1	▼ Parent Task 1	10	70
2	▼ Child task 1	4	80
3	▼ Child Task 2	5	65
4	Child task 3	6	77
5	▼ Parent Task 2	10	70
6	Child task 1	4	80
7	Child Task 2	5	65
8	Child task 3	6	77
9	Child task 4	6	77

Customize cell styles

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.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridEvents QueryCellInfo="querycellinfo"
TValue="TreeData"></TreeGridEvents>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="90"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="100"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
<style>
.intro {
background-color:#336c12;
color:white;
```

```

}
.intro1 {
background-color:#7b2b1d;
color:white;
}
</style>
@code{
public List<TreeData> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
private void querycellinfo(QueryCellInfoEventArgs<TreeData> Args)
{
if (Args.Column.Field == "Progress" && Args.Data.Progress > 70 &&
Args.Data.Progress <= 100)
{
String[] s1 = new String[1] { "intro" };
Args.Cell.AddClass(s1);
}
else if (Args.Column.Field == "Progress" && Args.Data.Progress > 20)
{
String[] s2 = new String[1] { "intro1" };
Args.Cell.AddClass(s2);
}
}
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public DateTime? StartDate { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<TreeData> GetSelfDataSource()
{
List<TreeData> TreeDataCollection = new List<TreeData>();
TreeDataCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent Task
1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
TreeDataCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child task
1", Duration = 4, Progress = 80, Priority = "Low", ParentId = 1 });
TreeDataCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child Task
2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
TreeDataCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child task
3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
TreeDataCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent Task
2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
TreeDataCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child task
1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5 });
}
}
}

```

```

TreeDataCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
return TreeDataCollection;
}
}
}

```

Task ID	Task Name	Duration	Progress	Priority
1	▼ Parent Task 1	10	70	Critical
2	▼ Child task 1	50	80	Low
3	▼ Child Task 2	5	65	Critical
4	Child task 3	6	77	High
5	▼ Parent Task 2	10	70	Critical
6	Child task 1	4	80	Critical
7	Child Task 2	5	65	Low
8	Child task 3	6	77	High
9	Child task 4	6	77	Low

Auto wrap

The auto wrap allows the cell content of the tree grid to wrap to the next line when it exceeds the boundary of the cell width. The Cell Content wrapping works based on the position of white space between words. To enable auto wrap, set the [AllowTextWrap](#) property to **true**.

Note: When a column width is not specified, then auto wrap of columns will be adjusted with respect to the tree grid's width.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" AllowTextWrap="true">
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="100"></TreeGridColumn>

```

```

<TreeGridColumn Field="StartDate" HeaderText="Start Date" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="100"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public DateTime? StartDate { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
}
public static List<BusinessObject> GetSelfDataSource()
{
List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", StartDate = new DateTime(2017, 10, 23), Duration =
10, Progress = 70, Priority = "Critical", ParentId = null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", StartDate = new DateTime(2017, 10, 23), Duration = 4, Progress
= 80, Priority = "Low", ParentId = 1 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", StartDate = new DateTime(2017, 10, 24), Duration = 5, Progress
= 65, Priority = "Critical", ParentId = 2 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", StartDate = new DateTime(2017, 10, 25), Duration = 6, Priority
= "High", Progress = 77, ParentId = 3 });
return BusinessObjectCollection;
}
}
}

```


Task ID	Task Name	Start Date	Duration	Progress	Priority
1	Parent Task 1	10/23/2017	10	70	Critical
2	Child task 1	10/23/2017	4	80	Low
3	Child Task 2	10/24/2017	5	65	Critical
4	Child task 3	10/25/2017	6	77	High
5	Parent Task 2	10/23/2017	10	70	Critical
6	Child task 1	10/23/2017	4	80	Critical
7	Child Task 2	10/24/2017	5	65	Low
8	Child task 3	10/25/2017	6	77	High

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. |

C#

```
@using TreeGridComponent.Data
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid IdMapping="TaskId" ParentIdMapping="ParentId"
DataSource="@TreeGridData" TreeColumnIndex="1"
GridLines="Syncfusion.Blazor.Grids.GridLine.None">
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText="Start Date" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
```

```

<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="100"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public DateTime? StartDate { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
}
public static List<BusinessObject> GetSelfDataSource()
{
List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", StartDate = new DateTime(2017, 10, 23), Duration =
10, Progress = 70, Priority = "Critical", ParentId = null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", StartDate = new DateTime(2017, 10, 23), Duration = 4, Progress
= 80, Priority = "Low", ParentId = 1 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", StartDate = new DateTime(2017, 10, 24), Duration = 5, Progress
= 65, Priority = "Critical", ParentId = 2 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", StartDate = new DateTime(2017, 10, 25), Duration = 6, Priority
= "High", Progress = 77, ParentId = 3 });
return BusinessObjectCollection;
}
}
}

```

Task ID	Task Name	Start Date	Duration	Progress	Priority
1	▼ Parent Task 1	10/23/2017	10	70	Critical
2	▼ Child task 1	10/23/2017	4	80	Low
3	▼ Child Task 2	10/24/2017	5	65	Critical
4	Child task 3	10/25/2017	6	77	High
5	▼ Parent Task 2	10/23/2017	10	70	Critical
6	Child task 1	10/23/2017	4	80	Critical
7	Child Task 2	10/24/2017	5	65	Low
8	Child task 3	10/25/2017	6	77	High
9	Child task 4	10/25/2017	6	77	Low

By default, the tree grid renders with **Default** mode.

Clip Mode

The clip mode provides options to display its overflow cell content and it can be defined by the [ClipMode](#) property.

There are three types of [ClipMode](#). They are:

- **Clip**: Truncates the cell content when it overflows its area.
- **Ellipsis**: Displays ellipsis when the cell content overflows its area.
- **EllipsisWithTooltip**: Displays ellipsis when the cell content overflows its area, also it will display the tooltip while hover on ellipsis is applied.

C#

```
@using TreeGridComponent.Data
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid IdMapping="TaskId" ParentIdMapping="ParentId"
DataSource="@TreeGridData" TreeColumnIndex="1">
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name" Width="100"
ClipMode="Syncfusion.Blazor.Grids.ClipMode.EllipsisWithTooltip"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText="Start Date"
ClipMode="Syncfusion.Blazor.Grids.ClipMode.Ellipsis" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority" Width="60"
ClipMode="Syncfusion.Blazor.Grids.ClipMode.Clip"></TreeGridColumn>
</TreeGridColumn>
```

```

</SfTreeGrid>
@code{
public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
    this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
    public class BusinessObject
    {
        public int TaskId { get; set; }
        public string TaskName { get; set; }
        public DateTime? StartDate { get; set; }
        public int? Duration { get; set; }
        public int? Progress { get; set; }
        public string Priority { get; set; }
        public int? ParentId { get; set; }
    }
    public static List<BusinessObject> GetSelfDataSource()
    {
        List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
            "Parent Task 1", StartDate = new DateTime(2017, 10, 23), Duration =
            10, Progress = 70, Priority = "Critical", ParentId = null });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
            "Child task 1", StartDate = new DateTime(2017, 10, 23), Duration = 4, Progress
            = 80, Priority = "Low", ParentId = 1 });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
            "Child Task 2", StartDate = new DateTime(2017, 10, 24), Duration = 5, Progress
            = 65, Priority = "Critical", ParentId = 2 });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
            "Child task 3", StartDate = new DateTime(2017, 10, 25), Duration = 6, Priority
            = "High", Progress = 77, ParentId = 3 });
        return BusinessObjectCollection;
    }
}
}

```

Task ID	Task Name	Start Date	Duration	Progress	Priority
1	▼ Parent Task 1	Mon Oct...	10	70	Critical
2	▼ Child task 1	Mon Oct...	4	80	Low
3	▼ Child Task 2	Tue Oct ...	5	65	Critical
4	Child task 3	Wed Oct...	6	77	High
5	▼ Parent Task 2	Mon Oct...	10	70	Critical
6	Child task 1	Mon Oct...	4	80	Critical
7	Child Task 2	Tue Oct ...	5	65	Low
8	Child task 3	Wed Oct...	6	77	High
9	Child task 4	Wed Oct...	6	77	Low

By default, [ClipMode](#) value is **Ellipsis**.

Editing in Blazor TreeGrid Component

The Tree Grid component has options to dynamically insert, delete and update records. Editing feature is enabled by using the [TreeGridEditSettings](#) property and it requires a primary key column for CRUD operations.

To know more about editing feature in Blazor tree grid component, you can check on this video.

{% youtube

"youtube:https://www.youtube.com/watch?v=5_g3yr8ASys"%}

To define the primary key, set the [TreeGridColumn.IsPrimaryKey](#) to **true** in particular column.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" >
<TreeGridEditSettings AllowEditing="true"/>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" IsPrimaryKey="true"
Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
```

```

</TreeGridColumn>
</SfTreeGrid>
@code{
    public List<TreeData.BusinessObject> TreeGridData { get; set; }
    protected override void OnInitialized()
    {
        this.TreeGridData = TreeData.GetSelfDataSource().ToList();
    }
}

```

C#

```

namespace TreeGridComponent.Data {
    public class TreeData
    {
        public class BusinessObject
        {
            public int TaskId { get; set; }
            public string TaskName { get; set; }
            public int? Duration { get; set; }
            public int? Progress { get; set; }
            public string Priority { get; set; }
            public int? ParentId { get; set; }
        }
        public static List<BusinessObject> GetSelfDataSource()
        {
            List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
                "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
                null });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
                "Child task 1", Progress = 80, Priority = "Low", ParentId = 1 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
                "Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
            });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
                "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
                "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
                null });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
                "Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
                5 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
                "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
                "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
                "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
            return BusinessObjectCollection;
        }
    }
}

```

The editing can be disabled for a particular column, by specifying [TreeGridColumn.AllowEditing](#) to false.

To perform the editing operation in the Blazor Tree Grid, it is recommended to mention the respective model class type (List<TreeData.BusinessObject>), in the Tree Grid's dataSource property value.

CSHARP

```
@code{
    public List<TreeData.BusinessObject> TreeGridData { get; set; }
    protected override void OnInitialized()
    {
        this.TreeGridData = TreeData.GetSelfDataSource().ToList();
    }
}
```

Toolbar with edit option

The tree grid toolbar has the built-in items to execute Editing actions. It can be defined by using the [Toolbar](#) property.

C#






```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
    ParentIdMapping="ParentId"
    TreeColumnIndex="1" Toolbar="@ (new List<string>() { "Add", "Edit", "Delete",
    "Update", "Cancel" }) ">
    <TreeGridEditSettings AllowEditing="true" AllowAdding="true"
    AllowDeleting="true"/>
    <TreeGridColumns>
        <TreeGridColumn Field="TaskId" HeaderText="Task ID" IsPrimaryKey="true"
        Width="80"
        TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
        <TreeGridColumn Field="TaskName" HeaderText="Task Name"
        Width="160"></TreeGridColumn>
        <TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
        TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
        <TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
        TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
        <TreeGridColumn Field="Priority" HeaderText="Priority"
        Width="80"></TreeGridColumn>
    </TreeGridColumns>
</SfTreeGrid>
@code{
    public List<TreeData.BusinessObject> TreeGridData { get; set; }
    protected override void OnInitialized()
    {
        this.TreeGridData = TreeData.GetSelfDataSource().ToList();
    }
}
```

C#

```
namespace TreeGridComponent.Data {
    public class TreeData
    {
        public class BusinessObject
        {
```

```
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
}

public static List<BusinessObject> GetSelfDataSource()
{
    List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
    "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
    null });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
    "Child task 1", Progress = 80, Priority = "Low", ParentId = 1 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
    "Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
    });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
    "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
    "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
    null });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
    "Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
    5 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
    "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
    "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
    "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
    return BusinessObjectCollection;
}
}
```


 Add  Edit  Delete  Update  Cancel				
Task ID	Task Name	Duration	Progress	Priority
1	▼ Parent Task 1	10	70	Critical
2	▼ Child task 1	4	80	Low
3	▼ Child Task 2	5	65	Critical
4	Child task 3	6	77	High
5	▼ Parent Task 2	10	70	Critical
6	Child task 1	4	80	Critical
7	Child Task 2	5	65	Low
8	Child task 3	6	77	High
9	Child task 4	6	77	Low

Cell edit type and its params

The [TreeGridColumn.EditType](#) is used to customize the edit type of the particular column. The [TreeGridColumn.EditType](#) can be set based on the data type of the column.

- [NumericTextBox](#) component for integers, double, and decimal data types.
- [TextBox](#) component for string data type.
- [DropDownList](#) component for list data type.
- [DatePicker](#) component for date values.
- [DateTimePicker](#) component for datetime type.
- [Checkbox](#) component for boolean type.

Also, the model of the [TreeGridColumn.EditType](#) component can be customized through the [TreeGridColumn.Edit.params](#).

The following table describes cell edit type component and their corresponding edit params of the column.

Component | Example

[NumericTextBox](#) | `@(new { @params = new { format = "n" } })`

[TextBox](#) | -

[DropDownList](#) | `@(new { @params = new { value = "Germany" } })`

[DatePicker](#) | `@(new { @params = new { format = "yyyy-MM-dd" } })`

[DateTimePicker](#) | @(new { @params = new { strictMode = true } })

[Checkbox](#) | @(new { @params = new { checked = true } })

C#









```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" Toolbar="@ (new List<string>()
{ "Add", "Edit", "Delete", "Update", "Cancel" }) ">
<TreeGridEditSettings AllowEditing="true" AllowAdding="true"
AllowDeleting="true" />
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" IsPrimaryKey="true"
Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText="Start Date" Width="150"
Format="d" EditorSettings="DateParams"
Type="Syncfusion.Blazor.Grids.ColumnType.DateTime"
EditType="Syncfusion.Blazor.Grids.EditType.DateTimePickerEdit"></TreeGridCol
umn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
EditorSettings="NumericParams"
EditType="Syncfusion.Blazor.Grids.EditType.NumericEdit"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Approved" HeaderText="Approved"
Type="Syncfusion.Blazor.Grids.ColumnType.Boolean" DisplayAsCheckBox="true"
EditType="Syncfusion.Blazor.Grids.EditType.BooleanEdit"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public List<TreeData.BusinessObject> TreeGridData { get; set; }
public Syncfusion.Blazor.Grids.NumericEditCellParams NumericParams = new
Syncfusion.Blazor.Grids.NumericEditCellParams()
{
Params = new Syncfusion.Blazor.Inputs.NumericTextBoxModel<object>() { Format
= "N2" }
};
public Syncfusion.Blazor.Grids.DateEditCellParams DateParams = new
Syncfusion.Blazor.Grids.DateEditCellParams()
{
Params = new Syncfusion.Blazor.Calendars.DatePickerModel() { Format = "d" }
};
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}
```

C#

```

namespace TreeGridComponent.Data {
    public class TreeData
    {
        public class BusinessObject
        {
            public int TaskId { get; set; }
            public string TaskName { get; set; }
            public DateTime? StartDate { get; set; }
            public bool Approved { get; set; }
            public int? Duration { get; set; }
            public int? Progress { get; set; }
            public int? ParentId { get; set; }
        }
        public static List<BusinessObject> GetSelfDataSource()
        {
            List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
            "Parent Task 1", StartDate = new DateTime(2017, 10, 23), Duration =
            10, Progress = 70, ParentId = null });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
            "Child task 1", StartDate = new DateTime(2017, 10, 23), Progress = 80, ParentId
            = 1 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
            "Child Task 2", StartDate = new DateTime(2017, 10, 24), Duration = 5, Progress
            = 65, ParentId = 2 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
            "Child task 3", StartDate = new DateTime(2017, 10, 25), Duration = 6, Progress
            = 77, ParentId = 3 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
            "Parent Task 2", StartDate = new DateTime(2017, 10, 23), Duration =
            10, Progress = 70, ParentId = null });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
            "Child task 1", StartDate = new DateTime(2017, 10, 23), Duration = 4, Progress
            = 80, ParentId = 5 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
            "Child Task 2", StartDate = new DateTime(2017, 10, 24), Duration = 5, Progress
            = 65, ParentId = 5 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
            "Child task 3", StartDate = new DateTime(2017, 10, 25), Duration = 6, Progress
            = 77, ParentId = 5 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
            "Child task 4", StartDate = new DateTime(2017, 10, 25), Duration = 6, Progress
            = 77, ParentId = 5 });
            return BusinessObjectCollection;
        }
    }
}

```

 Add  Edit  Delete  Update  Cancel					
Task ID	Task Name	Start Date	Duration	Progress	Appr...
1	▼ Parent Task 1	10/23/2017	10	70	<input type="checkbox"/>
2	▼ Child task 1	10/23/   	4	80	<input type="checkbox"/>
3	▼ Child Task 2	10/24/2017	5	65	<input type="checkbox"/>
4	Child task 3	10/25/2017	6	77	<input type="checkbox"/>
5	▼ Parent Task 2	10/23/2017	10	70	<input type="checkbox"/>
6	Child task 1	10/23/2017	4	80	<input type="checkbox"/>
7	Child Task 2	10/24/2017	5	65	<input type="checkbox"/>
8	Child task 3	10/25/2017	6	77	<input type="checkbox"/>
9	Child task 4	10/25/2017	6	77	<input type="checkbox"/>

If edit type is not defined in the column, then it will be considered as the **stringedit** type (Textbox component).

Cell Edit Template

The cell edit template is used to add a custom component for a particular column when the column is edited. The following code example describes, how to define the Edit template for a particular column.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
@using Syncfusion.Blazor.Inputs;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" Toolbar="@ (new List<string>()
{ "Add", "Edit", "Delete", "Update", "Cancel" })">
<TreeGridEditSettings AllowEditing="true" AllowAdding="true"
AllowDeleting="true" />
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" IsPrimaryKey="true"
Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority" Width="80">
<EditTemplate>
@{
var task = context as TreeData;
```

```

}
<SfDropDownList ID="Priority" @bind-Value="task.Priority" TItem="string"
TValue="string" DataSource="@DropDownData"></SfDropDownList>
</EditTemplate>
</TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public List<TreeData> TreeGridData { get; set; }
public List<string> DropDownData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
this.DropDownData = TreeData.GetSelfDataSource().Select(s =>
s.Priority).Distinct().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<TreeData> GetSelfDataSource()
{
List<TreeData> TreeDataCollection = new List<TreeData>();
TreeDataCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent Task
1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
TreeDataCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child task
1", Progress = 80, Duration = 50, Priority = "Low", ParentId = 1 });
TreeDataCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child Task
2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
TreeDataCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child task
3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
TreeDataCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent Task
2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null});
TreeDataCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child task
1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5});
TreeDataCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child Task
2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5});
TreeDataCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child task
3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5});
TreeDataCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child task
4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5});
return TreeDataCollection;
}
}
}

```

+ Add ✎ Edit 🗑 Delete 💾 Update ✕ Cancel				
Task ID	Task Name	Duration	Progress	Priority
1	▼ Parent Task 1	10	70	Critical
2	▼ Child task 1	50	80	Low
3	▼ Child Task 2	5	65	Critical
4	Child task 3	6	77	Low
5	▼ Parent Task 2	10	70	High
6	Child task 1	4	80	Critical
7	Child Task 2	5	65	Low
8	Child task 3	6	77	High
9	Child task 4	6	77	Low

Edit Modes

Tree Grid supports the following types of edit modes, they are:

- Cell
- Row
- Dialog

Cell

In Cell edit mode, when a cell is double clicked, it is changed to edit state. The cell value can be changed and saved to the data source. To enable Cell edit, set the [TreeGridEditSettings.Mode](#) as **Cell**.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" Toolbar="@ (new List<string>()
{ "Add", "Edit", "Delete", "Update", "Cancel" })">
<TreeGridEditSettings AllowEditing="true" AllowAdding="true"
AllowDeleting="true" Mode="Syncfusion.Blazor.TreeGrid.EditMode.Cell" />
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" IsPrimaryKey="true"
Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
```

```

<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public DateTime? StartDate { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
}
public static List<BusinessObject> GetSelfDataSource()
{
List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", StartDate = new DateTime(2017, 10, 23), Duration =
10, Progress = 70, Priority = "Critical", ParentId = null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", StartDate = new DateTime(2017, 10, 23), Progress = 80, Priority
= "Low", ParentId = 1 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", StartDate = new DateTime(2017, 10, 24), Duration = 5, Progress
= 65, Priority = "Critical", ParentId = 2 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", StartDate = new DateTime(2017, 10, 25), Duration = 6, Priority
= "High", Progress = 77, ParentId = 3 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Parent Task 2", StartDate = new DateTime(2017, 10, 23), Duration =
10, Progress = 70, Priority = "Critical", ParentId = null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child task 1", StartDate = new DateTime(2017, 10, 23), Duration = 4, Progress
= 80, Priority = "Critical", ParentId = 5 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Child Task 2", StartDate = new DateTime(2017, 10, 24), Duration = 5, Progress
= 65, Priority = "Low", ParentId = 5 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child task 3", StartDate = new DateTime(2017, 10, 25), Duration = 6, Progress
= 77, Priority = "High", ParentId = 5 });
}
}

```

```
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child task 4", StartDate = new DateTime(2017, 10, 25), Duration = 6, Progress
= 77, Priority = "Low", ParentId = 5});
return BusinessObjectCollection;
}
}
}
```

+ Add ✎ Edit 🗑 Delete 💾 Update ✕ Cancel				
Task ID	Task Name	Duration	Progress	Priority
1	▼ Parent Task 1	10	70	Critical
2	▼ Child task 1	4	80	Low
3	Child Task 2	5	65	Critical
4	Child task 3	6	77	High
5	▼ Parent Task 2	10	70	Critical
6	Child task 1	4	80	Critical
7	Child Task 2	5	65	Low
8	Child task 3	6	77	High
9	Child task 4	6	77	Low

Cell edit mode is default mode of editing.

Row

In Row edit mode, when the currently selected record is edited, the entire row is changed to edit state. The cell values of the row can be changed and save edited data to the data source. To enable Row edit, set the [TreeGridEditSettings.Mode](#) as **Row**.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" Toolbar="@ (new List<string>()
{ "Add", "Edit", "Delete", "Update", "Cancel" })">
<TreeGridEditSettings AllowEditing="true" AllowAdding="true"
AllowDeleting="true" Mode="Syncfusion.Blazor.TreeGrid.EditMode.Row" />
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" IsPrimaryKey="true"
Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
```



```

<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
}
public static List<BusinessObject> GetSelfDataSource()
{
List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", Progress = 80, Priority = "Low", ParentId = 1 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
5 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
return BusinessObjectCollection;
}
}
}

```

```

}
}
}

```

+ Add ✎ Edit 🗑 Delete 📄 Update ✕ Cancel				
Task ID	Task Name	Duration	Progress	Priority
1	▼ Parent Task 1	10	70	Critical
2	Child task 1	4	80	Low
3	▼ Child Task 2	5	65	Critical
4	Child task 3	6	77	High
5	▼ Parent Task 2	10	70	Critical
6	Child task 1	4	80	Critical
7	Child Task 2	5	65	Low
8	Child task 3	6	77	High
9	Child task 4	6	77	Low

Dialog

In Dialog edit mode, while editing the currently selected row, data will be shown on a dialog. The cell values and save edited data can be changed to the data source. To enable Dialog edit, set the [TreeGridEditSettings.Mode](#) as **Dialog**.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" Toolbar="@ (new List<string>()
{ "Add", "Edit", "Delete", "Update", "Cancel" })">
<TreeGridEditSettings AllowEditing="true" AllowAdding="true"
AllowDeleting="true" Mode="Syncfusion.Blazor.TreeGrid.EditMode.Dialog" />
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" IsPrimaryKey="true"
Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>

```

```

</TreeGridColumn>
</SfTreeGrid>
@code{
    public List<TreeData.BusinessObject> TreeGridData { get; set; }
    protected override void OnInitialized()
    {
        this.TreeGridData = TreeData.GetSelfDataSource().ToList();
    }
}

```

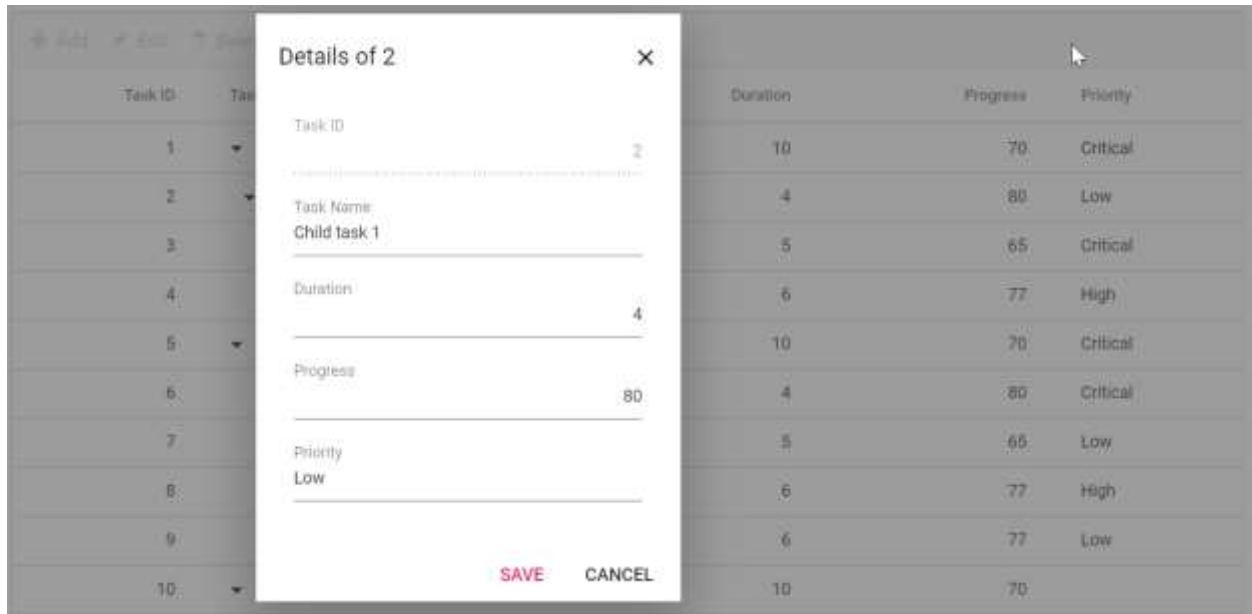
C#

```

namespace TreeGridComponent.Data {
    public class TreeData
    {
        public class BusinessObject
        {
            public int TaskId { get; set; }
            public string TaskName { get; set; }
            public int? Duration { get; set; }
            public int? Progress { get; set; }
            public string Priority { get; set; }
            public int? ParentId { get; set; }
        }
        public static List<BusinessObject> GetSelfDataSource()
        {
            List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
                "Parent Task 1", Duration = 10, Progress = 70, StartDate = new DateTime(2017,
                10, 23), Priority = "Critical", ParentId = null });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
                "Child task 1", Progress = 80, Priority = "Low", StartDate = new DateTime(2017,
                10, 23), ParentId = 1 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
                "Child Task 2", Duration = 5, Progress = 65, StartDate = new DateTime(2017, 10,
                23), Priority = "Critical", ParentId = 2 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
                "Child task 3", Duration = 6, Priority = "High", StartDate = new DateTime(2017,
                10, 23), Progress = 77, ParentId = 3 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
                "Parent Task 2", Duration = 10, Progress = 70, StartDate = new DateTime(2017,
                10, 23), Priority = "Critical", ParentId = null });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
                "Child task 1", Duration = 4, Progress = 80, StartDate = new DateTime(2017, 10,
                23), Priority = "Critical", ParentId = 5 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
                "Child Task 2", Duration = 5, Progress = 65, StartDate = new DateTime(2017, 10,
                23), Priority = "Low", ParentId = 5 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
                "Child task 3", Duration = 6, Progress = 77, StartDate = new DateTime(2017, 10,
                23), Priority = "High", ParentId = 5 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
                "Child task 4", Duration = 6, Progress = 77, StartDate = new DateTime(2017, 10,
                23), Priority = "Low", ParentId = 5 });
            return BusinessObjectCollection;
        }
    }
}

```

```
}
}
```



Batch

In the Batch edit mode, when the tree grid cell is double-clicked, the target cell goes into edit state. It can be bulk saved (added, changed and deleted data in the single request) to the data source by clicking on the toolbar's **Update** button or by externally invoking the [EndEdit](#) method.

To enable Batch edit, set the [Mode](#) property of the `TreeGridEditSettings` as **Batch**.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" AllowPaging="true"
Toolbar="@ (new List<string>() { "Add", "Delete", "Update", "Cancel" })">
<TreeGridPageSettings PageSize="2"></TreeGridPageSettings>
<TreeGridEditSettings AllowAdding="true" AllowEditing="true"
AllowDeleting="true" Mode="EditMode.Batch"
NewRowPosition="RowPosition.Below"></TreeGridEditSettings>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="60"
IsPrimaryKey="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="155"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText="Start Date" Format="d"
Type=Syncfusion.Blazor.Grids.ColumnType.Date Width="85"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
EditType=Syncfusion.Blazor.Grids.EditType.DatePickerEdit></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="70"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="70"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
```

```

<TreeGridColumn Field="Priority" HeaderText="Priority" Width="70"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public List<BusinessObject> TreeData { get; set; }
protected override void OnInitialized()
{
this.TreeData = BusinessObject.GetSelfDataSource().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public DateTime? StartDate { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<BusinessObject> GetSelfDataSource()
{
List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", StartDate = new DateTime(2017, 10, 23), Duration =
10, Progress = 70, Priority = "Critical", ParentId = null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", StartDate = new DateTime(2017, 10, 23), Duration = 5, Progress
= 80, Priority = "Low", ParentId = 1 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", StartDate = new DateTime(2017, 10, 24), Duration = 5, Progress
= 65, Priority = "Critical", ParentId = 2 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", StartDate = new DateTime(2017, 10, 25), Duration = 6, Priority
= "High", Progress = 77, ParentId = 3 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Child Task 5", StartDate = new DateTime(2017, 10, 26), Duration = 9,
Progress = 25, ParentId = 4, Priority = "Normal" });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child Task 6", StartDate = new DateTime(2017, 10, 27), Duration = 9,
Progress = 7, ParentId = 5, Priority = "Normal" });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Parent Task 3", StartDate = new DateTime(2017, 10, 28), Duration = 4,
Progress = 45, ParentId = null, Priority = "High" });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child Task 7", StartDate = new DateTime(2017, 10, 29), Duration = 3,
Progress = 38, ParentId = 7, Priority = "Critical" });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child Task 8", StartDate = new DateTime(2017, 10, 30), Duration = 7,
Progress = 70, ParentId = 7, Priority = "Low" });
return BusinessObjectCollection;
}
}

```

```
}
}
```

Dialog template

To know about customizing the Dialog Template in Blazor tree grid component, you can check this video.

{% youtube

"youtube:https://www.youtube.com/watch?v=TxHrtyVwY4A"%}

The dialog template editing provides an option to customize the default behavior of dialog editing. Using the dialog template, render your own editors by defining the [TreeGridEditSettings.Mode](#) as **Dialog** and [Template](#) using the **Template** of the **TreeGridEditSettings**.

In some cases, the new field editors must be added in the dialog which are not present in the column model. In that situation, the dialog template will help to customize the default edit dialog.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
@using Syncfusion.Blazor.DropDowns;
<SfTreeGrid DataSource="@TreeGridData" AllowPaging="true" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" Toolbar="@ (new List<string>()
{ "Add", "Edit", "Delete", "Update", "Cancel" }) ">
<TreeGridEvents TValue="TreeData"
OnActionComplete="OnComplete"></TreeGridEvents>
<TreeGridEditSettings AllowEditing="true" AllowAdding="true"
AllowDeleting="true" Mode="Syncfusion.Blazor.TreeGrid.EditMode.Dialog"
NewRowPosition="RowPosition.Child">
<Template>
@{
var employee = (context as TreeData);
}
<div>
<div class="form-row">
<div class="form-group col-md-6">
<SfNumericTextBox ID="TaskId" @bind-Value="@ (employee.TaskId) "
Enabled="@Check" FloatLabelType="FloatLabelType.Always" Placeholder="Task
ID"></SfNumericTextBox>
</div>
<div class="form-group col-md-6">
<SfAutoComplete TItem="TreeData" ID="TaskName" @bind-
Value="@ (employee.TaskName) " TValue="string" DataSource="@TreeGridData"
FloatLabelType="FloatLabelType.Always" Placeholder="Task Name">
<AutoCompleteFieldSettings Value="TaskName"></AutoCompleteFieldSettings>
</SfAutoComplete>
</div>
</div>
<div class="form-row">
<div class="form-group col-md-6">
<SfNumericTextBox ID="Duration" @bind-Value="@ (employee.Duration) "
TValue="int?" FloatLabelType="FloatLabelType.Always"
Placeholder="Duration"></SfNumericTextBox>
</div>
<div class="form-group col-md-6">
```

```

<SfNumericTextBox ID="Progress" @bind-Value="@ (employee.Progress) "
TValue="int?" FloatLabelType="FloatLabelType.Always"
Placeholder="Progress"></SfNumericTextBox>
</div>
</div>
<div class="form-row">
<div class="form-group col-md-6">
<SfDropDownList ID="Priority" TItem="TreeData" @bind-
Value="@ (employee.Priority) " TValue="string" DataSource="@TreeGridData"
FloatLabelType="FloatLabelType.Always" Placeholder="Priority">
<DropDownListFieldSettings Value="Priority"
Text="Priority"></DropDownListFieldSettings>
</SfDropDownList>
</div>
</div>
</div>
</Template>
</TreeGridEditSettings>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" IsPrimaryKey="true"
Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
private Boolean Check = false;
public List<TreeData> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
private void OnComplete(ActionEventArgs<TreeData> args)
{
if (args.RequestType.ToString() == "Add")
{
Check = true;
}
else
{
Check = false;
}
}
}
}

```

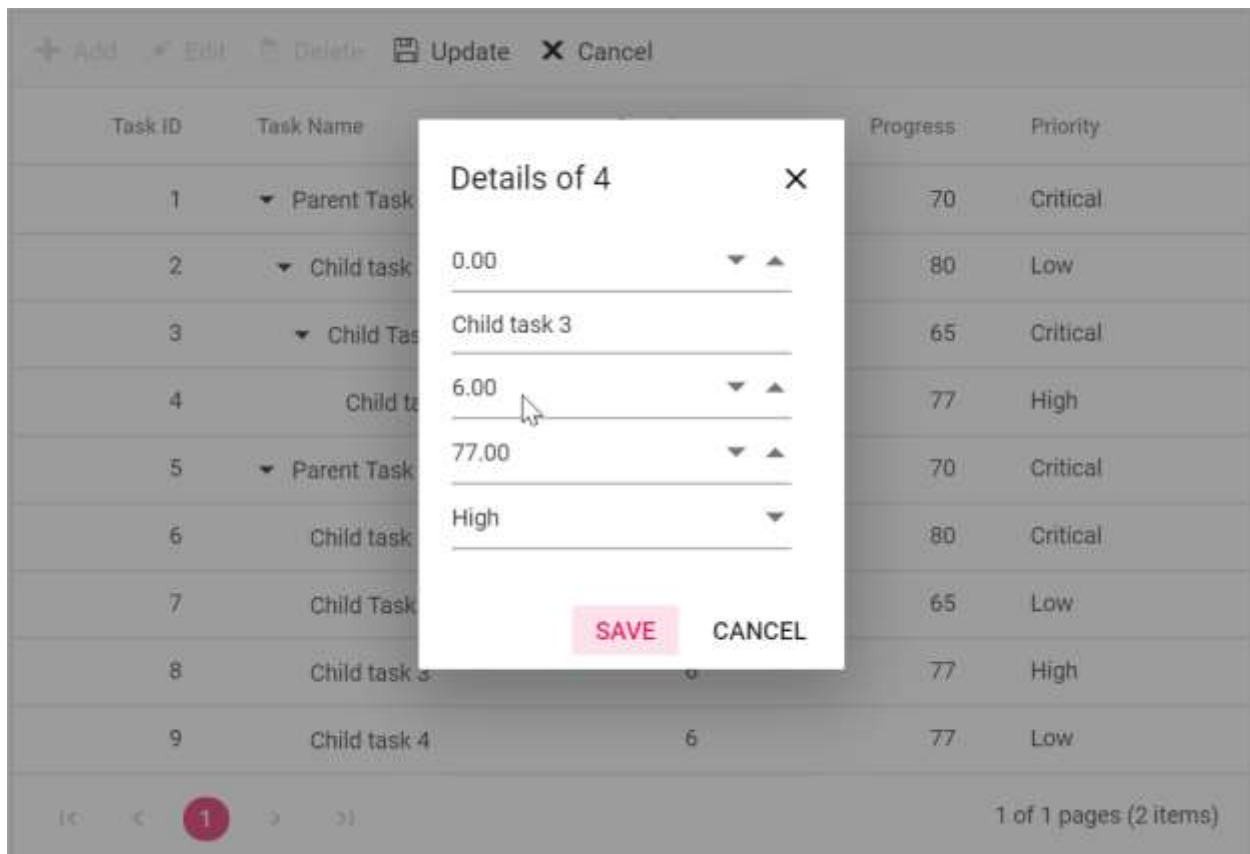
C#

```

namespace TreeGridComponent.Data {
public class TreeData

```

```
{
    public int TaskId { get; set; }
    public string TaskName { get; set; }
    public int? Duration { get; set; }
    public int? Progress { get; set; }
    public string Priority { get; set; }
    public int? ParentId { get; set; }
    public static List<TreeData> GetSelfDataSource()
    {
        List<TreeData> TreeDataCollection = new List<TreeData>();
        TreeDataCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
        TreeDataCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child task 1", Progress = 80, Priority = "Low", Duration = 50, ParentId = 1 });
        TreeDataCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
        TreeDataCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
        TreeDataCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
        TreeDataCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5 });
        TreeDataCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
        TreeDataCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
        TreeDataCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
        return TreeDataCollection;
    }
}
```

The template form editors should have **name** attribute.

Adding row position

The Tree Grid control provides the support to add the new row in the top, bottom, above selected row, below selected row and child position of tree grid content using the [TreeGridEditSettings.NewRowPosition](#) property. By default, a new row will be added at the top of the tree grid.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" Toolbar="@ (new List<string>()
{ "Add", "Edit", "Delete", "Update", "Cancel" })">
<TreeGridEditSettings AllowEditing="true" AllowAdding="true"
AllowDeleting="true" NewRowPosition="RowPosition.Child" />
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" IsPrimaryKey="true"
Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
```

```

<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
}
public static List<BusinessObject> GetSelfDataSource()
{
List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", Progress = 80, Priority = "Low", ParentId = 1 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5});
return BusinessObjectCollection;
}
}
}

```

The following examples shows how to set new row position as **Child** in tree grid.

Command column

The command column provides an option to add CRUD action buttons in a column. This can be defined by the [TreeGridColumn.Commands](#) property.

The available built-in command buttons are:

Command Button	Actions
Edit	Edit the current row.
Delete	Delete the current row.
Save	Update the edited row.
Cancel	Cancel the edited state.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
@using Syncfusion.Blazor.Grids;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" Toolbar="@ (new List<string>()
{ "Add", "Edit", "Delete", "Update", "Cancel" }) ">
<TreeGridEditSettings AllowEditing="true" AllowAdding="true"
AllowDeleting="true" Mode="Syncfusion.Blazor.TreeGrid.EditMode.Row" />
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" IsPrimaryKey="true"
Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn HeaderText="Manage Records" Width="150">
<TreeGridCommandColumns>
<TreeGridCommandColumn Type="CommandButtonType.Edit" ButtonOption="@ (new
CommandButtonOptions() { IconCss="e-icons e-edit", CssClass="e-flat"
}) "></TreeGridCommandColumn>
<TreeGridCommandColumn Type="CommandButtonType.Delete" ButtonOption="@ (new
CommandButtonOptions() { IconCss="e-icons e-delete", CssClass="e-flat"
}) "></TreeGridCommandColumn>
<TreeGridCommandColumn Type="CommandButtonType.Save" ButtonOption="@ (new
CommandButtonOptions() { IconCss="e-icons e-save", CssClass="e-flat"
}) "></TreeGridCommandColumn>
<TreeGridCommandColumn Type="CommandButtonType.Cancel" ButtonOption="@ (new
CommandButtonOptions() { IconCss="e-icons e-cancel-icon", CssClass="e-flat"
}) "></TreeGridCommandColumn>
</TreeGridCommandColumns>
</TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
```

```

public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
    this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}
























```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
    public class BusinessObject
    {
        public int TaskId { get; set; }
        public string TaskName { get; set; }
        public int? Duration { get; set; }
        public int? Progress { get; set; }
        public int? ParentId { get; set; }
    }
    public static List<BusinessObject> GetSelfDataSource()
    {
        List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", Duration = 10, Progress = 70, ParentId = null });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", Progress = 80, ParentId = 1 });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", Duration = 5, Progress = 65, ParentId = 2 });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", Duration = 6, Progress = 77, ParentId = 3 });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Parent Task 2", Duration = 10, Progress = 70, ParentId = null });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child task 1", Duration = 4, Progress = 80, ParentId = 5 });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Child Task 2", Duration = 5, Progress = 65, ParentId = 5 });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child task 3", Duration = 6, Progress = 77, ParentId = 5 });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child task 4", Duration = 6, Progress = 77, ParentId = 5 });
        return BusinessObjectCollection;
    }
}
}

```

 Add  Edit  Delete  Update  Cancel					
Task ID	Task Name	Duration	Progress	Manage Records	
1	▼ Parent Task 1	10	70		
2	▼ Child task 1	4	80		
3	▼ Child Task 2	5	65		
4	Child task 3	6	77		
5	▼ Parent Task 2	10	70		
6	Child task 1	4	80		
7	Child Task 2	5	65		
8	Child task 3	6	77		
9	Child task 4	6	77		

Custom command

The custom command buttons can be added in a column by using the [Commands](#) property of the [TreeGridColumn](#) component and the action for the custom buttons can be defined in the [CommandClicked](#) event.

The following sample code demonstrates adding custom command in the **Manage Records** column and the [CommandClicked](#) event which triggers when the command is clicked,

ASPX-CS

```
@using Syncfusion.Blazor.TreeGrid;
@using Syncfusion.Blazor.Grids;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" Toolbar="@ (new List<string>()
{ "Add", "Edit", "Delete", "Update", "Cancel" })">
<TreeGridEvents CommandClicked="OnCommandClicked"
TValue="TreeData.BusinessObject"></TreeGridEvents>
<TreeGridEditSettings AllowEditing="true" AllowAdding="true"
AllowDeleting="true" Mode="Syncfusion.Blazor.TreeGrid.EditMode.Row" />
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" IsPrimaryKey="true"
Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
```

```

<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn HeaderText="Manage Records" Width="150">
<TreeGridCommandColumns>
<TreeGridCommandColumn ButtonOption="@ (new CommandButtonOptions() { Content
= "Details", CssClass = "e-flat" })"></TreeGridCommandColumn>
</TreeGridCommandColumns>
</TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public class TreeData
{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public int? ParentId { get; set; }
}
public static List<BusinessObject> GetSelfDataSource()
{
List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", Duration = 10, Progress = 70, ParentId = null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", Progress = 80, ParentId = 1 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", Duration = 5, Progress = 65, ParentId = 2 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", Duration = 6, Progress = 77, ParentId = 3 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Parent Task 2", Duration = 10, Progress = 70, ParentId = null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child task 1", Duration = 4, Progress = 80, ParentId = 5 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Child Task 2", Duration = 5, Progress = 65, ParentId = 5 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child task 3", Duration = 6, Progress = 77, ParentId = 5 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child task 4", Duration = 6, Progress = 77, ParentId = 5 });
return BusinessObjectCollection;
}
}
public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
public void OnCommandClicked(CommandClickEventArgs<TreeData.BusinessObject>
args)
{
// Perform required operations here
}
}

```

The following image represents the custom command added in the **Manage Records** column of the Tree Grid component,

+ Add Edit Delete Update Cancel				
Task ID	Task Name	Duration	Progress	Manage Records
1	▼ Parent Task 1	10	70	<button>Details</button>
2	▼ Child task 1		80	<button>Details</button>
3	▼ Child Task 2	5	65	<button>Details</button>
4	Child task 3	6	77	<button>Details</button>
5	▼ Parent Task 2	10	70	<button>Details</button>
6	Child task 1	4	80	<button>Details</button>
7	Child Task 2	5	65	<button>Details</button>
8	Child task 3	6	77	<button>Details</button>
9	Child task 4	6	77	<button>Details</button>

Confirmation messages

Delete confirmation

The delete confirm dialog can be shown when deleting a record by defining the [ShowDeleteConfirmDialog](#) as **true**

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" Toolbar="@ (new List<string>()
{ "Add", "Edit", "Delete", "Update", "Cancel" })">
<TreeGridEditSettings AllowEditing="true" AllowAdding="true"
AllowDeleting="true" ShowDeleteConfirmDialog="true"/>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" IsPrimaryKey="true"
Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
</TreeGridColumns>
```

```

</SfTreeGrid>
@code{
public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
    this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
    public class BusinessObject
    {
        public int TaskId { get; set; }
        public string TaskName { get; set; }
        public int? Duration { get; set; }
        public int? Progress { get; set; }
        public int? ParentId { get; set; }
    }
    public static List<BusinessObject> GetSelfDataSource()
    {
        List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", Duration = 10, Progress = 70, ParentId = null });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", Progress = 80, ParentId = 1 });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", Duration = 5, Progress = 65, ParentId = 2 });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", Duration = 6, Progress = 77, ParentId = 3 });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Parent Task 2", Duration = 10, Progress = 70, ParentId = null });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child task 1", Duration = 4, Progress = 80, ParentId = 5 });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Child Task 2", Duration = 5, Progress = 65, ParentId = 5 });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child task 3", Duration = 6, Progress = 77, ParentId = 5 });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child task 4", Duration = 6, Progress = 77, ParentId = 5 });
        return BusinessObjectCollection;
    }
}
}

```


+ Add ✎ Edit 🗑 Delete 🔄 Update ✕ Cancel				
Task ID	Task Name	Duration	Progress	
1	▼ Parent Task 1	10	70	
2	▼ Child task 1	4	80	
3			65	
4			77	
5			70	
6	Child task 1	4	80	
7	Child Task 2	5	65	
8	Child task 3	6	77	
9	Child task 4	6	77	

Are you sure you want to Delete Record?

OK

CANCEL

The **ShowDeleteConfirmDialog** supports all type of edit modes.

Entity Framework

This section uses and follows the code explained in the [Entity Framework data binding](#) section hence it is recommended to refer Entity Framework data binding section before continue this section.

Handle CRUD in data access layer class

Now add methods **AddTask**, **UpdateTask**, **DeleteTask** in the “**TasksDataAccessLayer.cs**” to handle the insert, update and remove operations respectively. **CRUD** record details are bound to the **Tasks** parameter.

CSHARP

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using TreeGridWebApiEFSample.Shared.Models;
using Microsoft.EntityFrameworkCore;
namespace TreeGridWebApiEFSample.Shared.DataAccess
{
    public class TasksDataAccessLayer
    {
        TasksContext treedb = new TasksContext();
        //To Get all Task details
        public IEnumerable<Shared.Models.Task> GetAllRecords()
        {
```

```
try
{
return treedb.Tasks.ToList();
}
catch
{
throw;
}
}
//To Add new Tasks record
public void AddTask(Shared.Models.Task task)
{
try
{
treedb.Tasks.Add(task);
treedb.SaveChanges();
}
catch
{
throw;
}
}
//To Update the records of a particular Tasks
public void UpdateTask(Shared.Models.Task task)
{
try
{
treedb.Entry(task).State = EntityState.Modified;
treedb.SaveChanges();
}
catch
{
throw;
}
}
//Get the details of a particular Tasks
public Shared.Models.Task GetTaskData(int id)
{
try
{
Shared.Models.Task task = treedb.Tasks.Find(id);
return task;
}
catch
{
throw;
}
}
//To Delete the record of a particular Tasks
public void DeleteTask(int id)
{
try
{
Shared.Models.Task emp = treedb.Tasks.Find(id);
treedb.Tasks.Remove(emp);
treedb.SaveChanges();
}
}
```

```

catch
{
    throw;
}
}
}
}

```

Enable CRUD in Web API

Now, create a new **Post, Put, Delete** method in the Web API controller which will perform the CRUD operations and returns the appropriate resultant data. The '**SfDataManager**' will make requests to this action based on route name.

C# SHARP

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using TreeGridWebApiEFSSample.Shared.Models;
using TreeGridWebApiEFSSample.Shared.DataAccess;
using Microsoft.Extensions.Primitives;
using System.Web;
using Microsoft.AspNetCore.Http;
using Newtonsoft.Json.Linq;
namespace TreeGridWebApiEFSSample.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class TreeGridController : ControllerBase
    {
        TasksDataAccessLayer db = new TasksDataAccessLayer();
        // GET: api/<TreeGridController>
        [HttpGet]
        public object Get()
        {
            var queryString = Request.Query;
            IQueryable<TreeGridWebApiEFSSample.Shared.Models.Task> data1 =
            db.GetAllRecords().AsQueryable();
            if (queryString.Keys.Contains("$filter") &&
            !queryString.Keys.Contains("$top"))
            {
                StringValues filter;
                queryString.TryGetValue("$filter", out filter);
                int fltr = Int32.Parse(filter[0].ToString().Split("eq")[1]);
                data1 = data1.Where(f => f.ParentID == fltr).AsQueryable();
                return new { Items = data1, Count = data1.Count() };
            }
            if (queryString.Keys.Contains("$select"))
            {
                data1 = (from ord in data1
                select new TreeGridWebApiEFSSample.Shared.Models.Task
                {
                    ParentID = ord.ParentID
                })
            }
        }
    }
}

```

```

);
return data1;
}
data1 = data1.Where(p => p.ParentID == null);
var count = data1.Count();
if (queryString.Keys.Contains("$inlinecount"))
{
    StringValues Skip;
    StringValues Take;
    int skip = (queryString.TryGetValue("$skip", out Skip)) ?
    Convert.ToInt32(Skip[0]) : 0;
    int top = (queryString.TryGetValue("$top", out Take)) ?
    Convert.ToInt32(Take[0]) : data1.Count();
    return new { Items = data1.Skip(skip).Take(top), Count = count };
}
else
{
    return data1;
}
}
// GET api/<TreeGridController>/5
[HttpGet("{id}")]
public string Get(int id)
{
    return "value";
}
[HttpPost]
public object Post([FromBody]TreeGridWebApiEFSample.Shared.Models.Task Task)
{
    IQueryable<TreeGridWebApiEFSample.Shared.Models.Task> data1 =
    db.GetAllRecords().AsQueryable();
    var i = 0;
    // For loop for finding the parent record and setting its
    "HasChildMapping"(here it is "IsParent") to true
    for (; i < data1.ToList().Count; i++)
    {
        if (data1.ToList()[i].TaskID == Task.ParentID)
        {
            if (data1.ToList()[i].IsParent == null)
            {
                data1.ToList()[i].IsParent = true;
            }
            break;
        }
    }
    db.AddTask(Task);
    return Task;
}
public int FindChildRecords(int? id)
{
    var count = 0;
    IQueryable<TreeGridWebApiEFSample.Shared.Models.Task> data1 =
    db.GetAllRecords().AsQueryable();
    for (var i = 0; i < data1.ToList().Count; i++)
    {
        if (data1.ToList()[i].ParentID == id)
        {

```

```

count++;
count += FindChildRecords(data1.ToList()[i].TaskID);
}
}
return count;
}

[HttpPut]
public object Put([FromBody]TreeGridWebApiEFSample.Shared.Models.Task Task)
{
    IQueryable<TreeGridWebApiEFSample.Shared.Models.Task> data2 =
    db.GetAllRecords().AsQueryable();
    TreeGridWebApiEFSample.Shared.Models.Task val = data2.Where(or => or.TaskID
    == Task.TaskID).FirstOrDefault();
    val.TaskID = Task.TaskID;
    val.TaskName = Task.TaskName;
    val.Duration = Task.Duration;
    val.Progress = Task.Progress;
    db.UpdateTask(val);
    return val;
}

[HttpDelete("{id}")]
public void Delete(int id)
{
    db.DeleteTask(id);
}
}
}

```

Configure the Tree Grid to perform CRUD operations

ASPX-CS

```

@using Syncfusion.Blazor.Grids
@using Syncfusion.Blazor.Data
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.TreeGrid
@using Shared.Models

<SfTreeGrid TValue="Shared.Models.Task" @ref="treeGrid" IdMapping="TaskID"
AllowPaging="true"
ParentIdMapping="ParentID" HasChildMapping="IsParent"
TreeColumnIndex="0" Toolbar="ToolbarItems">
<SfDataManager Url="api/TreeGrid" Adaptor="Adaptors.WebApiAdaptor"
CrossDomain="true"></SfDataManager>
<TreeGridEditSettings AllowEditing="true"
AllowAdding="true"
AllowDeleting="true"
Mode="Syncfusion.Blazor.TreeGrid.EditMode.Row"
NewRowPosition="Syncfusion.Blazor.TreeGrid.RowPosition.Child"></TreeGridEdit
Settings>
<TreeGridColumns>
<TreeGridColumn Field="TaskID" HeaderText="Task ID" Width="80"
IsPrimaryKey="true" Type=ColumnType.Number></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name" Width="160"
Type=ColumnType.String></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="160"
Type=ColumnType.Number></TreeGridColumn>

```

```

<TreeGridColumn Field="Progress" HeaderText="Progress" Width="160"
Type=ColumnType.Number></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid<Shared.Models.Task> treeGrid { get; set; }
private List<Object> ToolbarItems = new List<Object>() { "Add", "Edit",
"Delete", "Update", "Cancel" };
}

```

You can find the fully working sample [here](#).

Default column values on add new

The tree grid provides an option to set the default value for the columns when adding a new record in it. To set a default value for the particular column by defining the [TreeGridColumn.DefaultValue](#).

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" Toolbar="@ (new List<string>()
{ "Add", "Edit", "Delete", "Update", "Cancel" })">
<TreeGridEditSettings AllowEditing="true" AllowAdding="true"
AllowDeleting="true" ShowDeleteConfirmDialog="true" />
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" IsPrimaryKey="true"
Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
DefaultValue="@DefaultVal"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public List<TreeData.BusinessObject> TreeGridData { get; set; }
public int DefaultVal { get; set; } = 45;
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
}
}
}

```

```

public int? Duration { get; set; }
public int? Progress { get; set; }
public int? ParentId { get; set; }
}
public static List<BusinessObject> GetSelfDataSource()
{
    List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
    "Parent Task 1", Duration = 10, Progress = 70, ParentId = null });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
    "Child task 1", Progress = 80, ParentId = 1 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
    "Child Task 2", Duration = 5, Progress = 65, ParentId = 2 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
    "Child task 3", Duration = 6, Progress = 77, ParentId = 3 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
    "Parent Task 2", Duration = 10, Progress = 70, ParentId = null });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
    "Child task 1", Duration = 4, Progress = 80, ParentId = 5 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
    "Child Task 2", Duration = 5, Progress = 65, ParentId = 5 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
    "Child task 3", Duration = 6, Progress = 77, ParentId = 5 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
    "Child task 4", Duration = 6, Progress = 77, ParentId = 5 });
    return BusinessObjectCollection;
}
}
}

```

Disable editing for particular column

The editing can be disabled for particular columns by using the [TreeGridColumn.AllowEditing](#).

In the following demo, editing is disabled for the **Duration** column.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" Toolbar="@ (new List<string>()
{ "Add", "Edit", "Delete", "Update", "Cancel" })">
<TreeGridEditSettings AllowEditing="true" AllowAdding="true"
AllowDeleting="true" ShowDeleteConfirmDialog="true" />
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" IsPrimaryKey="true"
Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
AllowEditing="false"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
</TreeGridColumn>

```

```

</SfTreeGrid>
@code{
public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
    this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
    public class BusinessObject
    {
        public int TaskId { get; set; }
        public string TaskName { get; set; }
        public int? Duration { get; set; }
        public int? Progress { get; set; }
        public int? ParentId { get; set; }
    }
    public static List<BusinessObject> GetSelfDataSource()
    {
        List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", Duration = 10, Progress = 70, ParentId = null });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", Progress = 80, ParentId = 1 });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", Duration = 5, Progress = 65, ParentId = 2 });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", Duration = 6, Progress = 77, ParentId = 3 });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Parent Task 2", Duration = 10, Progress = 70, ParentId = null });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child task 1", Duration = 4, Progress = 80, ParentId = 5 });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Child Task 2", Duration = 5, Progress = 65, ParentId = 5 });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child task 3", Duration = 6, Progress = 77, ParentId = 5 });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child task 4", Duration = 6, Progress = 77, ParentId = 5 });
        return BusinessObjectCollection;
    }
}
}

```

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 tree grid, then edit or delete action take places the first row.

Filtering in Blazor TreeGrid Component

Filtering allows to view specific or related records based on the filter criteria. To enable filtering in the Tree Grid, set the [AllowFiltering](#) to true. Filtering options can be configured through the [FilterSettings](#).

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
@using Syncfusion.Blazor.Data;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" AllowFiltering="true">
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText="Start Date" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="60"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
public List<BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = BusinessObject.GetSelfDataSource().ToList();
}
}
```

C#

```
namespace TreeGridComponent.Data {
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<BusinessObject> GetSelfDataSource()
{
List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", Duration = 4, Progress = 80, Priority = "Low", ParentId = 1 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
});
}
```

```

BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5});
return BusinessObjectCollection;
}
}
}

```

Task ID	Task Name	Start Date	Duration	Progress	Priority
1	▼ Parent Task 1	Mon Oct 23 2...	10	70	Critical
2	▼ Child task 1	Mon Oct 23 2...	4	80	Low
3	▼ Child Task 2	Tue Oct 24 20...	5	65	Critical
4	Child task 3	Wed Oct 25 2...	6	77	High
5	▼ Parent Task 2	Mon Oct 23 2...	10	70	Critical
6	Child task 1	Mon Oct 23 2...	4	80	Critical
7	Child Task 2	Tue Oct 24 20...	5	65	Low
8	Child task 3	Wed Oct 25 2...	6	77	High
9	Child task 4	Wed Oct 25 2...	6	77	Low

* Apply and clear filtering by using the [FilterByColumn](#) and [ClearFiltering](#) methods.

* To disable filtering for a particular column, set the [AllowFiltering](#) property of [Column](#) to false.

Filter hierarchy modes

Tree Grid provides support for a set of filtering modes with [HierarchyMode](#) of [FilterSettings](#) property. The below are the types of filter mode available in the Tree Grid.

- **Parent** : This is the default filter hierarchy mode in the Tree Grid. The filtered records are displayed with its parent records, if the filtered records not have any parent record then the filtered records are only displayed.
- **Child** : The filtered records are displayed with its child record, if the filtered records does not have any child record then the filtered records are only displayed.

- **Both** : The filtered records are displayed with its both parent and child record, if the filtered records does not have any parent and child record then the filtered records are only displayed.
- **None** : The filtered records are only displayed.

Initial filter

To apply the filter at initial rendering, set the filter **PredicateModel** in [Columns](#) property of the [FilterSettings](#).

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
@using Syncfusion.Blazor.Data;
@using Syncfusion.Blazor.Grids;
<SfTreeGrid IdMapping="TaskId" DataSource="@TreeGridData"
AllowFiltering="true" ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridFilterSettings>
<TreeGridFilterColumns>
<TreeGridFilterColumn Field="TaskName" MatchCase="false"
Operator="Syncfusion.Blazor.Operator.StartsWith" Predicate="and"
Value="@Name"></TreeGridFilterColumn>
<TreeGridFilterColumn Field="Duration" MatchCase="false"
Operator="Syncfusion.Blazor.Operator.Equal" Predicate="and"
Value="@Duration"></TreeGridFilterColumn>
</TreeGridFilterColumns>
</TreeGridFilterSettings>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="@TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="@TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="@TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public string Name { get; set; } = "Child";
public int Duration { get; set; } = 5;
public List<BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = BusinessObject.GetSelfDataSource().ToList();
}
}
```

C#

```
namespace TreeGridComponent.Data {
public class BusinessObject
{
public int TaskId { get; set; }
```

```

public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<BusinessObject> GetSelfDataSource()
{
    List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
    "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
    null });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
    "Child task 1", Duration = 4, Progress = 80, Priority = "Low", ParentId = 1 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
    "Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
    });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
    "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
    "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
    null });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
    "Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
    5 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
    "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
    "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
    "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
    return BusinessObjectCollection;
}
}
}

```

Task ID	Task Name	Duration	Progress	Prio...
	Child	5		
1	▼ Parent Task 1	10	70	Critical
2	▼ Child task 1	4	80	Low
3	Child Task 2	5	65	Critical
5	▼ Parent Task 2	10	70	Critical
7	Child Task 2	5	65	Low

Filter operators

The filter operator for a column can be defined in the **Operator** property of the [Columns](#) property of the [FilterSettings](#).

The available operators and its supported data types are:

Operator | Description | Supported Types

= | =value | equal | Number

!= | !=value | notequal | Number

|>value | greaterthan | Number

< | <value | lessthan | Number

= | >=value | greaterthanorequal | Number

<= | <=value | lessthanorequal | Number

/value | startswith | String

% | %value | endswith | String

N/A | N/A | **Equal** operator will always be used for date filter. | Date

N/A | N/A | **Equal** operator will always be used for Boolean filter. | Boolean

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
@using Syncfusion.Blazor.Data;
<SfTreeGrid IdMapping="TaskId" DataSource="@TreeGridData"
ParentIdMapping="ParentId" TreeColumnIndex="1" AllowFiltering="true">
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText="Start Date" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="60"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
public List<BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = BusinessObject.GetSelfDataSource().ToList();
}
}
```

C#

```

namespace TreeGridComponent.Data {
    public class BusinessObject
    {
        public int TaskId { get; set; }
        public string TaskName { get; set; }
        public int? Duration { get; set; }
        public int? Progress { get; set; }
        public string Priority { get; set; }
        public int? ParentId { get; set; }
        public static List<BusinessObject> GetSelfDataSource()
        {
            List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
                "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
                null });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
                "Child task 1", Duration = 4, Progress = 80, Priority = "Low", ParentId = 1 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
                "Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
            });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
                "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
                "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
                null});
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
                "Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
                5});
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
                "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5});
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
                "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5});
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
                "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5});
            return BusinessObjectCollection;
        }
    }
}

```

Filter bar template with custom component

The [FilterTemplate](#) property is used to add custom components to a particular column. To access the filtered values inside the [FilterTemplate](#), you can use the implicit named parameter context. You can type cast the context as `PredicateModel<T>` to get filter values inside template.

In the following sample, the dropdown is used as a custom component in the Duration column.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
@using Syncfusion.Blazor.Data;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeGridData" IdMapping="TaskId"
    ParentIdMapping="ParentId" TreeColumnIndex="1" AllowFiltering="true">
    <TreeGridColumn>
        <TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
            TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
    </TreeGridColumn>
</SfTreeGrid>

```

```

<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText="Start Date" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right">
<FilterTemplate>
<SfDropDownList TValue="string" DataSource="@DropDownData" TItem="string">
<DropDownListEvents ValueChange="change"
TValue="string"></DropDownListEvents>
<SfDropDownList TValue="string" DataSource="@DropDownData" TItem="string"
Value="@((string) (context as PredicateModel).Value)">
<DropDownListEvents ValueChange="change" TValue="string"
></DropDownListEvents>
</SfDropDownList>
</FilterTemplate>
</TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="60"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid<BusinessObject> TreeGrid;
public List<string> DropDownData { get; set; } = new List<string>();
public List<BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = BusinessObject.GetSelfDataSource().ToList();
this.DropDownData.Add("10");
this.DropDownData.Add("50");
this.DropDownData.Add("5");
this.DropDownData.Add("6");
this.DropDownData.Add("4");
this.DropDownData.Add("All");
}
public void change(Syncfusion.Blazor.DropDowns.ChangeEventArgs<string> Args)
{
if (Args.Value == "All")
{
TreeGrid.ClearFiltering("Duration");
}
else
{
int val = Convert.ToInt32(Args.Value);
TreeGrid.FilterByColumn("Duration", "equal", val);
}
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class BusinessObject
{

```

```
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<BusinessObject> GetSelfDataSource()
{
    List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
    "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
    null });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
    "Child task 1", Duration = 4, Progress = 80, Priority = "Low", ParentId = 1 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
    "Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
    });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
    "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
    "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
    null });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
    "Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
    5 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
    "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
    "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
    "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
    return BusinessObjectCollection;
}
}
```


Task ID	Task Name	Duration	Progress	Priority
		All		
1	▼ Parent Task 1	10	70	Critical
2	▼ Child task 1	50	80	Low
3	▼ Child Task 2	5	65	Critical
4	Child task 3	6	77	High
5	▼ Parent Task 2	4	70	Critical
6	Child task 1	All	80	Critical
7	Child Task 2	5	65	Low
8	Child task 3	6	77	High
9	Child task 4	6	77	Low

Filter menu

The filter menu can be enabled by setting the [Type](#) of [FilterSettings](#) as **Menu**. The filter menu UI will be rendered based on its column type, which allows to filter data. The records can be filtered with different operators.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
@using Syncfusion.Blazor.Data;
<SfTreeGrid IdMapping="TaskId" DataSource="@TreeGridData"
ParentIdMapping="ParentId" TreeColumnIndex="1" AllowFiltering="true">
<TreeGridFilterSettings HierarchyMode="@FilterHierarchyMode.Parent"
Type="Syncfusion.Blazor.TreeGrid.FilterType.Menu"></TreeGridFilterSettings>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public List<BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{

```

```

this.TreeGridData = BusinessObject.GetSelfDataSource().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<BusinessObject> GetSelfDataSource()
{
List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", Duration = 4, Progress = 80, Priority = "Low", ParentId = 1 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5});
return BusinessObjectCollection;
}
}
}

```

Task ID	Task Name	Duration	Progress	Priority
	Starts With	10	70	Critical
	Enter the value	4	80	Low
		5	65	Critical
		6	77	High
5	Parent Task 2	10	70	Critical
6	Child task 1	4	80	Critical
7	Child Task 2	5	65	Low
8	Child task 3	6	77	High
9	Child task 4	6	77	Low

* [AllowFiltering](#) must be set as true to enable filter menu.

* Setting [AllowFiltering](#) property of [TreeGridColumn](#) as false will prevent filter menu rendering for a particular column.

Custom component in filter menu

The [FilterTemplate](#) property of [Column] is used to add custom filter components to a particular column. In the following sample the FilterTemplate property is used to add custom components to a particular column. To access the filtered values inside the FilterTemplate, you can use the implicit named parameter context. You can type cast the context as [PredicateModel<T>](#) to get filter values inside template.

In the following sample, dropdown is used as custom component in the duration column.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
@using Syncfusion.Blazor.Data;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" AllowFiltering="true">
<TreeGridFilterSettings
Type="Syncfusion.Blazor.TreeGrid.FilterType.Menu"></TreeGridFilterSettings>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText="Start Date" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right">
<FilterTemplate>
```

```

<SfDropDownList TValue="string" DataSource="@DropDownData" TItem="string"
Value="@((string) (context as PredicateModel).Value)">
<DropDownListEvents ValueChange="change" TValue="string"
></DropDownListEvents>
</SfDropDownList>
</FilterTemplate>
</TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="60"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid<BusinessObject> TreeGrid;
public List<BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = BusinessObject.GetSelfDataSource().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<BusinessObject> GetSelfDataSource()
{
List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", Duration = 4, Progress = 80, Priority = "Low", ParentId = 1 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5});
}
}

```

```

BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5});
return BusinessObjectCollection;
}
}
}

```

Task ID	Task Name	Duration	Progress	Priority
1	Equal		70	Critical
2			80	Low
3			65	Critical
4			77	High
5		10	70	Critical
6		4	80	Critical
7		5	65	Low
8	Child task 3	6	77	High
9	Child task 4	6	77	Low

Enable different filter for a column

Both the **Menu** and **Excel** filter can be used in a same Tree Grid. To do so, set the type as **Menu** or **Excel** using the **Filter** property of the [TreeGridColumn](#).

In the following sample menu filter is enabled by default and excel filter is enabled for the Task Name column using the [Filter] property of [TreeGridColumn](#).

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
@using Syncfusion.Blazor.Data;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" AllowFiltering="true">
<TreeGridFilterSettings HierarchyMode="FilterHierarchyMode.None"
Type="Syncfusion.Blazor.TreeGrid.FilterType.Excel"></TreeGridFilterSettings>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
FilterSettings="@ (new Syncfusion.Blazor.Grids.FilterSettings{ Type =
Syncfusion.Blazor.Grids.FilterType.Menu })"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="100"></TreeGridColumn>

```

```

<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority" Width="100"
FilterSettings="@ (new Syncfusion.Blazor.Grids.FilterSettings{ Type =
Syncfusion.Blazor.Grids.FilterType.Menu })"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
public List<BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = BusinessObject.GetSelfDataSource().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<BusinessObject> GetSelfDataSource()
{
List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", Duration = 4, Progress = 80, Priority = "Low", ParentId = 1 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5});
return BusinessObjectCollection;
}
}

```

```
}

```

Excel like filter

Excel like filter can be enabled by defining the [Type](#) of [FilterSettings](#) as **Excel**. The excel menu contains an option such as Sorting, Clear filter, Sub menu for advanced filtering.

C#

```
@using TreeGridComponent.Data
@using Syncfusion.Blazor.TreeGrid;
@using Syncfusion.Blazor.Data;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" AllowFiltering="true">
<TreeGridFilterSettings HierarchyMode="FilterHierarchyMode.Parent"
Type="Syncfusion.Blazor.TreeGrid.FilterType.Excel"></TreeGridFilterSettings>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText="Start Date" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="60"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
public List<BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = BusinessObject.GetSelfDataSource().ToList();
}
}
```

C#

```
namespace TreeGridComponent.Data {
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<BusinessObject> GetSelfDataSource()
{
List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null });
}
```

```

BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", Duration = 4, Progress = 80, Priority = "Low", ParentId = 1 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5});
return BusinessObjectCollection;
}
}
}

```

Task ID	Task Name	Start Date	Duration	Progress	Priority
1	Parent Task		10	70	Critical
2	Child task		4	80	Low
3	Child Tas		5	65	Critical
4	Child ta		6	77	High
5	Parent Task		10	70	Critical
6	Child task		4	80	Critical
7	Child Task		5	65	Low
8	Child task		6	77	High
9	Child task		6	77	Low
10	Parent Task		10	70	

Clear Filter

Text Filters

Search

☒ Select All

☒ Child task 1

☒ Child Task 2

☒ Child task 3

☒ Child task 4

☒ Child task 5

OK CANCEL

Sorting in Blazor TreeGrid Component

Sorting enables 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. Sorting of any one of the multi-sorted columns can be cleared by pressing and holding the SHIFT key and clicking the specific column header.

To enable sorting in the Tree Grid, set the [AllowSorting](#) to true. Sorting options can be configured through the [TreeGridSortSettings](#).

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" AllowSorting="true"
IdMapping="TaskId" ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridSortSettings Columns="@Sort"></TreeGridSortSettings>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public List<TreeData.BusinessObject> TreeGridData { get; set; }
public List<TreeGridSortColumn> Sort { get; set; }
protected override void OnInitialized()
{
this.Sort = new List<TreeGridSortColumn>();
this.Sort.Add(new TreeGridSortColumn() { Field = "TaskName", Direction =
Syncfusion.Blazor.Grids.SortDirection.Descending });
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
}
public static List<BusinessObject> GetSelfDataSource()
{
List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", Progress = 80, Priority = "Low", ParentId = 1 });
}
}

```

```

BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5});
return BusinessObjectCollection;
}
}
}

```

* Tree Grid columns are sorted in the **Ascending** order. If you click the already sorted column, the sort direction toggles.

* Apply and clear sorting by invoking [SortByColumn](#) 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 the [Columns](#) property of [SortSettings](#).

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" AllowSorting="true"
IdMapping="TaskId" ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridSortSettings>
<TreeGridSortColumns>
<TreeGridSortColumn Field="TaskName"
Direction="Syncfusion.Blazor.Grids.SortDirection.Descending"></TreeGridSortC
olumn>
</TreeGridSortColumns>
</TreeGridSortSettings>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumn>
</TreeGridColumns>

```

```

</SfTreeGrid>
@code{
public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
    this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
    public class BusinessObject
    {
        public int TaskId { get; set; }
        public string TaskName { get; set; }
        public int? Duration { get; set; }
        public int? Progress { get; set; }
        public string Priority { get; set; }
        public int? ParentId { get; set; }
    }
    public static List<BusinessObject> GetSelfDataSource()
    {
        List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
        "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
        null });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
        "Child task 1", Progress = 80, Priority = "Low", ParentId = 1 });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
        "Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
        });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
        "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
        "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
        null});
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
        "Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
        5});
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
        "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5});
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
        "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5});
        BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
        "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5});
        return BusinessObjectCollection;
    }
}
}

```

Task ID	Task Name	↓	Duration	Progress	Priority
5	▼ Parent Task 2		10	70	Critical
9	Child task 4		6	77	Low
8	Child task 3		6	77	High
7	Child Task 2		5	65	Low
6	Child task 1		4	80	Critical
1	▼ Parent Task 1		10	70	Critical
2	▼ Child task 1		4	80	Low
3	▼ Child Task 2		5	65	Critical
4	Child task 3		6	77	High

Sorting events

During the sort action, the tree 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 the needed actions can be performed.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
@inject IJSRuntime JsRuntime;
<SfTreeGrid DataSource="@TreeGridData" AllowSorting="true"
  IdMapping="TaskId" ParentIdMapping="ParentId" TreeColumnIndex="1">
  <TreeGridEvents OnActionBegin="actionbegin"
    OnActionComplete="actioncomplete" TValue="TreeData"></TreeGridEvents>
  <TreeGridColumns>
    <TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
      TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
    <TreeGridColumn Field="TaskName" HeaderText="Task Name"
      Width="160"></TreeGridColumn>
    <TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
      TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
    <TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
      TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
    <TreeGridColumn Field="Priority" HeaderText="Priority"
      Width="80"></TreeGridColumn>
  </TreeGridColumns>
</SfTreeGrid>
@code{
  public List<TreeData> TreeGridData { get; set; }
  protected override void OnInitialized()
```

```

{
    this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
private void actionbegin(ActionEventArgs<TreeData> args)
{
    JsRuntime.InvokeAsync<string>("window.alert", args.RequestType.ToString());
}
private void actioncomplete(ActionEventArgs<TreeData> args)
{
    JsRuntime.InvokeAsync<string>("window.alert", args.RequestType.ToString());
}
}

```

C#

```

namespace TreeGridComponent.Data {
    public class TreeData
    {
        public int TaskId { get; set; }
        public string TaskName { get; set; }
        public int? Duration { get; set; }
        public int? Progress { get; set; }
        public string Priority { get; set; }
        public int? ParentId { get; set; }
        public static List<TreeData> GetSelfDataSource()
        {
            List<TreeData> TreeDataCollection = new List<TreeData>();
            TreeDataCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
            TreeDataCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child task 1", Progress = 80, Priority = "Low", ParentId = 1 });
            TreeDataCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
            TreeDataCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
            TreeDataCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
            TreeDataCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5 });
            TreeDataCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
            TreeDataCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
            TreeDataCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
            return TreeDataCollection;
        }
    }
}

```

The **args.requestType** is the current action name. For example, in sorting the **args.requestType** value is *sorting*.

Touch interaction

When the tree grid header is tapped on the touchscreen devices, the selected column header is sorted.



A popup is displayed for the multi-column sorting. To sort multiple columns, tap the popup



, and then tap the desired tree grid headers.

<!-- markdownlint-disable MD033 -->

<!-- markdownlint-enable MD033 -->

Searching in Blazor TreeGrid Component

In a Tree Grid, the records are searched by using the [Search](#) method with search key as a parameter.

This also provides an option to integrate search text box in tree grid's toolbar by adding **Search** item to the [Toolbar](#).

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
@using Syncfusion.Blazor.Data;
<SfTreeGrid IdMapping="TaskId" DataSource="@TreeGridData"
ParentIdMapping="ParentId" TreeColumnIndex="1" Toolbar="@ (new List<string>()
{ "Search" }) ">
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
public List<BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = BusinessObject.GetSelfDataSource().ToList();
}
}
```

C#

```
namespace TreeGridComponent.Data {
public class BusinessObject
{
```

```
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<BusinessObject> GetSelfDataSource()
{
    List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
    "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
    null });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
    "Child task 1", Progress = 80, Priority = "Low", ParentId = 1 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
    "Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
    });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
    "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
    "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
    null });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
    "Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
    5 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
    "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
    "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
    "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
    return BusinessObjectCollection;
}
}
```

<div> <div>Search</div> <div></div> </div>				
Task ID	Task Name	Duration	Progress	Priority
1	▼ Parent Task 1	10	70	Critical
2	▼ Child task 1	4	80	Low
3	▼ Child Task 2	5	65	Critical
4	Child task 3	6	77	High
5	▼ Parent Task 2	10	70	Critical
6	Child task 1	4	80	Critical
7	Child Task 2	5	65	Low
8	Child task 3	6	77	High
9	Child task 4	6	77	Low

Initial search

To apply search at initial rendering, set the Fields, Operator, Key, and IgnoreCase in the [TreeGridSearchSettings](#).

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
@using Syncfusion.Blazor.Data;
<SfTreeGrid IdMapping="TaskId" TValue="TreeData.BusinessObject"
ParentIdMapping="ParentId" TreeColumnIndex="1" Toolbar="@ (new List<string>()
{ "Search" }) ">
<SfDataManager Json="@TreeGridData"
Adaptor="Syncfusion.Blazor.Adaptors.JsonAdaptor">
</SfDataManager>
<TreeGridSearchSettings Key="Child Task 1"></TreeGridSearchSettings>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
```



```

</TreeGridColumns>
</SfTreeGrid>
@code{
    public TreeData.BusinessObject[] TreeGridData { get; set; }
    protected override void OnInitialized()
    {
        this.TreeGridData =
            TreeData.GetSelfDataSource().ToList().Cast<TreeData.BusinessObject>().ToArray();
    }
}

```

C#

```

namespace TreeGridComponent.Data {
    public class BusinessObject
    {
        public int TaskId { get; set; }
        public string TaskName { get; set; }
        public int? Duration { get; set; }
        public int? Progress { get; set; }
        public string Priority { get; set; }
        public int? ParentId { get; set; }
        public static List<BusinessObject> GetSelfDataSource()
        {
            List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
                "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
                null });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
                "Child task 1", Progress = 80, Priority = "Low", ParentId = 1 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
                "Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
            });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
                "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
                "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
                null });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
                "Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
                5 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
                "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
                "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
                "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
            return BusinessObjectCollection;
        }
    }
}

```

Child Task 1 Q				
Task ID	Task Name	Duration	Progress	Priority
1	▼ Parent Task 1	10	70	Critical
2	Child task 1	4	80	Low
5	▼ Parent Task 2	10	70	Critical
6	Child task 1	4	80	Critical

By default, tree grid searches all the bound column values. To customize this behavior define the [TreeGridSearchSettings.Fields](#) property.

Search operators

The search operator can be defined in the [Operators](#) property of the [TreeGridSearchSettings](#) 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.

By default, the [Operators](#) value is **contains**.

Search by external button

To search tree grid records from an external button, invoke the [Search](#) method.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid
@using Syncfusion.Blazor.Data;
<button id="hide" @onclick="search">Search Tree</button>
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" Toolbar="@ (new List<string>()
{ "Search" })">
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
```

```

<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid<BusinessObject> TreeGrid;
public List<BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = BusinessObject.GetSelfDataSource().ToList();
}
private void search()
{
this.TreeGrid.Search("Child Task 1");
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<BusinessObject> GetSelfDataSource()
{
List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", Progress = 80, Priority = "Low", ParentId = 1 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
5 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
return BusinessObjectCollection;
}
}

```

```
}
}
}
```

Search specific columns

By default, the tree grid searches all the visible columns. Specific columns can be searched by defining the specific column's field names in the [Fields](#) property of the [TreeGridSearchSettings](#).

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
@using Syncfusion.Blazor.Data;
@{
    var Tool = (new List<string>() { "Search" });
    var SpecificCols = (new string[] { "TaskId", "Duration" });
}
<SfTreeGrid IdMapping="TaskId" DataSource="@TreeGridData"
ParentIdMapping="ParentId" TreeColumnIndex="1" Toolbar="@Tool">
<TreeGridSearchSettings Fields="@SpecificCols"></TreeGridSearchSettings>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
    public List<BusinessObject> TreeGridData { get; set; }
    protected override void OnInitialized()
    {
        this.TreeGridData = BusinessObject.GetSelfDataSource().ToList();
    }
}
```

C#

```
namespace TreeGridComponent.Data {
    public class BusinessObject
    {
        public int TaskId { get; set; }
        public string TaskName { get; set; }
        public int? Duration { get; set; }
        public int? Progress { get; set; }
        public string Priority { get; set; }
        public int? ParentId { get; set; }
        public static List<BusinessObject> GetSelfDataSource()
        {
            List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
```

```

BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", Progress = 80, Priority = "Low", ParentId = 1 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5});
return BusinessObjectCollection;
}
}
}

```

Paging in Blazor TreeGrid Component

Paging provides an option to display Tree 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 tree grid. Paging options can be configured through the [TreeGridPageSettings](#).

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" AllowPaging="true">
<TreeGridPageSettings PageCount="2" PageSize="2"
PageSizeMode="PageSizeMode.Root">
</TreeGridPageSettings>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public List<TreeData.BusinessObject> TreeGridData { get; set; }
}

```

```
protected override void OnInitialized()
{
    this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}
```

C#

```
namespace TreeGridComponent.Data {
    public class TreeData
    {
        public class BusinessObject
        {
            public int TaskId { get; set; }
            public string TaskName { get; set; }
            public int? Duration { get; set; }
            public int? Progress { get; set; }
            public string Priority { get; set; }
            public int? ParentId { get; set; }
        }
        public static List<BusinessObject> GetSelfDataSource()
        {
            List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
                "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
                null });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
                "Child task 1", Progress = 80, Priority = "Low", ParentId = 1 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
                "Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
            });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
                "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
                "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
                null });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
                "Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
                5 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
                "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
                "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
                "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
            return BusinessObjectCollection;
        }
    }
}
```

Task ID	Task Name	Duration	Progress	Priority
1	▼ Parent Task 1	10	70	Critical
2	▼ Child task 1	4	80	Low
3	▼ Child Task 2	5	65	Critical
4	Child task 3	6	77	High
5	▼ Parent Task 2	10	70	Critical
6	Child task 1	4	80	Critical
7	Child Task 2	5	65	Low
8	Child task 3	6	77	High
9	Child task 4	6	77	Low
< < 1 2 ... > >				1 of 3 pages (6 items)

Better performance can be achieved by using tree grid paging to fetch only a pre-defined number of records from the data source.

Page size mode

Two behaviors are available in Tree Grid paging to display certain number of records in a current page. Following are the two types of [PageSizeMode](#) property of [TreeGridPageSettings](#).

- **All** : This is the default mode. The number of records in a page is based on [PageSize](#) property.
- **Root** : The number of root nodes or the 0th level records to be displayed per page is based on [PageSize](#) property. With [PageSizeMode](#) property as **Root**, only the root level or the 0th level records are considered in records count.

By default, Blazor Tree Grid works with **PageSizeMode** as *Root* and to behave as *All* mode, **Adaptor** property should be set for Tree Grid **SfDataManager**.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" AllowPaging="true">
<TreeGridPageSettings PageCount="2" PageSize="2"
PageSizeMode="PageSizeMode.Root">
</TreeGridPageSettings>
<TreeGridColumns>
```

```

<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
}
public static List<BusinessObject> GetSelfDataSource()
{
List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", Progress = 80, Priority = "Low", ParentId = 1 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5});
}
}

```



```

BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5});
return BusinessObjectCollection;
}
}
}

```

Task ID	Task Name	Duration	Progress	Priority
1	▼ Parent Task 1	10	70	Critical
2	▼ Child task 1	4	80	Low
3	▼ Child Task 2	5	65	Critical
4	Child task 3	6	77	High
5	▼ Parent Task 2	10	70	Critical
6	Child task 1	4	80	Critical
7	Child Task 2	5	65	Low
8	Child task 3	6	77	High
9	Child task 4	6	77	Low
<< < 1 2 ... > >>				1 of 3 pages (6 items)

<!-- Template

You can use custom elements inside the pager instead of default elements.

The custom elements can be defined by using the [Template](#) property.

Inside this template, you can access the [CurrentPage](#), [PageSize](#), [PageCount](#), **TotalPage** and **TotalRecordCount** values.

-->

[Pager with page size dropdown](#)

The pager Dropdown allows to change the number of records in the Tree Grid dynamically. It can be enabled by defining the [PageSizes](#) property of [TreeGridPageSettings](#) as **true**.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" AllowPaging="true">

```

```

<TreeGridPageSettings PageCount="2" PageSize="2"
  PageSizeMode="PageSizeMode.Root" PageSizes="new List<int>() { 2, 5,
  10}"></TreeGridPageSettings>
<TreeGridColumns>
  <TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
    TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
  <TreeGridColumn Field="TaskName" HeaderText="Task Name"
    Width="160"></TreeGridColumn>
  <TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
    TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
  <TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
    TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
  <TreeGridColumn Field="Priority" HeaderText="Priority"
    Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
  public List<TreeData.BusinessObject> TreeGridData { get; set; }
  protected override void OnInitialized()
  {
    this.TreeGridData = TreeData.GetSelfDataSource().ToList();
  }
}

```

C#

```

namespace TreeGridComponent.Data {
  public class TreeData
  {
    public class BusinessObject
    {
      public int TaskId { get; set; }
      public string TaskName { get; set; }
      public int? Duration { get; set; }
      public int? Progress { get; set; }
      public string Priority { get; set; }
      public int? ParentId { get; set; }
    }
    public static List<BusinessObject> GetSelfDataSource()
    {
      List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
      BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
        "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
        null });
      BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
        "Child task 1", Progress = 80, Priority = "Low", ParentId = 1 });
      BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
        "Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
      });
      BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
        "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
      BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
        "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
        null });
    }
  }
}

```

```

BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5});
return BusinessObjectCollection;
}
}
}

```

Task ID	Task Name	Duration	Progress	Priority
1	▼ Parent Task 1	10	70	Critical
2	▼ Child task 1	4	80	Low
3	▼ Child Task 2	5	65	Critical
4	Child task 3	6	77	High
5	▼ Parent Task 2	10	70	Critical
6	Child task 1	4	80	Critical
7	Child Task 2	5	65	Low
8	Child task 3	6	77	High
9	Child task 4	6	77	Low
<div> <div> < < 1 2 ... > > </div> <div>2</div> <div>▼</div> </div> <div>Items per page</div> <div>1 of 3 pages (6 items)</div>				

<!--How to render Pager at the Top of the Tree Grid

By default, Pager will be rendered at the bottom of the Tree Grid. You can also render the Pager at the top of the Tree Grid by using the [DataBound](#) event.

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.

-->

Scrolling in Blazor TreeGrid Component

The scrollbar will be displayed in the tree grid when the 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 tree grid exceeds its element height. The horizontal scrollbar appears when the sum of the columns' width exceeds the tree grid element width. The [Height](#) and [Width](#) are used to set the tree grid height and width, respectively.

The default value for [Height](#) and [Width](#) is **auto**.

Set width and height

To specify the [Height](#) and [Width](#) of the scroller in the pixel, set the pixel value to a number.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid Width="500" Height="300" DataSource="@TreeGridData"
IdMapping="TaskId" ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}
```

C#

```
namespace TreeGridComponent.Data {
public class TreeData
{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
}
public static List<BusinessObject> GetSelfDataSource()
{
```

```

List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", Progress = 80, Priority = "Low", ParentId = 1 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5});
return BusinessObjectCollection;
}
}
}

```

Task ID	Task Name	Duration	Progress
1	▼ Parent Task 1	10	70
2	▼ Child task 1	4	80
3	▼ Child Task 2	5	65
4	Child task 3	6	77
5	▼ Parent Task 2	10	70
6	Child task 1	4	80
7	Child Task 2	5	65
8	Child task 3	6	77

Responsive with parent container

Specify the [Height](#) and [Width](#) as **100%** to make the tree grid element fill its parent container. Setting the [Height](#) to **100%** requires the tree grid parent element to have explicit height.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<div class="e-resizable">
<SfTreeGrid Width="100%" Height="100%" DataSource="@TreeGridData"
IdMapping="TaskId" ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
</div>
@code{
public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}
<style>
.e-resizable {
resize: both;
overflow: auto;
border: 1px solid red;
padding: 10px;
height: 300px;
min-height: 250px;
min-width: 250px;
}
</style>

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
}
public static List<BusinessObject> GetSelfDataSource()
{

```

```

List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", Progress = 80, Priority = "Low", ParentId = 1 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5});
return BusinessObjectCollection;
}
}
}

```

<!--

Scroll to selected row

You can scroll the tree grid content to the selected row position by using the [RowSelected](#) event.

CSHARP

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
@using Syncfusion.Blazor.Grids
<SfTreeGrid ref="@treeGrid" Height="300" DataSource="@TreeGridData"
IdMapping="TaskId" RowSelected="onRowSelected" ParentIdMapping="ParentId"
TreeColumnIndex="1">
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid treeGrid;

```

```

public object[] TreeGridData { get; set; }
protected override void OnInitialized()
{
    this.TreeGridData =
    TreeData.GetSelfDataSource().ToList().Cast<object>().ToArray();
}
private void onRowSelected(object args)
{
    treeGrid.GetContent().
}
}

```

-->

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 tree 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.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" Height="400"
FrozenColumns="2" FrozenRows="3">
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="100"
TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="220"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="180"
TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress"
Width="180"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="180"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
    public List<TreeData.BusinessObject> TreeGridData { get; set; }
    protected override void OnInitialized()
    {
        this.TreeGridData = TreeData.GetSelfDataSource().ToList();
    }
}

```

C#

```

namespace TreeGridComponent.Data {
    public class TreeData
    {
        public class BusinessObject
        {

```



```
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
}

public static List<BusinessObject> GetSelfDataSource()
{
    List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
    "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
    null });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
    "Child task 1", Progress = 80, Priority = "Low", ParentId = 1 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
    "Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
    });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
    "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
    "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
    null });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
    "Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
    5 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
    "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
    "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
    "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
    return BusinessObjectCollection;
}
}
```

Task ID	Task Name	Duration	Progress
1	▼ Parent Task 1	3	40
2	Child Task 1	4	40
3	Child Task 2	2	40
4	Child Task 3	2	40
5	▼ Parent Task 2	6	40
6	Child Task 1	11	40
7	Child Task 2	7	40
8	Child Task 3	10	40
9	Child Task 4	15	40
10	▼ Parent Task 3	17	40

Freeze particular columns

To freeze particular column in the tree grid, the [IsFrozen](#) property can be used.

In this demo, the columns with the field name **TaskName** and **Duration** is frozen using the [IsFrozen](#) property.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" Height="400">
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="100"
TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name" Width="220"
IsFrozen="true"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="180"
TextAlign="TextAlign.Right" IsFrozen="true"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress"
Width="180"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="180"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}
```

C#

```
namespace TreeGridComponent.Data {
public class TreeData
{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
}
public static List<BusinessObject> GetSelfDataSource()
{
List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", Progress = 80, Priority = "Low", ParentId = 1 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5});
return BusinessObjectCollection;
}
}
}
```

Task Name	Duration	Task ID	Progress
▼ Parent Task 1	3	1	40
Child Task 1	4	2	40
Child Task 2	2	3	40
Child Task 3	2	4	40
▼ Parent Task 2	6	5	40
Child Task 1	11	6	40
Child Task 2	7	7	40
Child Task 3	10	8	40
Child Task 4	15	9	40
▼ Parent Task 3	17	10	40

Limitations

The following features are not supported in frozen rows and columns:

- Row Template
- Detail Template
- Cell Editing

Virtualization in Blazor TreeGrid Component

Tree Grid allows to load large amount of data without performance degradation.

Row Virtualization

Row virtualization allows to load and render rows only in the content viewport. It is an alternative way of paging in which the rows will be appended while scrolling vertically. To setup the row virtualization, define the [EnableVirtualization](#) as true and content height by [Height](#) property.

The number of records displayed in the Tree Grid is determined implicitly by height of the content area and a buffer records will be maintained in the Tree Grid content in addition to the original set of rows.

Expand and Collapse state of any child record will be persisted.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
@using Syncfusion.Blazor.Grids;
<SfTreeGrid TValue="VirtualData" DataSource="@TreeGridData"
ChildMapping="Children" EnableVirtualization="true" Height="350"
TreeColumnIndex="1">
<TreeGridColumn>
```

```

<TreeGridColumn Field="TaskID" HeaderText="Player Jersey" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="FIELD1" HeaderText="Player Name"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="FIELD2" HeaderText="Year" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="FIELD3" HeaderText="Stint" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="FIELD4" HeaderText="TMID"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public VirtualData[] TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = VirtualData.GetVirtualData().ToArray();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class VirtualData
{
public int? TaskID { get; set; }
public string FIELD1 { get; set; }
public int? FIELD2 { get; set; }
public int? FIELD3 { get; set; }
public int? FIELD4 { get; set; }
public List<VirtualData> Children { get; set; }
public static List<VirtualData> GetVirtualData()
{
List<VirtualData> DataCollection = new List<VirtualData>();
for (var i = 1; i <= 1000; i++)
{
VirtualData Parent1 = new VirtualData()
{
TaskID = 1,
FIELD1 = "VINET",
FIELD2 = 1967,
FIELD3 = 395,
FIELD4 = 87,
Children = new List<VirtualData>()
};
VirtualData Child1 = new VirtualData()
{
TaskID = 2,
FIELD1 = "TOMSP",
FIELD2 = 1968,
FIELD3 = 295,
FIELD4 = 44
};
VirtualData Child2 = new VirtualData()
{
TaskID = 3,

```

```
FIELD1 = "HANAR",
FIELD2 = 1969,
FIELD3 = 376,
FIELD4 = 22
};
VirtualData Child3 = new VirtualData()
{
    TaskID = 4,
    FIELD1 = "VICTE",
    FIELD2 = 1970,
    FIELD3 = 123,
    FIELD4 = 35
};
VirtualData Child4 = new VirtualData()
{
    TaskID = 5,
    FIELD1 = "SUPRD",
    FIELD2 = 1971,
    FIELD3 = 567,
    FIELD4 = 98
};
VirtualData Child5 = new VirtualData()
{
    TaskID = 6,
    FIELD1 = "RICSU",
    FIELD2 = 1972,
    FIELD3 = 378,
    FIELD4 = 56
};
VirtualData Parent2 = new VirtualData()
{
    TaskID = 1,
    FIELD1 = "TOMSP",
    FIELD2 = 1968,
    FIELD3 = 295,
    FIELD4 = 44,
    Children = new List<VirtualData>()
};
VirtualData Child6 = new VirtualData()
{
    TaskID = 2,
    FIELD1 = "VINET",
    FIELD2 = 1967,
    FIELD3 = 395,
    FIELD4 = 87
};
VirtualData Child7 = new VirtualData()
{
    TaskID = 3,
    FIELD1 = "VICTE",
    FIELD2 = 1970,
    FIELD3 = 123,
    FIELD4 = 35
};
VirtualData Child8 = new VirtualData()
{
    TaskID = 4,
```

```
FIELD1 = "RICSU",
FIELD2 = 1972,
FIELD3 = 378,
FIELD4 = 56
};
VirtualData Child9 = new VirtualData()
{
    TaskID = 5,
    FIELD1 = "HANAR",
    FIELD2 = 1969,
    FIELD3 = 376,
    FIELD4 = 22
};
VirtualData Child10 = new VirtualData()
{
    TaskID = 6,
    FIELD1 = "SUPRD",
    FIELD2 = 1971,
    FIELD3 = 567,
    FIELD4 = 98
};
Parent1.Children.Add(Child1);
Parent1.Children.Add(Child2);
Parent1.Children.Add(Child3);
Parent1.Children.Add(Child4);
Parent1.Children.Add(Child5);
Parent2.Children.Add(Child6);
Parent2.Children.Add(Child7);
Parent2.Children.Add(Child8);
Parent2.Children.Add(Child9);
Parent2.Children.Add(Child10);
DataCollection.Add(Parent1);
DataCollection.Add(Parent2);
}
return DataCollection;
}
}
```

Player Jersey	Player Name	Year	Stint	TMID
1	VINET	1967	395	87
2	TOMSP	1968	295	44
3	HANAR	1969	376	22
4	VICTE	1970	123	35
5	SUPRD	1971	567	98
6	RICSU	1972	378	56
1	TOMSP	1968	295	44
2	VINET	1967	395	87
3	VICTE	1970	123	35
4	RICSU	1972	378	56

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**.

CSHARP

```
@using Syncfusion.Blazor.Grids
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeData" IdMapping="TaskID" TreeColumnIndex="1"
ParentIdMapping="ParentID" Height="400" Width="600" EnableHover="false"
EnableVirtualization="true" EnableColumnVirtualization="true">
<TreeGridPageSettings PageSize="40"></TreeGridPageSettings>
<TreeGridColumns>
<TreeGridColumn Field="TaskID" HeaderText="Jersey No"
TextAlign="TextAlign.Right" Width="150"></TreeGridColumn>
<TreeGridColumn Field="FIELD1" HeaderText="Name"
Width="150"></TreeGridColumn>
<TreeGridColumn Field="FIELD2" HeaderText="Year" TextAlign="TextAlign.Right"
Width="150"></TreeGridColumn>
<TreeGridColumn Field="FIELD3" HeaderText="Stint"
TextAlign="TextAlign.Right" Width="150"></TreeGridColumn>
<TreeGridColumn Field="FIELD4" HeaderText="TMID" TextAlign="TextAlign.Right"
Width="150"></TreeGridColumn>
<TreeGridColumn Field="FIELD5" HeaderText="LGID" TextAlign="TextAlign.Right"
Width="150"></TreeGridColumn>
<TreeGridColumn Field="FIELD6" HeaderText="GP" TextAlign="TextAlign.Right"
Width="150"></TreeGridColumn>
<TreeGridColumn Field="Field7" HeaderText="GS" TextAlign="TextAlign.Right"
Width="150"></TreeGridColumn>
<TreeGridColumn Field="Field8" HeaderText="Minutes"
TextAlign="TextAlign.Right" Width="150"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
```



```

<TreeGridColumn Field="Field9" HeaderText="Points"
TextAlign="TextAlign.Right" Width="150"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public List<VirtualData> TreeData { get; set; }
protected override void OnInitialized()
{
this.TreeData = VirtualData.GetTreeVirtualData().ToList();
}
public class VirtualData
{
public int TaskID { get; set; }
public string FIELD1 { get; set; }
public int FIELD2 { get; set; }
public int FIELD3 { get; set; }
public int FIELD4 { get; set; }
public int FIELD5 { get; set; }
public int FIELD6 { get; set; }
public int Field7 { get; set; }
public int Field8 { get; set; }
public int Field9 { get; set; }
public int? ParentID { get; set; }
public static List<VirtualData> GetTreeVirtualData()
{
string[] Names = new string[] { "VINET", "TOMSP", "HANAR", "VICTE", "SUPRD",
"HANAR", "CHOPS", "RICSU", "WELLI", "HILAA", "ERNSH", "CENTC",
"OTTIK", "QUEDE", "RATTC", "ERNSH", "FOLKO", "BLONP", "WARTH", "FRANK",
"GROSR", "WHITC", "WARTH", "SPLIR", "RATTC", "QUICK", "VINET",
"MAGAA", "TORTU", "MORGK", "BERGS", "LEHMS", "BERGS", "ROMEY", "ROMEY",
"LILAS", "LEHMS", "QUICK", "QUICK", "RICAR", "REGGC", "BSBEV",
"COMMI", "QUEDE", "TRADH", "TORTU", "RATTC", "VINET", "LILAS", "BLONP",
"HUNGO", "RICAR", "MAGAA", "WANDK", "SUPRD", "GODOS", "TORTU",
"OLDWO", "ROMEY", "LONEP", "ANATR", "HUNGO", "THEBI", "DUMON", "WANDK",
"QUICK", "RATTC", "ISLAT", "RATTC", "LONEP", "ISLAT", "TORTU",
"WARTH", "ISLAT", "PERIC", "KOENE", "SAVEA", "KOENE", "BOLID", "FOLKO",
"FURIB", "SPLIR", "LILAS", "BONAP", "MEREP", "WARTH", "VICTE",
"HUNGO", "PRINI", "FRANK", "OLDWO", "MEREP", "BONAP", "SIMOB", "FRANK",
"LEHMS", "WHITC", "QUICK", "RATTC", "FAMIA" };
List<VirtualData> DataCollection = new List<VirtualData>();
Random random = new Random();
var RecordID = 0;
for (var i = 1; i <= 2000; i++)
{
var name = random.Next(0, 100);
VirtualData Parent = new VirtualData()
{
TaskID = ++RecordID,
FIELD1 = Names[name],
FIELD2 = 1967 + random.Next(0, 10),
FIELD3 = 395 + random.Next(100, 600),
FIELD4 = 87 + random.Next(50, 250),
FIELD5 = 410 + random.Next(100, 600),
FIELD6 = 67 + random.Next(50, 250),
Field7 = (int)Math.Floor(random.NextDouble() * 100),
Field8 = (int)Math.Floor(random.NextDouble() * 10),
Field9 = (int)Math.Floor(random.NextDouble() * 10),

```

```
ParentID = null
};
DataCollection.Add(Parent);
for (var j = 1; j <= 4; j++)
{
    var childName = random.Next(0, 100);
    DataCollection.Add(new VirtualData()
    {
        TaskID = ++RecordID,
        FIELD1 = Names[childName],
        FIELD2 = 1967 + random.Next(0, 10),
        FIELD3 = 395 + random.Next(100, 600),
        FIELD4 = 87 + random.Next(50, 250),
        FIELD5 = 410 + random.Next(100, 600),
        FIELD6 = 67 + random.Next(50, 250),
        Field7 = (int)Math.Floor(random.NextDouble() * 100),
        Field8 = (int)Math.Floor(random.NextDouble() * 10),
        Field9 = (int)Math.Floor(random.NextDouble() * 10),
        ParentID = Parent.TaskID
    });
}
}
return DataCollection;
}
```

Column's [Width](#) is required for column virtualization. If column's width is not defined then Tree Grid will consider its value as **200px**.

The following GIF represent a Tree Grid with Column virtualization.

Jersey No	Name	Year	Stir
1	▼ CENTC	1967	58
2	ANATR	1967	56
3	ANATR	1969	91
4	ISLAT	1972	78
5	BERGS	1970	67
6	▼ MEREP	1975	78
7	RATTC	1974	60
8	WARTH	1969	72
9	TORTU	1968	78
10	CHOPS	1971	74
11	▼ BONAP	1970	78

Limitations for Virtualization

- While using column virtualization, 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 tree grid is limited by the browser capability.
- Cell selection will not be persisted in both row and column virtualization.
- Stacked Header is not compatible with detail template.
- Virtual scrolling is not compatible with detail template.
- Row count of the page does not depend on the **PageSize** property of the **TreeGridPageSettings**. Row count for the page is determined by the [Height](#) given to the Tree Grid.
- The virtual height of the tree 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. In order to increase the row height to accommodate the content then the row height can be specified as below to ensure all the table rows are in same height.
- Programmatic selection using the **SelectRows** method is not supported in virtual scrolling.
- Frozen column feature is not supported with Virtual Scrolling.

Aggregate in Blazor TreeGrid Component

To know about Aggregate in Blazor tree grid Component, you can check this video.

{% youtube

"youtube:https://www.youtube.com/watch?v=h-ySOPTLaXk"%}

Aggregate values are displayed in the Tree Grid footer and in parent row footer for child row aggregate values. It can be configured through [TreeGridAggregateColumn](#) 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 Tree Grid footer, and footer of child rows. To show the aggregate value in one of the cells, use the [FooterTemplate](#).

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

Footer aggregate

Footer aggregate value is calculated for all the rows, and it is displayed in the footer cells. Use the [FooterTemplate](#) to render the aggregate value in footer cells.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" AllowPaging="true" TreeColumnIndex="1">
<TreeGridAggregates>
<TreeGridAggregate ShowChildSummary="false">
<TreeGridAggregateColumns>
<TreeGridAggregateColumn Field="Duration"
Type="Syncfusion.Blazor.Grids.AggregateType.Sum" Format="C2">
<FooterTemplate>
@{
var sumvalue = (context as
Syncfusion.Blazor.Grids.AggregateTemplateContext);
<div>
<p>Sum: @sumvalue.Sum</p>
</div>
}
</FooterTemplate>
</TreeGridAggregateColumn>
<TreeGridAggregateColumn Field="Approved"
Type="Syncfusion.Blazor.Grids.AggregateType.TrueCount" Format="C2">
<FooterTemplate>
@{
var truecount = (context as
Syncfusion.Blazor.Grids.AggregateTemplateContext);
<div>
<p>Approved: @truecount.TrueCount</p>
```

```

</div>
}
</FooterTemplate>
</TreeGridAggregateColumn>
</TreeGridAggregateColumns>
</TreeGridAggregate>
</TreeGridAggregates>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Approved" HeaderText="Approved"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center"
DisplayAsCheckBox="true" Width="100">
</TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public List<TreeData> TreeGridData { get; set; }
public Syncfusion.Blazor.Grids.AggregateTemplateContext model = new
Syncfusion.Blazor.Grids.AggregateTemplateContext();
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public DateTime? StartDate { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public bool Approved { get; set; }
public int? ParentId { get; set; }
public static List<TreeData> GetSelfDataSource()
{
List<TreeData> TreeDataCollection = new List<TreeData>();
TreeDataCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent Task
1", Duration = 10, Progress = 70, Approved = true, ParentId = null });
TreeDataCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child task
1", Duration = 4, Progress = 80, Approved = false, Duration = 50, ParentId =
1 });
TreeDataCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child Task
2", Duration = 5, Progress = 65, Approved = true, ParentId = 2 });
TreeDataCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child task
3", Duration = 6, Approved = false, Progress = 77, ParentId = 3 });
}
}

```

```

TreeDataCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent Task 2", Duration = 10, Progress = 70, Approved = true, ParentId = null });
TreeDataCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child task 1", Duration = 4, Progress = 80, Approved = false, ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child Task 2", Duration = 5, Progress = 65, Approved = true, ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child task 3", Duration = 6, Progress = 77, Approved = false, ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child task 4", Duration = 6, Progress = 77, Approved = true, ParentId = 5 });
return TreeDataCollection;
}
}
}

```

Task ID	Task Name	Duration	Progress	Approved
1	▼ Parent Task 1	10	70	<input checked="" type="checkbox"/>
2	▼ Child task 1	50	80	<input type="checkbox"/>
3	▼ Child Task 2	5	65	<input checked="" type="checkbox"/>
4	Child task 3	6	77	<input type="checkbox"/>
5	▼ Parent Task 2	10	70	<input checked="" type="checkbox"/>
6	Child task 1	4	80	<input type="checkbox"/>
7	Child Task 2	5	65	<input checked="" type="checkbox"/>
8	Child task 3	6	77	<input type="checkbox"/>
9	Child task 4	6	77	<input checked="" type="checkbox"/>
Sum: \$102.00			Approved: 5	
< < 1 > >				1 of 1 pages (2 items)

The aggregate values must be accessed inside the template using their corresponding **AggregateType**.

How to format aggregate value

The aggregate value result can be formatted by using the [Format](#) property.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" AllowPaging="true" TreeColumnIndex="1">
<TreeGridAggregates>
<TreeGridAggregate ShowChildSummary="false">

```

```

<TreeGridAggregateColumns>
<TreeGridAggregateColumn Field="Duration"
Type="Syncfusion.Blazor.Grids.AggregateType.Sum" Format="C2">
<FooterTemplate>
@{
var sumvalue = (context as
Syncfusion.Blazor.Grids.AggregateTemplateContext);
<div>
<p>Sum: @sumvalue.Sum</p>
</div>
}
</FooterTemplate>
</TreeGridAggregateColumn>
</TreeGridAggregateColumns>
</TreeGridAggregate>
</TreeGridAggregates>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Approved" HeaderText="Approved"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center"
DisplayAsCheckBox="true" Width="100">
</TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public List<TreeData> TreeGridData { get; set; }
public Syncfusion.Blazor.Grids.AggregateTemplateContext model = new
Syncfusion.Blazor.Grids.AggregateTemplateContext();
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public DateTime? StartDate { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public bool Approved { get; set; }
public int? ParentId { get; set; }
public static List<TreeData> GetSelfDataSource()
{
List<TreeData> TreeDataCollection = new List<TreeData>();
}
}

```

```

TreeDataCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent Task
1", Duration = 10, Progress = 70, Approved = true, ParentId = null });
TreeDataCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child task
1",Duration = 4, Progress = 80, Approved = false, Duration = 50, ParentId =
1 });
TreeDataCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child Task
2", Duration = 5, Progress = 65, Approved = true, ParentId = 2 });
TreeDataCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child task
3", Duration = 6, Approved = false, Progress = 77, ParentId = 3 });
TreeDataCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent Task
2", Duration = 10, Progress = 70, Approved = true, ParentId = null });
TreeDataCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child task
1", Duration = 4, Progress = 80, Approved = false, ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child Task
2", Duration = 5, Progress = 65, Approved = true, ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child task
3", Duration = 6, Progress = 77, Approved = false, ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child task
4", Duration = 6, Progress = 77, Approved = true, ParentId = 5 });
return TreeDataCollection;
}
}
}

```

Task ID	Task Name	Duration	Progress	Approved
1	▼ Parent Task 1	10	70	<input checked="" type="checkbox"/>
2	▼ Child task 1	50	80	<input type="checkbox"/>
3	▼ Child Task 2	5	65	<input checked="" type="checkbox"/>
4	Child task 3	6	77	<input type="checkbox"/>
5	▼ Parent Task 2	10	70	<input checked="" type="checkbox"/>
6	Child task 1	4	80	<input type="checkbox"/>
7	Child Task 2	5	65	<input checked="" type="checkbox"/>
8	Child task 3	6	77	<input type="checkbox"/>
9	Child task 4	6	77	<input checked="" type="checkbox"/>
Sum: \$102.00				
<div> << < 1 > >> 1 of 1 pages (2 items) </div>				

<!-- Custom aggregate

To calculate the aggregate value with your own aggregate functions, use the custom aggregate option. To use custom aggregation, specify the [Type](#) as **Custom**, and provide the custom aggregate function in the [CustomAggregate](#) property.

To access the custom aggregate value inside the template, use the key as **Custom**.

-->

Limitations

- By default, Footer Aggregate or total aggregate will be shown only for the current page records and not for the dataSource. To aggregate for all page records, set adaptor in **SfDataManager**.

Globalization in Blazor TreeGrid Component

Add **UseRequestLocalization** middle-ware in Configure method in **Startup.cs** file to get browser Culture Info.

Refer the following code to add configuration in Startup.cs file

CSHARP

```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Localization;
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            app.UseRequestLocalization();
            ....
            ....
        }
    }
}
```

Localization

The **Localization** library allows to localize default text content of the Tree Grid. The Tree Grid component has static text on some features (like pager information text, context menu options text, etc.) that can be changed to other cultures (Arabic, Deutsch, French, etc.).

Blazor Server Side

In the following examples, demonstrate how to enable **Localization** for Tree Grid in server side Blazor samples. Here, the Resource file is used to translate the static text of the Tree Grid.

The Resource file is an XML file which contains the strings(key and value pairs) that has to be translated into different language. Refer [Localization](#) link to know more about how to configure and use localization in the ASP.NET Core application framework.

- Open the **Startup.cs** file and add the below configuration in the **ConfigureServices** function as follows.

CSHARP

```
using Syncfusion.Blazor;
using System.Globalization;
using Microsoft.AspNetCore.Localization;
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void ConfigureServices(IServiceCollection services)
        {
            ....
            ....
            services.AddSyncfusionBlazor();
            services.AddLocalization(options => options.ResourcesPath = "Resources");
            services.Configure<RequestLocalizationOptions>(options =>
            {
                // define the list of cultures your app will support
                var supportedCultures = new List<CultureInfo>()
                {
                    new CultureInfo("de")
                };
                // set the default culture
                options.DefaultRequestCulture = new RequestCulture("de");
                options.SupportedCultures = supportedCultures;
                options.SupportedUICultures = supportedCultures;
                options.RequestCultureProviders = new List<IRequestCultureProvider>() {
                    new QueryStringRequestCultureProvider() // Here, You can also use other
                    localization provider
                };
            });
            services.AddSingleton(typeof(ISyncfusionStringLocalizer),
                typeof(SampleLocalizer));
        }
    }
}
```

Add [UseRequestLocalization\(\)](#) middle-ware in Configure method in **Startup.cs** file to get browser Culture Information.

- Then, write a **class** by inheriting **ISyncfusionStringLocalizer** interface and override the Manager property to get the resource file details from the application end.

CSHARP

```
using Syncfusion.Blazor;
namespace BlazorApplication
{
    public class SampleLocalizer : ISyncfusionStringLocalizer
    {
        public string Get(string key)
        {
            return this.Manager.GetString(key);
        }
    }
}
```

```

    }
    public System.Resources.ResourceManager Manager
    {
        get
        {
            return TreeGridBlazor.Resources.SyncfusionBlazorLocale.ResourceManager;
        }
    }
}

```

- Add **.resx** file to [Resource](#) folder and enter the key value (Locale Keywords) in the **Name** column and the translated string in the **Value** column as follows.

Name | Value (in Deutsch culture)

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 Tree 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

Column chooser | 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

Matches | 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

previousPagerTooltip | Go to previous pager

pagerDropDown | Items per page

pagerAllDropDown | Items

All | All

Loading translations

The following example demonstrates the Tree Grid in **Deutsch** culture. Here use **LoadLocaleData** method to load the **locale.json** file and **SetCulture** method to set the culture of the Tree Grid.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" AllowPaging="true" Locale="de-DE"
IdMapping="TaskId" AllowSelection="true" ParentIdMapping="ParentId"
TreeColumnIndex="1"
Toolbar="@ (new List<string>() { "Print", "ExpandAll", "CollapseAll" }) ">
<TreeGridPageSettings PageSize="1"></TreeGridPageSettings>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
[Inject]
IJSRuntime JsRuntime { get; set; }
public List<TreeData> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
protected override void OnAfterRender(bool firstRender)
{
if (firstRender)
{
this.JsRuntime.Sf().LoadLocaleData("locale.json").SetCulture("de-DE");
}
}
}
```

C#

```
namespace TreeGridComponent.Data {
```

```

public class TreeData
{
    public int TaskId { get; set; }
    public string TaskName { get; set; }
    public int? Duration { get; set; }
    public int? Progress { get; set; }
    public string Priority { get; set; }
    public int? ParentId { get; set; }
    public static List<TreeData> GetSelfDataSource()
    {
        List<TreeData> BusinessObjectCollection = new List<TreeData>();
        BusinessObjectCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
        BusinessObjectCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child task 1", Progress = 80, Priority = "Low", ParentId = 1 });
        BusinessObjectCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
        BusinessObjectCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 2 });
        BusinessObjectCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
        BusinessObjectCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5 });
        BusinessObjectCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
        BusinessObjectCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
        BusinessObjectCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
        return BusinessObjectCollection;
    }
}

```

JSON

```

{
    "de-DE": {
        "treegrid": {
            "EmptyRecord": "Keine Aufzeichnungen angezeigt",
            "ExpandAll": "Alle erweitern",
            "CollapseAll": "Alles einklappen",
            "Print": "Drucken",
            "Pdfexport": "PDF-Export",
            "Excelexport": "Excel-Export",
            "Wordexport": "Word-Export",
            "FilterButton": "Filter",
            "ClearButton": "klar",
            "StartsWith": "Beginnt mit",
            "EndsWith": "Endet mit",
            "Contains": "Enthält",
            "Equal": "Gleich",
            "NotEqual": "Nicht gleich",
            "LessThan": "Weniger als",
            "LessThanOrEqual": "Weniger als oder gleich",

```

```

"GreaterThan": "Größer als",
"GreaterThanOrEqual": "Größer als oder gleich",
"EnterValue": "Geben Sie den Wert ein",
"FilterMenu": "Filter"
},
"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 Seite",
"nextPagerTooltip": "Zum nächsten Pager",
"previousPagerTooltip": "Zum vorherigen Pager"
},
"dropdowns": {
"noRecordsTemplate": "Keine Aufzeichnungen gefunden"
},
"datepicker": {
"placeholder": "Wählen Sie ein Datum",
"today": "heute"
}
}
}
}

```

Drucken Alle erweitern Alles einklappen				
Task ID	Task Name	Duration	Progress	Priority
1	▼ Parent Task 1	10	70	Critical
2	▼ Child task 1	50	80	Low
3	Child Task 2	5	65	Critical
4	Child task 3	6	77	High
< < 1 > >				1 von 2 Seiten (2 Beiträge)

Internationalization

The **Internationalization** library is used to globalize number, date, and time values in Tree Grid component using format strings in the **Format**. In the below sample we set the culture and currency using the **SetCulture** and **SetCurrencyCode** methods.

C#

```

@using TreeGridComponent.Data;
@using Microsoft.JSInterop
@using Syncfusion.Blazor.TreeGrid;

```



```

<SfTreeGrid DataSource="@TreeGridData" AllowPaging="true" Locale="de-DE"
IdMapping="TaskId" AllowSelection="true" ParentIdMapping="ParentId"
TreeColumnIndex="1"
Toolbar="@ (new List<string>() { "Print", "ExpandAll", "CollapseAll" })">
<TreeGridPageSettings PageSize="1"></TreeGridPageSettings>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
Format="C2"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText=" Start Date" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
[Inject]
IJSRuntime JsRuntime { get; set; }
public List<TreeData> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
protected override void OnAfterRender(bool firstRender)
{
if (firstRender)
{
this.JsRuntime.Sf().SetCulture("de-DE").SetCurrencyCode("EUR");
}
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<TreeData> GetSelfDataSource()
{
List<TreeData> TreeDataCollection = new List<TreeData>();
TreeDataCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent Task
1", StartDate = new DateTime(2017, 10, 23), Duration = 10, Progress = 70,
Priority = "Critical", ParentId = null });
}
}

```

```

TreeDataCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child task
1", StartDate = new DateTime(2017, 10, 24), Progress = 80, Priority = "Low",
Duration = 50, ParentId = 1 });
TreeDataCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child Task
2", StartDate = new DateTime(2017, 10, 25), Duration = 5, Progress = 65,
Priority = "Critical", ParentId = 2 });
TreeDataCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child task
3", StartDate = new DateTime(2017, 10, 26), Duration = 6, Priority = "High",
Progress = 77, ParentId = 2 });
TreeDataCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent Task
2", StartDate = new DateTime(2017, 10, 27), Duration = 10, Progress = 70,
Priority = "Critical", ParentId = null });
TreeDataCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child task
1", StartDate = new DateTime(2017, 10, 28), Duration = 4, Progress = 80,
Priority = "Critical", ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child Task
2", StartDate = new DateTime(2017, 10, 11), Duration = 5, Progress = 65,
Priority = "Low", ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child task
3", StartDate = new DateTime(2017, 10, 14), Duration = 6, Progress = 77,
Priority = "High", ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child task
4", StartDate = new DateTime(2017, 10, 17), Duration = 6, Progress = 77,
Priority = "Low", ParentId = 5 });
return TreeDataCollection;
}
}
}

```

JSON

```

{
  "de-DE": {
    "treegrid": {
      "EmptyRecord": "Keine Aufzeichnungen angezeigt",
      "ExpandAll": "Alle erweitern",
      "CollapseAll": "Alles einklappen",
      "Print": "Drucken",
      "Pdfexport": "PDF-Export",
      "Excelexport": "Excel-Export",
      "Wordexport": "Word-Export",
      "FilterButton": "Filter",
      "ClearButton": "klar",
      "StartsWith": "Beginnt mit",
      "EndsWith": "Endet mit",
      "Contains": "Enthält",
      "Equal": "Gleich",
      "NotEqual": "Nicht gleich",
      "LessThan": "Weniger als",
      "LessThanOrEqual": "Weniger als oder gleich",
      "GreaterThan": "Größer als",
      "GreaterThanOrEqual": "Größer als oder gleich",
      "EnterValue": "Geben Sie den Wert ein",
      "FilterMenu": "Filter"
    },
    "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": "Zum nächsten Pager",
"previousPagerTooltip": "Zum vorherigen Pager"
},
"dropdowns": {
"noRecordsTemplate": "Keine Aufzeichnungen gefunden"
},
"datepicker": {
"placeholder": "Wählen Sie ein Datum",
"today": "heute"
}
}
}

```

JSON

```

{
  "main": {
    "de": {
      "identity": {
        "version": {
          "_number": "$Revision: 14842 $",
          "_cldrVersion": "35.1"
        },
        "language": "de"
      },
      "numbers": {
        "defaultNumberingSystem": "latn",
        "otherNumberingSystems": {
          "native": "latn"
        },
        "minimumGroupingDigits": "1",
        "symbols-numberSystem-latn": {
          "decimal": ",",
          "group": ".",
          "list": ";",
          "percentSign": "%",
          "plusSign": "+",
          "minusSign": "-",
          "exponential": "E",
          "superscriptingExponent": "°",
          "perMille": "‰",
          "infinity": "∞",
          "nan": "NaN",
          "timeSeparator": ":"
        },
        "decimalFormats-numberSystem-latn": {
          "standard": "#,##0.###",
          "long": {
            "decimalFormat": {
              "1000-count-one": "0 Tausend",

```

```

"1000-count-other": "0 Tausend",
"10000-count-one": "00 Tausend",
"10000-count-other": "00 Tausend",
"100000-count-one": "000 Tausend",
"100000-count-other": "000 Tausend",
"1000000-count-one": "0 Million",
"1000000-count-other": "0 Millionen",
"10000000-count-one": "00 Millionen",
"10000000-count-other": "00 Millionen",
"100000000-count-one": "000 Millionen",
"100000000-count-other": "000 Millionen",
"1000000000-count-one": "0 Milliarde",
"1000000000-count-other": "0 Milliarden",
"10000000000-count-one": "00 Milliarden",
"10000000000-count-other": "00 Milliarden",
"100000000000-count-one": "000 Milliarden",
"100000000000-count-other": "000 Milliarden",
"1000000000000-count-one": "0 Billion",
"1000000000000-count-other": "0 Billionen",
"10000000000000-count-one": "00 Billionen",
"10000000000000-count-other": "00 Billionen",
"100000000000000-count-one": "000 Billionen",
"100000000000000-count-other": "000 Billionen"
}
},
"short": {
  "decimalFormat": {
    "1000-count-one": "0",
    "1000-count-other": "0",
    "10000-count-one": "0",
    "10000-count-other": "0",
    "100000-count-one": "0",
    "100000-count-other": "0",
    "1000000-count-one": "0 Mio'.'",
    "1000000-count-other": "0 Mio'.'",
    "10000000-count-one": "00 Mio'.'",
    "10000000-count-other": "00 Mio'.'",
    "100000000-count-one": "000 Mio'.'",
    "100000000-count-other": "000 Mio'.'",
    "1000000000-count-one": "0 Mrd'.'",
    "1000000000-count-other": "0 Mrd'.'",
    "10000000000-count-one": "00 Mrd'.'",
    "10000000000-count-other": "00 Mrd'.'",
    "100000000000-count-one": "000 Mrd'.'",
    "100000000000-count-other": "000 Mrd'.'",
    "1000000000000-count-one": "0 Bio'.'",
    "1000000000000-count-other": "0 Bio'.'",
    "10000000000000-count-one": "00 Bio'.'",
    "10000000000000-count-other": "00 Bio'.'",
    "100000000000000-count-one": "000 Bio'.'",
    "100000000000000-count-other": "000 Bio'.'"
  }
}
},
"scientificFormats-numberSystem-latn": {
  "standard": "#E0"
},

```

```

"percentFormats-numberSystem-latn": {
  "standard": "#,##0 %"
},
"currencyFormats-numberSystem-latn": {
  "currencySpacing": {
    "beforeCurrency": {
      "currencyMatch": "[:^S:]",
      "surroundingMatch": "[:digit:]",
      "insertBetween": " "
    },
    "afterCurrency": {
      "currencyMatch": "[:^S:]",
      "surroundingMatch": "[:digit:]",
      "insertBetween": " "
    }
  },
  "standard": "#,##0.00 ¤",
  "accounting": "#,##0.00 ¤",
  "short": {
    "standard": {
      "1000-count-one": "0",
      "1000-count-other": "0",
      "10000-count-one": "0",
      "10000-count-other": "0",
      "100000-count-one": "0",
      "100000-count-other": "0",
      "1000000-count-one": "0 Mio'.' ¤",
      "1000000-count-other": "0 Mio'.' ¤",
      "10000000-count-one": "00 Mio'.' ¤",
      "10000000-count-other": "00 Mio'.' ¤",
      "100000000-count-one": "000 Mio'.' ¤",
      "100000000-count-other": "000 Mio'.' ¤",
      "1000000000-count-one": "0 Mrd'.' ¤",
      "1000000000-count-other": "0 Mrd'.' ¤",
      "10000000000-count-one": "00 Mrd'.' ¤",
      "10000000000-count-other": "00 Mrd'.' ¤",
      "100000000000-count-one": "000 Mrd'.' ¤",
      "100000000000-count-other": "000 Mrd'.' ¤",
      "1000000000000-count-one": "0 Bio'.' ¤",
      "1000000000000-count-other": "0 Bio'.' ¤",
      "10000000000000-count-one": "00 Bio'.' ¤",
      "10000000000000-count-other": "00 Bio'.' ¤",
      "100000000000000-count-one": "000 Bio'.' ¤",
      "100000000000000-count-other": "000 Bio'.' ¤"
    }
  },
  "unitPattern-count-one": "{0} {1}",
  "unitPattern-count-other": "{0} {1}"
},
"miscPatterns-numberSystem-latn": {
  "approximately": "≈{0}",
  "atLeast": "{0}+",
  "atMost": "≤{0}",
  "range": "{0}-{1}"
},
"minimalPairs": {
  "pluralMinimalPairs-count-one": "{0} Tag",

```

```

"pluralMinimalPairs-count-other": "{0} Tage",
"other": "{0}. Abzweigung nach rechts nehmen"
}
}
}
}
}
}

```

JSON

```

{
  "main": {
    "de": {
      "identity": {
        "version": {
          "_number": "$Revision: 14842 $",
          "_cldrVersion": "35.1"
        },
        "language": "de"
      },
      "numbers": {
        "currencies": {
          "ADP": {
            "displayName": "Andorranische Pesete",
            "displayName-count-one": "Andorranische Pesete",
            "displayName-count-other": "Andorranische Peseten",
            "symbol": "ADP"
          },
          "AED": {
            "displayName": "VAE-Dirham",
            "displayName-count-one": "VAE-Dirham",
            "displayName-count-other": "VAE-Dirham",
            "symbol": "AED"
          },
          "AFA": {
            "displayName": "Afghanische Afghani (1927-2002)",
            "displayName-count-one": "Afghanische Afghani (1927-2002)",
            "displayName-count-other": "Afghanische Afghani (1927-2002)",
            "symbol": "AFA"
          },
          "AFN": {
            "displayName": "Afghanischer Afghani",
            "displayName-count-one": "Afghanischer Afghani",
            "displayName-count-other": "Afghanische Afghani",
            "symbol": "AFN"
          },
          "ALK": {
            "displayName": "Albanischer Lek (1946-1965)",
            "displayName-count-one": "Albanischer Lek (1946-1965)",
            "displayName-count-other": "Albanische Lek (1946-1965)",
            "symbol": "ALK"
          },
          "ALL": {
            "displayName": "Albanischer Lek",
            "displayName-count-one": "Albanischer Lek",
            "displayName-count-other": "Albanische Lek",

```

```

"symbol": "ALL"
},
"AMD": {
"displayname": "Armenischer Dram",
"displayname-count-one": "Armenischer Dram",
"displayname-count-other": "Armenische Dram",
"symbol": "AMD"
},
"ANG": {
"displayname": "Niederländische-Antillen-Gulden",
"displayname-count-one": "Niederländische-Antillen-Gulden",
"displayname-count-other": "Niederländische-Antillen-Gulden",
"symbol": "ANG"
},
"AOA": {
"displayname": "Angolanischer Kwanza",
"displayname-count-one": "Angolanischer Kwanza",
"displayname-count-other": "Angolanische Kwanza",
"symbol": "AOA",
"symbol-alt-narrow": "Kz"
},
"AOK": {
"displayname": "Angolanischer Kwanza (1977-1990)",
"displayname-count-one": "Angolanischer Kwanza (1977-1990)",
"displayname-count-other": "Angolanische Kwanza (1977-1990)",
"symbol": "AOK"
},
"AON": {
"displayname": "Angolanischer Neuer Kwanza (1990-2000)",
"displayname-count-one": "Angolanischer Neuer Kwanza (1990-2000)",
"displayname-count-other": "Angolanische Neue Kwanza (1990-2000)",
"symbol": "AON"
},
"AOR": {
"displayname": "Angolanischer Kwanza Reajustado (1995-1999)",
"displayname-count-one": "Angolanischer Kwanza Reajustado (1995-1999)",
"displayname-count-other": "Angolanische Kwanza Reajustado (1995-1999)",
"symbol": "AOR"
},
"ARA": {
"displayname": "Argentinischer Austral",
"displayname-count-one": "Argentinischer Austral",
"displayname-count-other": "Argentinische Austral",
"symbol": "ARA"
},
"ARL": {
"displayname": "Argentinischer Peso Ley (1970-1983)",
"displayname-count-one": "Argentinischer Peso Ley (1970-1983)",
"displayname-count-other": "Argentinische Pesos Ley (1970-1983)",
"symbol": "ARL"
},
"ARM": {
"displayname": "Argentinischer Peso (1881-1970)",
"displayname-count-one": "Argentinischer Peso (1881-1970)",
"displayname-count-other": "Argentinische Pesos (1881-1970)",
"symbol": "ARM"
},

```

```

"ARP": {
  "displayName": "Argentinischer Peso (1983-1985)",
  "displayName-count-one": "Argentinischer Peso (1983-1985)",
  "displayName-count-other": "Argentinische Peso (1983-1985)",
  "symbol": "ARP"
},
"ARS": {
  "displayName": "Argentinischer Peso",
  "displayName-count-one": "Argentinischer Peso",
  "displayName-count-other": "Argentinische Pesos",
  "symbol": "ARS",
  "symbol-alt-narrow": "$"
},
"ATS": {
  "displayName": "Österreichischer Schilling",
  "displayName-count-one": "Österreichischer Schilling",
  "displayName-count-other": "Österreichische Schilling",
  "symbol": "öS"
},
"AUD": {
  "displayName": "Australischer Dollar",
  "displayName-count-one": "Australischer Dollar",
  "displayName-count-other": "Australische Dollar",
  "symbol": "AU$",
  "symbol-alt-narrow": "$"
},
"AWG": {
  "displayName": "Aruba-Florin",
  "displayName-count-one": "Aruba-Florin",
  "displayName-count-other": "Aruba-Florin",
  "symbol": "AWG"
},
"AZM": {
  "displayName": "Aserbaidtschan-Manat (1993-2006)",
  "displayName-count-one": "Aserbaidtschan-Manat (1993-2006)",
  "displayName-count-other": "Aserbaidtschan-Manat (1993-2006)",
  "symbol": "AZM"
},
"AZN": {
  "displayName": "Aserbaidtschan-Manat",
  "displayName-count-one": "Aserbaidtschan-Manat",
  "displayName-count-other": "Aserbaidtschan-Manat",
  "symbol": "AZN"
},
"BAD": {
  "displayName": "Bosnien und Herzegowina Dinar (1992-1994)",
  "displayName-count-one": "Bosnien und Herzegowina Dinar (1992-1994)",
  "displayName-count-other": "Bosnien und Herzegowina Dinar (1992-1994)",
  "symbol": "BAD"
},
"BAM": {
  "displayName": "Bosnien und Herzegowina Konvertierbare Mark",
  "displayName-count-one": "Bosnien und Herzegowina Konvertierbare Mark",
  "displayName-count-other": "Bosnien und Herzegowina Konvertierbare Mark",
  "symbol": "BAM",
  "symbol-alt-narrow": "KM"
},

```



```
"BAN": {
  "displayName": "Bosnien und Herzegowina Neuer Dinar (1994-1997)",
  "displayName-count-one": "Bosnien und Herzegowina Neuer Dinar (1994-1997)",
  "displayName-count-other": "Bosnien und Herzegowina Neue Dinar (1994-1997)",
  "symbol": "BAN"
},
"BBD": {
  "displayName": "Barbados-Dollar",
  "displayName-count-one": "Barbados-Dollar",
  "displayName-count-other": "Barbados-Dollar",
  "symbol": "BBD",
  "symbol-alt-narrow": "$"
},
"BDT": {
  "displayName": "Bangladesch-Taka",
  "displayName-count-one": "Bangladesch-Taka",
  "displayName-count-other": "Bangladesch-Taka",
  "symbol": "BDT",
  "symbol-alt-narrow": "৳"
},
"BEC": {
  "displayName": "Belgischer Franc (konvertibel)",
  "displayName-count-one": "Belgischer Franc (konvertibel)",
  "displayName-count-other": "Belgische Franc (konvertibel)",
  "symbol": "BEC"
},
"BEF": {
  "displayName": "Belgischer Franc",
  "displayName-count-one": "Belgischer Franc",
  "displayName-count-other": "Belgische Franc",
  "symbol": "BEF"
},
"BEL": {
  "displayName": "Belgischer Finanz-Franc",
  "displayName-count-one": "Belgischer Finanz-Franc",
  "displayName-count-other": "Belgische Finanz-Franc",
  "symbol": "BEL"
},
"BGL": {
  "displayName": "Bulgarische Lew (1962-1999)",
  "displayName-count-one": "Bulgarische Lew (1962-1999)",
  "displayName-count-other": "Bulgarische Lew (1962-1999)",
  "symbol": "BGL"
},
"BGM": {
  "displayName": "Bulgarischer Lew (1952-1962)",
  "displayName-count-one": "Bulgarischer Lew (1952-1962)",
  "displayName-count-other": "Bulgarische Lew (1952-1962)",
  "symbol": "BGK"
},
"BGN": {
  "displayName": "Bulgarischer Lew",
  "displayName-count-one": "Bulgarischer Lew",
  "displayName-count-other": "Bulgarische Lew",
  "symbol": "BGN"
},
}
```

```

"BGO": {
  "displayName": "Bulgarischer Lew (1879-1952)",
  "displayName-count-one": "Bulgarischer Lew (1879-1952)",
  "displayName-count-other": "Bulgarische Lew (1879-1952)",
  "symbol": "BGJ"
},
"BHD": {
  "displayName": "Bahrain-Dinar",
  "displayName-count-one": "Bahrain-Dinar",
  "displayName-count-other": "Bahrain-Dinar",
  "symbol": "BHD"
},
"BIF": {
  "displayName": "Burundi-Franc",
  "displayName-count-one": "Burundi-Franc",
  "displayName-count-other": "Burundi-Francs",
  "symbol": "BIF"
},
"BMD": {
  "displayName": "Bermuda-Dollar",
  "displayName-count-one": "Bermuda-Dollar",
  "displayName-count-other": "Bermuda-Dollar",
  "symbol": "BMD",
  "symbol-alt-narrow": "$"
},
"BND": {
  "displayName": "Brunei-Dollar",
  "displayName-count-one": "Brunei-Dollar",
  "displayName-count-other": "Brunei-Dollar",
  "symbol": "BND",
  "symbol-alt-narrow": "$"
},
"BOB": {
  "displayName": "Bolivianischer Boliviano",
  "displayName-count-one": "Bolivianischer Boliviano",
  "displayName-count-other": "Bolivianische Bolivianos",
  "symbol": "BOB",
  "symbol-alt-narrow": "Bs"
},
"BOL": {
  "displayName": "Bolivianischer Boliviano (1863-1963)",
  "displayName-count-one": "Bolivianischer Boliviano (1863-1963)",
  "displayName-count-other": "Bolivianische Bolivianos (1863-1963)",
  "symbol": "BOL"
},
"BOP": {
  "displayName": "Bolivianischer Peso",
  "displayName-count-one": "Bolivianischer Peso",
  "displayName-count-other": "Bolivianische Peso",
  "symbol": "BOP"
},
"BOV": {
  "displayName": "Boliviansiche Mvdol",
  "displayName-count-one": "Boliviansiche Mvdol",
  "displayName-count-other": "Bolivianische Mvdol",
  "symbol": "BOV"
},

```

```

"BRB": {
  "displayName": "Brasilianischer Cruzeiro Novo (1967-1986)",
  "displayName-count-one": "Brasilianischer Cruzeiro Novo (1967-1986)",
  "displayName-count-other": "Brasilianische Cruzeiro Novo (1967-1986)",
  "symbol": "BRB"
},
"BRC": {
  "displayName": "Brasilianischer Cruzado (1986-1989)",
  "displayName-count-one": "Brasilianischer Cruzado (1986-1989)",
  "displayName-count-other": "Brasilianische Cruzado (1986-1989)",
  "symbol": "BRC"
},
"BRE": {
  "displayName": "Brasilianischer Cruzeiro (1990-1993)",
  "displayName-count-one": "Brasilianischer Cruzeiro (1990-1993)",
  "displayName-count-other": "Brasilianische Cruzeiro (1990-1993)",
  "symbol": "BRE"
},
"BRL": {
  "displayName": "Brasilianischer Real",
  "displayName-count-one": "Brasilianischer Real",
  "displayName-count-other": "Brasilianische Real",
  "symbol": "R$",
  "symbol-alt-narrow": "R$"
},
"BRN": {
  "displayName": "Brasilianischer Cruzado Novo (1989-1990)",
  "displayName-count-one": "Brasilianischer Cruzado Novo (1989-1990)",
  "displayName-count-other": "Brasilianische Cruzado Novo (1989-1990)",
  "symbol": "BRN"
},
"BRR": {
  "displayName": "Brasilianischer Cruzeiro (1993-1994)",
  "displayName-count-one": "Brasilianischer Cruzeiro (1993-1994)",
  "displayName-count-other": "Brasilianische Cruzeiro (1993-1994)",
  "symbol": "BRR"
},
"BRZ": {
  "displayName": "Brasilianischer Cruzeiro (1942-1967)",
  "displayName-count-one": "Brasilianischer Cruzeiro (1942-1967)",
  "displayName-count-other": "Brasilianischer Cruzeiro (1942-1967)",
  "symbol": "BRZ"
},
"BSD": {
  "displayName": "Bahamas-Dollar",
  "displayName-count-one": "Bahamas-Dollar",
  "displayName-count-other": "Bahamas-Dollar",
  "symbol": "BSD",
  "symbol-alt-narrow": "$"
},
"BTN": {
  "displayName": "Bhutan-Ngultrum",
  "displayName-count-one": "Bhutan-Ngultrum",
  "displayName-count-other": "Bhutan-Ngultrum",
  "symbol": "BTN"
},
"BUK": {

```

```

"displayName": "Birmanischer Kyat",
"displayName-count-one": "Birmanischer Kyat",
"displayName-count-other": "Birmanische Kyat",
"symbol": "BUK"
},
"BWP": {
"displayName": "Botswanischer Pula",
"displayName-count-one": "Botswanischer Pula",
"displayName-count-other": "Botswanische Pula",
"symbol": "BWP",
"symbol-alt-narrow": "P"
},
"BYB": {
"displayName": "Belarus-Rubel (1994-1999)",
"displayName-count-one": "Belarus-Rubel (1994-1999)",
"displayName-count-other": "Belarus-Rubel (1994-1999)",
"symbol": "BYB"
},
"BYN": {
"displayName": "Weißrussischer Rubel",
"displayName-count-one": "Weißrussischer Rubel",
"displayName-count-other": "Weißrussische Rubel",
"symbol": "BYN",
"symbol-alt-narrow": "p."
},
"BYR": {
"displayName": "Weißrussischer Rubel (2000-2016)",
"displayName-count-one": "Weißrussischer Rubel (2000-2016)",
"displayName-count-other": "Weißrussische Rubel (2000-2016)",
"symbol": "BYR"
},
"BZD": {
"displayName": "Belize-Dollar",
"displayName-count-one": "Belize-Dollar",
"displayName-count-other": "Belize-Dollar",
"symbol": "BZD",
"symbol-alt-narrow": "$"
},
"CAD": {
"displayName": "Kanadischer Dollar",
"displayName-count-one": "Kanadischer Dollar",
"displayName-count-other": "Kanadische Dollar",
"symbol": "CA$",
"symbol-alt-narrow": "$"
},
"CDF": {
"displayName": "Kongo-Franc",
"displayName-count-one": "Kongo-Franc",
"displayName-count-other": "Kongo-Francs",
"symbol": "CDF"
},
"CHE": {
"displayName": "WIR-Euro",
"displayName-count-one": "WIR-Euro",
"displayName-count-other": "WIR-Euro",
"symbol": "CHE"
},

```

```

"CHF": {
  "displayName": "Schweizer Franken",
  "displayName-count-one": "Schweizer Franken",
  "displayName-count-other": "Schweizer Franken",
  "symbol": "CHF"
},
"CHW": {
  "displayName": "WIR Franken",
  "displayName-count-one": "WIR Franken",
  "displayName-count-other": "WIR Franken",
  "symbol": "CHW"
},
"CLE": {
  "displayName": "Chilenischer Escudo",
  "displayName-count-one": "Chilenischer Escudo",
  "displayName-count-other": "Chilenische Escudo",
  "symbol": "CLE"
},
"CLF": {
  "displayName": "Chilenische Unidades de Fomento",
  "displayName-count-one": "Chilenische Unidades de Fomento",
  "displayName-count-other": "Chilenische Unidades de Fomento",
  "symbol": "CLF"
},
"CLP": {
  "displayName": "Chilenischer Peso",
  "displayName-count-one": "Chilenischer Peso",
  "displayName-count-other": "Chilenische Pesos",
  "symbol": "CLP",
  "symbol-alt-narrow": "$"
},
"CNH": {
  "displayName": "Renminbi Yuan (Off-Shore)",
  "displayName-count-one": "Renminbi Yuan (Off-Shore)",
  "displayName-count-other": "Renminbi Yuan (Off-Shore)",
  "symbol": "CNH"
},
"CNX": {
  "displayName": "Dollar der Chinesischen Volksbank",
  "displayName-count-one": "Dollar der Chinesischen Volksbank",
  "displayName-count-other": "Dollar der Chinesischen Volksbank",
  "symbol": "CNX"
},
"CNY": {
  "displayName": "Renminbi Yuan",
  "displayName-count-one": "Chinesischer Yuan",
  "displayName-count-other": "Renminbi Yuan",
  "symbol": "CN¥",
  "symbol-alt-narrow": "¥"
},
"COP": {
  "displayName": "Kolumbianischer Peso",
  "displayName-count-one": "Kolumbianischer Peso",
  "displayName-count-other": "Kolumbianische Pesos",
  "symbol": "COP",
  "symbol-alt-narrow": "$"
},

```

```

"COU": {
  "displayName": "Kolumbianische Unidades de valor real",
  "displayName-count-one": "Kolumbianische Unidad de valor real",
  "displayName-count-other": "Kolumbianische Unidades de valor real",
  "symbol": "COU"
},
"CRC": {
  "displayName": "Costa-Rica-Colón",
  "displayName-count-one": "Costa-Rica-Colón",
  "displayName-count-other": "Costa-Rica-Colón",
  "symbol": "CRC",
  "symbol-alt-narrow": "₡"
},
"CSD": {
  "displayName": "Serbischer Dinar (2002-2006)",
  "displayName-count-one": "Serbischer Dinar (2002-2006)",
  "displayName-count-other": "Serbische Dinar (2002-2006)",
  "symbol": "CSD"
},
"CSK": {
  "displayName": "Tschechoslowakische Krone",
  "displayName-count-one": "Tschechoslowakische Kronen",
  "displayName-count-other": "Tschechoslowakische Kronen",
  "symbol": "CSK"
},
"CUC": {
  "displayName": "Kubanischer Peso (konvertibel)",
  "displayName-count-one": "Kubanischer Peso (konvertibel)",
  "displayName-count-other": "Kubanische Pesos (konvertibel)",
  "symbol": "CUC",
  "symbol-alt-narrow": "Cub$"
},
"CUP": {
  "displayName": "Kubanischer Peso",
  "displayName-count-one": "Kubanischer Peso",
  "displayName-count-other": "Kubanische Pesos",
  "symbol": "CUP",
  "symbol-alt-narrow": "$"
},
"CVE": {
  "displayName": "Cabo-Verde-Escudo",
  "displayName-count-one": "Cabo-Verde-Escudo",
  "displayName-count-other": "Cabo-Verde-Escudos",
  "symbol": "CVE"
},
"CYP": {
  "displayName": "Zypern-Pfund",
  "displayName-count-one": "Zypern Pfund",
  "displayName-count-other": "Zypern Pfund",
  "symbol": "CYP"
},
"CZK": {
  "displayName": "Tschechische Krone",
  "displayName-count-one": "Tschechische Krone",
  "displayName-count-other": "Tschechische Kronen",
  "symbol": "CZK",
  "symbol-alt-narrow": "Kč"
}

```

```

},
"DDM": {
  "displayName": "Mark der DDR",
  "displayName-count-one": "Mark der DDR",
  "displayName-count-other": "Mark der DDR",
  "symbol": "DDM"
},
"DEM": {
  "displayName": "Deutsche Mark",
  "displayName-count-one": "Deutsche Mark",
  "displayName-count-other": "Deutsche Mark",
  "symbol": "DM"
},
"DJF": {
  "displayName": "Dschibuti-Franc",
  "displayName-count-one": "Dschibuti-Franc",
  "displayName-count-other": "Dschibuti-Franc",
  "symbol": "DJF"
},
"DKK": {
  "displayName": "Dänische Krone",
  "displayName-count-one": "Dänische Krone",
  "displayName-count-other": "Dänische Kronen",
  "symbol": "DKK",
  "symbol-alt-narrow": "kr"
},
"DOP": {
  "displayName": "Dominikanischer Peso",
  "displayName-count-one": "Dominikanischer Peso",
  "displayName-count-other": "Dominikanische Pesos",
  "symbol": "DOP",
  "symbol-alt-narrow": "$"
},
"DZD": {
  "displayName": "Algerischer Dinar",
  "displayName-count-one": "Algerischer Dinar",
  "displayName-count-other": "Algerische Dinar",
  "symbol": "DZD"
},
"ECS": {
  "displayName": "Ecuadorianischer Sucre",
  "displayName-count-one": "Ecuadorianischer Sucre",
  "displayName-count-other": "Ecuadorianische Sucre",
  "symbol": "ECS"
},
"ECV": {
  "displayName": "Verrechnungseinheit für Ecuador",
  "displayName-count-one": "Verrechnungseinheiten für Ecuador",
  "displayName-count-other": "Verrechnungseinheiten für Ecuador",
  "symbol": "ECV"
},
"EEK": {
  "displayName": "Estnische Krone",
  "displayName-count-one": "Estnische Krone",
  "displayName-count-other": "Estnische Kronen",
  "symbol": "EEK"
},

```

```

"EGP": {
  "displayName": "Ägyptisches Pfund",
  "displayName-count-one": "Ägyptisches Pfund",
  "displayName-count-other": "Ägyptische Pfund",
  "symbol": "EGP",
  "symbol-alt-narrow": "E£"
},
"ERN": {
  "displayName": "Eritreischer Nakfa",
  "displayName-count-one": "Eritreischer Nakfa",
  "displayName-count-other": "Eritreische Nakfa",
  "symbol": "ERN"
},
"ESA": {
  "displayName": "Spanische Peseta (A-Konten)",
  "displayName-count-one": "Spanische Peseta (A-Konten)",
  "displayName-count-other": "Spanische Peseten (A-Konten)",
  "symbol": "ESA"
},
"ESB": {
  "displayName": "Spanische Peseta (konvertibel)",
  "displayName-count-one": "Spanische Peseta (konvertibel)",
  "displayName-count-other": "Spanische Peseten (konvertibel)",
  "symbol": "ESB"
},
"ESP": {
  "displayName": "Spanische Peseta",
  "displayName-count-one": "Spanische Peseta",
  "displayName-count-other": "Spanische Peseten",
  "symbol": "ESP",
  "symbol-alt-narrow": "₧"
},
"ETB": {
  "displayName": "Äthiopischer Birr",
  "displayName-count-one": "Äthiopischer Birr",
  "displayName-count-other": "Äthiopische Birr",
  "symbol": "ETB"
},
"EUR": {
  "displayName": "Euro",
  "displayName-count-one": "Euro",
  "displayName-count-other": "Euro",
  "symbol": "€",
  "symbol-alt-narrow": "€"
},
"FIM": {
  "displayName": "Finnische Mark",
  "displayName-count-one": "Finnische Mark",
  "displayName-count-other": "Finnische Mark",
  "symbol": "FIM"
},
"FJD": {
  "displayName": "Fidschi-Dollar",
  "displayName-count-one": "Fidschi-Dollar",
  "displayName-count-other": "Fidschi-Dollar",
  "symbol": "FJD",
  "symbol-alt-narrow": "$"
}

```



```

},
"FKP": {
  "displayName": "Falkland-Pfund",
  "displayName-count-one": "Falkland-Pfund",
  "displayName-count-other": "Falkland-Pfund",
  "symbol": "FKP",
  "symbol-alt-narrow": "Fl£"
},
"FRF": {
  "displayName": "Französischer Franc",
  "displayName-count-one": "Französischer Franc",
  "displayName-count-other": "Französische Franc",
  "symbol": "FRF"
},
"GBP": {
  "displayName": "Britisches Pfund",
  "displayName-count-one": "Britisches Pfund",
  "displayName-count-other": "Britische Pfund",
  "symbol": "£",
  "symbol-alt-narrow": "£"
},
"GEK": {
  "displayName": "Georgischer Kupon Larit",
  "displayName-count-one": "Georgischer Kupon Larit",
  "displayName-count-other": "Georgische Kupon Larit",
  "symbol": "GEK"
},
"GEL": {
  "displayName": "Georgischer Lari",
  "displayName-count-one": "Georgischer Lari",
  "displayName-count-other": "Georgische Lari",
  "symbol": "GEL",
  "symbol-alt-narrow": "ლ",
  "symbol-alt-variant": "ლ"
},
"GHC": {
  "displayName": "Ghanaischer Cedi (1979-2007)",
  "displayName-count-one": "Ghanaischer Cedi (1979-2007)",
  "displayName-count-other": "Ghanaische Cedi (1979-2007)",
  "symbol": "GHC"
},
"GHS": {
  "displayName": "Ghanaischer Cedi",
  "displayName-count-one": "Ghanaischer Cedi",
  "displayName-count-other": "Ghanaische Cedi",
  "symbol": "GHS"
},
"GIP": {
  "displayName": "Gibraltar-Pfund",
  "displayName-count-one": "Gibraltar-Pfund",
  "displayName-count-other": "Gibraltar-Pfund",
  "symbol": "GIP",
  "symbol-alt-narrow": "£"
},
"GMD": {
  "displayName": "Gambia-Dalasi",
  "displayName-count-one": "Gambia-Dalasi",

```

```

"displayName-count-other": "Gambia-Dalasi",
"symbol": "GMD"
},
"GNF": {
"displayName": "Guinea-Franc",
"displayName-count-one": "Guinea-Franc",
"displayName-count-other": "Guinea-Franc",
"symbol": "GNF",
"symbol-alt-narrow": "F.G."
},
"GNS": {
"displayName": "Guineischer Syli",
"displayName-count-one": "Guineischer Syli",
"displayName-count-other": "Guineische Syli",
"symbol": "GNS"
},
"GQE": {
"displayName": "Äquatorialguinea-Ekwele",
"displayName-count-one": "Äquatorialguinea-Ekwele",
"displayName-count-other": "Äquatorialguinea-Ekwele",
"symbol": "GQE"
},
"GRD": {
"displayName": "Griechische Drachme",
"displayName-count-one": "Griechische Drachme",
"displayName-count-other": "Griechische Drachmen",
"symbol": "GRD"
},
"GTQ": {
"displayName": "Guatemaltekinscher Quetzal",
"displayName-count-one": "Guatemaltekinscher Quetzal",
"displayName-count-other": "Guatemaltekinsche Quetzales",
"symbol": "GTQ",
"symbol-alt-narrow": "Q"
},
"GWE": {
"displayName": "Portugiesisch Guinea Escudo",
"displayName-count-one": "Portugiesisch Guinea Escudo",
"displayName-count-other": "Portugiesisch Guinea Escudo",
"symbol": "GWE"
},
"GWP": {
"displayName": "Guinea-Bissau Peso",
"displayName-count-one": "Guinea-Bissau Peso",
"displayName-count-other": "Guinea-Bissau Pesos",
"symbol": "GWP"
},
"GYD": {
"displayName": "Guyana-Dollar",
"displayName-count-one": "Guyana-Dollar",
"displayName-count-other": "Guyana-Dollar",
"symbol": "GYD",
"symbol-alt-narrow": "$"
},
"HKD": {
"displayName": "Hongkong-Dollar",
"displayName-count-one": "Hongkong-Dollar",

```

```

"displayName-count-other": "Hongkong-Dollar",
"symbol": "HK$",
"symbol-alt-narrow": "$"
},
"HNL": {
"displayName": "Honduras-Lempira",
"displayName-count-one": "Honduras-Lempira",
"displayName-count-other": "Honduras-Lempira",
"symbol": "HNL",
"symbol-alt-narrow": "L"
},
"HRD": {
"displayName": "Kroatischer Dinar",
"displayName-count-one": "Kroatischer Dinar",
"displayName-count-other": "Kroatische Dinar",
"symbol": "HRD"
},
"HRK": {
"displayName": "Kroatischer Kuna",
"displayName-count-one": "Kroatischer Kuna",
"displayName-count-other": "Kroatische Kuna",
"symbol": "HRK",
"symbol-alt-narrow": "kn"
},
"HTG": {
"displayName": "Haitianische Gourde",
"displayName-count-one": "Haitianische Gourde",
"displayName-count-other": "Haitianische Gourdes",
"symbol": "HTG"
},
"HUF": {
"displayName": "Ungarischer Forint",
"displayName-count-one": "Ungarischer Forint",
"displayName-count-other": "Ungarische Forint",
"symbol": "HUF",
"symbol-alt-narrow": "Ft"
},
>IDR": {
"displayName": "Indonesische Rupiah",
"displayName-count-one": "Indonesische Rupiah",
"displayName-count-other": "Indonesische Rupiah",
"symbol": "IDR",
"symbol-alt-narrow": "Rp"
},
"IEP": {
"displayName": "Irisches Pfund",
"displayName-count-one": "Irisches Pfund",
"displayName-count-other": "Irische Pfund",
"symbol": "IEP"
},
"ILP": {
"displayName": "Israelisches Pfund",
"displayName-count-one": "Israelisches Pfund",
"displayName-count-other": "Israelische Pfund",
"symbol": "ILP"
},
"ILR": {

```

```

"displayName": "Israelischer Schekel (1980-1985)",
"displayName-count-one": "Israelischer Schekel (1980-1985)",
"displayName-count-other": "Israelische Schekel (1980-1985)",
"symbol": "ILR"
},
"ILS": {
"displayName": "Israelischer Neuer Schekel",
"displayName-count-one": "Israelischer Neuer Schekel",
"displayName-count-other": "Israelische Neue Schekel",
"symbol": "₪",
"symbol-alt-narrow": "₪"
},
"INR": {
"displayName": "Indische Rupie",
"displayName-count-one": "Indische Rupie",
"displayName-count-other": "Indische Rupien",
"symbol": "₹",
"symbol-alt-narrow": "₹"
},
"IQD": {
"displayName": "Irakischer Dinar",
"displayName-count-one": "Irakischer Dinar",
"displayName-count-other": "Irakische Dinar",
"symbol": "IQD"
},
"IRR": {
"displayName": "Iranischer Rial",
"displayName-count-one": "Iranischer Rial",
"displayName-count-other": "Iranische Rial",
"symbol": "IRR"
},
"ISJ": {
"displayName": "Isländische Krone (1918-1981)",
"displayName-count-one": "Isländische Krone (1918-1981)",
"displayName-count-other": "Isländische Kronen (1918-1981)",
"symbol": "ISJ"
},
"ISK": {
"displayName": "Isländische Krone",
"displayName-count-one": "Isländische Krone",
"displayName-count-other": "Isländische Kronen",
"symbol": "ISK",
"symbol-alt-narrow": "kr"
},
"ITL": {
"displayName": "Italienische Lira",
"displayName-count-one": "Italienische Lira",
"displayName-count-other": "Italienische Lire",
"symbol": "ITL"
},
"JMD": {
"displayName": "Jamaika-Dollar",
"displayName-count-one": "Jamaika-Dollar",
"displayName-count-other": "Jamaika-Dollar",
"symbol": "JMD",
"symbol-alt-narrow": "$"
},

```

```

"JOD": {
  "displayName": "Jordanischer Dinar",
  "displayName-count-one": "Jordanischer Dinar",
  "displayName-count-other": "Jordanische Dinar",
  "symbol": "JOD"
},
"JPY": {
  "displayName": "Japanischer Yen",
  "displayName-count-one": "Japanischer Yen",
  "displayName-count-other": "Japanische Yen",
  "symbol": "¥",
  "symbol-alt-narrow": "¥"
},
"KES": {
  "displayName": "Kenia-Schilling",
  "displayName-count-one": "Kenia-Schilling",
  "displayName-count-other": "Kenia-Schilling",
  "symbol": "KES"
},
"KGS": {
  "displayName": "Kirgisischer Som",
  "displayName-count-one": "Kirgisischer Som",
  "displayName-count-other": "Kirgisische Som",
  "symbol": "KGS"
},
"KHR": {
  "displayName": "Kambodschanischer Riel",
  "displayName-count-one": "Kambodschanischer Riel",
  "displayName-count-other": "Kambodschanische Riel",
  "symbol": "KHR",
  "symbol-alt-narrow": "៛"
},
"KMF": {
  "displayName": "Komoren-Franc",
  "displayName-count-one": "Komoren-Franc",
  "displayName-count-other": "Komoren-Francis",
  "symbol": "KMF",
  "symbol-alt-narrow": "FC"
},
"KPW": {
  "displayName": "Nordkoreanischer Won",
  "displayName-count-one": "Nordkoreanischer Won",
  "displayName-count-other": "Nordkoreanische Won",
  "symbol": "KPW",
  "symbol-alt-narrow": "₩"
},
"KRH": {
  "displayName": "Südkoreanischer Hwan (1953-1962)",
  "displayName-count-one": "Südkoreanischer Hwan (1953-1962)",
  "displayName-count-other": "Südkoreanischer Hwan (1953-1962)",
  "symbol": "KRH"
},
"KRO": {
  "displayName": "Südkoreanischer Won (1945-1953)",
  "displayName-count-one": "Südkoreanischer Won (1945-1953)",
  "displayName-count-other": "Südkoreanischer Won (1945-1953)",

```

```

"symbol": "KRO"
},
"KRW": {
"displayname": "Südkoreanischer Won",
"displayname-count-one": "Südkoreanischer Won",
"displayname-count-other": "Südkoreanische Won",
"symbol": "₩",
"symbol-alt-narrow": "₩"
},
"KWD": {
"displayname": "Kuwait-Dinar",
"displayname-count-one": "Kuwait-Dinar",
"displayname-count-other": "Kuwait-Dinar",
"symbol": "KWD"
},
"KYD": {
"displayname": "Kaiman-Dollar",
"displayname-count-one": "Kaiman-Dollar",
"displayname-count-other": "Kaiman-Dollar",
"symbol": "KYD",
"symbol-alt-narrow": "$"
},
"KZT": {
"displayname": "Kasachischer Tenge",
"displayname-count-one": "Kasachischer Tenge",
"displayname-count-other": "Kasachische Tenge",
"symbol": "KZT",
"symbol-alt-narrow": "₸"
},
"LAK": {
"displayname": "Laotischer Kip",
"displayname-count-one": "Laotischer Kip",
"displayname-count-other": "Laotische Kip",
"symbol": "LAK",
"symbol-alt-narrow": "₭"
},
"LBP": {
"displayname": "Libanesisches Pfund",
"displayname-count-one": "Libanesisches Pfund",
"displayname-count-other": "Libanesische Pfund",
"symbol": "LBP",
"symbol-alt-narrow": "L£"
},
"LKR": {
"displayname": "Sri-Lanka-Rupie",
"displayname-count-one": "Sri-Lanka-Rupie",
"displayname-count-other": "Sri-Lanka-Rupien",
"symbol": "LKR",
"symbol-alt-narrow": "Rs"
},
"LRD": {
"displayname": "Liberianischer Dollar",
"displayname-count-one": "Liberianischer Dollar",
"displayname-count-other": "Liberianische Dollar",
"symbol": "LRD",
"symbol-alt-narrow": "$"
},

```

```
"LSL": {
  "displayName": "Loti",
  "displayName-count-one": "Loti",
  "displayName-count-other": "Loti",
  "symbol": "LSL"
},
"LTL": {
  "displayName": "Litauischer Litas",
  "displayName-count-one": "Litauischer Litas",
  "displayName-count-other": "Litauische Litas",
  "symbol": "LTL",
  "symbol-alt-narrow": "Lt"
},
"LTT": {
  "displayName": "Litauischer Talonas",
  "displayName-count-one": "Litauische Talonas",
  "displayName-count-other": "Litauische Talonas",
  "symbol": "LTT"
},
"LUC": {
  "displayName": "Luxemburgischer Franc (konvertibel)",
  "displayName-count-one": "Luxemburgische Franc (konvertibel)",
  "displayName-count-other": "Luxemburgische Franc (konvertibel)",
  "symbol": "LUC"
},
"LUF": {
  "displayName": "Luxemburgischer Franc",
  "displayName-count-one": "Luxemburgische Franc",
  "displayName-count-other": "Luxemburgische Franc",
  "symbol": "LUF"
},
"LUL": {
  "displayName": "Luxemburgischer Finanz-Franc",
  "displayName-count-one": "Luxemburgische Finanz-Franc",
  "displayName-count-other": "Luxemburgische Finanz-Franc",
  "symbol": "LUL"
},
"LVL": {
  "displayName": "Lettischer Lats",
  "displayName-count-one": "Lettischer Lats",
  "displayName-count-other": "Lettische Lats",
  "symbol": "LVL",
  "symbol-alt-narrow": "Ls"
},
"LVR": {
  "displayName": "Lettischer Rubel",
  "displayName-count-one": "Lettische Rubel",
  "displayName-count-other": "Lettische Rubel",
  "symbol": "LVR"
},
"LYD": {
  "displayName": "Libyscher Dinar",
  "displayName-count-one": "Libyscher Dinar",
  "displayName-count-other": "Libysche Dinar",
  "symbol": "LYD"
},
"MAD": {
```

```

"displayName": "Marokkanischer Dirham",
"displayName-count-one": "Marokkanischer Dirham",
"displayName-count-other": "Marokkanische Dirham",
"symbol": "MAD"
},
"MAF": {
"displayName": "Marokkanischer Franc",
"displayName-count-one": "Marokkanische Franc",
"displayName-count-other": "Marokkanische Franc",
"symbol": "MAF"
},
"MCF": {
"displayName": "Monegassischer Franc",
"displayName-count-one": "Monegassischer Franc",
"displayName-count-other": "Monegassische Franc",
"symbol": "MCF"
},
"MDC": {
"displayName": "Moldau-Cupon",
"displayName-count-one": "Moldau-Cupon",
"displayName-count-other": "Moldau-Cupon",
"symbol": "MDC"
},
"MDL": {
"displayName": "Moldau-Leu",
"displayName-count-one": "Moldau-Leu",
"displayName-count-other": "Moldau-Leu",
"symbol": "MDL"
},
"MGA": {
"displayName": "Madagaskar-Ariary",
"displayName-count-one": "Madagaskar-Ariary",
"displayName-count-other": "Madagaskar-Ariary",
"symbol": "MGA",
"symbol-alt-narrow": "Ar"
},
"MGF": {
"displayName": "Madagaskar-Franc",
"displayName-count-one": "Madagaskar-Franc",
"displayName-count-other": "Madagaskar-Franc",
"symbol": "MGF"
},
"MKD": {
"displayName": "Mazedonischer Denar",
"displayName-count-one": "Mazedonischer Denar",
"displayName-count-other": "Mazedonische Denari",
"symbol": "MKD"
},
"MKN": {
"displayName": "Mazedonischer Denar (1992-1993)",
"displayName-count-one": "Mazedonischer Denar (1992-1993)",
"displayName-count-other": "Mazedonische Denar (1992-1993)",
"symbol": "MKN"
},
"MLF": {
"displayName": "Malischer Franc",
"displayName-count-one": "Malische Franc",

```



```

"displayName-count-other": "Malische Franc",
"symbol": "MLF"
},
"MMK": {
"displayName": "Myanmarischer Kyat",
"displayName-count-one": "Myanmarischer Kyat",
"displayName-count-other": "Myanmarische Kyat",
"symbol": "MMK",
"symbol-alt-narrow": "K"
},
"MNT": {
"displayName": "Mongolischer Tögrög",
"displayName-count-one": "Mongolischer Tögrög",
"displayName-count-other": "Mongolische Tögrög",
"symbol": "MNT",
"symbol-alt-narrow": "₮"
},
"MOP": {
"displayName": "Macao-Pataca",
"displayName-count-one": "Macao-Pataca",
"displayName-count-other": "Macao-Pataca",
"symbol": "MOP"
},
"MRO": {
"displayName": "Mauretanischer Ouguiya (1973-2017)",
"displayName-count-one": "Mauretanischer Ouguiya (1973-2017)",
"displayName-count-other": "Mauretanische Ouguiya (1973-2017)",
"symbol": "MRO"
},
"MRU": {
"displayName": "Mauretanischer Ouguiya",
"displayName-count-one": "Mauretanischer Ouguiya",
"displayName-count-other": "Mauretanische Ouguiya",
"symbol": "MRU"
},
"MTL": {
"displayName": "Maltesische Lira",
"displayName-count-one": "Maltesische Lira",
"displayName-count-other": "Maltesische Lira",
"symbol": "MTL"
},
"MTP": {
"displayName": "Maltesisches Pfund",
"displayName-count-one": "Maltesische Pfund",
"displayName-count-other": "Maltesische Pfund",
"symbol": "MTP"
},
"MUR": {
"displayName": "Mauritius-Rupie",
"displayName-count-one": "Mauritius-Rupie",
"displayName-count-other": "Mauritius-Rupien",
"symbol": "MUR",
"symbol-alt-narrow": "Rs"
},
"MVP": {
"displayName": "Malediven-Rupie (alt)",
"displayName-count-one": "Malediven-Rupie (alt)",

```

```

"displayName-count-other": "Malediven-Rupien (alt)",
"symbol": "MVP"
},
"MVR": {
"displayName": "Malediven-Rufiyaa",
"displayName-count-one": "Malediven-Rufiyaa",
"displayName-count-other": "Malediven-Rupien",
"symbol": "MVR"
},
"MWK": {
"displayName": "Malawi-Kwacha",
"displayName-count-one": "Malawi-Kwacha",
"displayName-count-other": "Malawi-Kwacha",
"symbol": "MWK"
},
"MXN": {
"displayName": "Mexikanischer Peso",
"displayName-count-one": "Mexikanischer Peso",
"displayName-count-other": "Mexikanische Pesos",
"symbol": "MX$",
"symbol-alt-narrow": "$"
},
"MXP": {
"displayName": "Mexikanischer Silber-Peso (1861-1992)",
"displayName-count-one": "Mexikanische Silber-Peso (1861-1992)",
"displayName-count-other": "Mexikanische Silber-Pesos (1861-1992)",
"symbol": "MXP"
},
"MXV": {
"displayName": "Mexicanischer Unidad de Inversion (UDI)",
"displayName-count-one": "Mexicanischer Unidad de Inversion (UDI)",
"displayName-count-other": "Mexikanische Unidad de Inversion (UDI)",
"symbol": "MXV"
},
"MYR": {
"displayName": "Malaysischer Ringgit",
"displayName-count-one": "Malaysischer Ringgit",
"displayName-count-other": "Malaysische Ringgit",
"symbol": "MYR",
"symbol-alt-narrow": "RM"
},
"MZE": {
"displayName": "Mosambikanischer Escudo",
"displayName-count-one": "Mozambikanische Escudo",
"displayName-count-other": "Mozambikanische Escudo",
"symbol": "MZE"
},
"MZM": {
"displayName": "Mosambikanischer Metical (1980-2006)",
"displayName-count-one": "Mosambikanischer Metical (1980-2006)",
"displayName-count-other": "Mosambikanische Meticais (1980-2006)",
"symbol": "MZM"
},
"MZN": {
"displayName": "Mosambikanischer Metical",
"displayName-count-one": "Mosambikanischer Metical",
"displayName-count-other": "Mosambikanische Meticais",

```

```

"symbol": "MZN"
},
"NAD": {
"displayname": "Namibia-Dollar",
"displayname-count-one": "Namibia-Dollar",
"displayname-count-other": "Namibia-Dollar",
"symbol": "NAD",
"symbol-alt-narrow": "$"
},
"NGN": {
"displayname": "Nigerianischer Naira",
"displayname-count-one": "Nigerianischer Naira",
"displayname-count-other": "Nigerianische Naira",
"symbol": "NGN",
"symbol-alt-narrow": "₦"
},
"NIC": {
"displayname": "Nicaraguanischer Córdoba (1988-1991)",
"displayname-count-one": "Nicaraguanischer Córdoba (1988-1991)",
"displayname-count-other": "Nicaraguanische Córdoba (1988-1991)",
"symbol": "NIC"
},
"NIO": {
"displayname": "Nicaragua-Córdoba",
"displayname-count-one": "Nicaragua-Córdoba",
"displayname-count-other": "Nicaragua-Córdobas",
"symbol": "NIO",
"symbol-alt-narrow": "C$"
},
"NLG": {
"displayname": "Niederländischer Gulden",
"displayname-count-one": "Niederländischer Gulden",
"displayname-count-other": "Niederländische Gulden",
"symbol": "NLG"
},
"NOK": {
"displayname": "Norwegische Krone",
"displayname-count-one": "Norwegische Krone",
"displayname-count-other": "Norwegische Kronen",
"symbol": "NOK",
"symbol-alt-narrow": "kr"
},
"NPR": {
"displayname": "Nepalesische Rupie",
"displayname-count-one": "Nepalesische Rupie",
"displayname-count-other": "Nepalesische Rupien",
"symbol": "NPR",
"symbol-alt-narrow": "Rs"
},
"NZD": {
"displayname": "Neuseeland-Dollar",
"displayname-count-one": "Neuseeland-Dollar",
"displayname-count-other": "Neuseeland-Dollar",
"symbol": "NZ$",
"symbol-alt-narrow": "$"
},
"OMR": {

```

```

"displayName": "Omanischer Rial",
"displayName-count-one": "Omanischer Rial",
"displayName-count-other": "Omanische Rials",
"symbol": "OMR"
},
"PAB": {
"displayName": "Panamaischer Balboa",
"displayName-count-one": "Panamaischer Balboa",
"displayName-count-other": "Panamaische Balboas",
"symbol": "PAB"
},
"PEI": {
"displayName": "Peruanischer Inti",
"displayName-count-one": "Peruanische Inti",
"displayName-count-other": "Peruanische Inti",
"symbol": "PEI"
},
"PEN": {
"displayName": "Peruanischer Sol",
"displayName-count-one": "Peruanischer Sol",
"displayName-count-other": "Peruanische Sol",
"symbol": "PEN"
},
"PES": {
"displayName": "Peruanischer Sol (1863-1965)",
"displayName-count-one": "Peruanischer Sol (1863-1965)",
"displayName-count-other": "Peruanische Sol (1863-1965)",
"symbol": "PES"
},
"PGK": {
"displayName": "Papua-Neuguineischer Kina",
"displayName-count-one": "Papua-Neuguineischer Kina",
"displayName-count-other": "Papua-Neuguineische Kina",
"symbol": "PGK"
},
"PHP": {
"displayName": "Philippinischer Peso",
"displayName-count-one": "Philippinischer Peso",
"displayName-count-other": "Philippinische Pesos",
"symbol": "PHP",
"symbol-alt-narrow": "₱"
},
"PKR": {
"displayName": "Pakistanische Rupie",
"displayName-count-one": "Pakistanische Rupie",
"displayName-count-other": "Pakistanische Rupien",
"symbol": "PKR",
"symbol-alt-narrow": "Rs"
},
"PLN": {
"displayName": "Polnischer Złoty",
"displayName-count-one": "Polnischer Złoty",
"displayName-count-other": "Polnische Złoty",
"symbol": "PLN",
"symbol-alt-narrow": "zł"
},
"PLZ": {

```

```

"displayName": "Polnischer Zloty (1950-1995)",
"displayName-count-one": "Polnischer Zloty (1950-1995)",
"displayName-count-other": "Polnische Zloty (1950-1995)",
"symbol": "PLZ"
},
"PTE": {
"displayName": "Portugiesischer Escudo",
"displayName-count-one": "Portugiesische Escudo",
"displayName-count-other": "Portugiesische Escudo",
"symbol": "PTE"
},
"PYG": {
"displayName": "Paraguayischer Guaraní",
"displayName-count-one": "Paraguayischer Guaraní",
"displayName-count-other": "Paraguayische Guaranies",
"symbol": "PYG",
"symbol-alt-narrow": "₲"
},
"QAR": {
"displayName": "Katar-Riyal",
"displayName-count-one": "Katar-Riyal",
"displayName-count-other": "Katar-Riyal",
"symbol": "QAR"
},
"RHD": {
"displayName": "Rhodesischer Dollar",
"displayName-count-one": "Rhodesische Dollar",
"displayName-count-other": "Rhodesische Dollar",
"symbol": "RHD"
},
"ROL": {
"displayName": "Rumänischer Leu (1952-2006)",
"displayName-count-one": "Rumänischer Leu (1952-2006)",
"displayName-count-other": "Rumänische Leu (1952-2006)",
"symbol": "ROL"
},
"RON": {
"displayName": "Rumänischer Leu",
"displayName-count-one": "Rumänischer Leu",
"displayName-count-other": "Rumänische Leu",
"symbol": "RON",
"symbol-alt-narrow": "L"
},
"RSD": {
"displayName": "Serbischer Dinar",
"displayName-count-one": "Serbischer Dinar",
"displayName-count-other": "Serbische Dinaren",
"symbol": "RSD"
},
"RUB": {
"displayName": "Russischer Rubel",
"displayName-count-one": "Russischer Rubel",
"displayName-count-other": "Russische Rubel",
"symbol": "RUB",
"symbol-alt-narrow": "₽"
},
"RUR": {

```

```

"displayName": "Russischer Rubel (1991-1998)",
"displayName-count-one": "Russischer Rubel (1991-1998)",
"displayName-count-other": "Russische Rubel (1991-1998)",
"symbol": "RUR",
"symbol-alt-narrow": "p.",
},
"RWF": {
"displayName": "Ruanda-Franc",
"displayName-count-one": "Ruanda-Franc",
"displayName-count-other": "Ruanda-Francs",
"symbol": "RWF",
"symbol-alt-narrow": "F.Rw"
},
"SAR": {
"displayName": "Saudi-Rial",
"displayName-count-one": "Saudi-Rial",
"displayName-count-other": "Saudi-Rial",
"symbol": "SAR"
},
"SBD": {
"displayName": "Salomonen-Dollar",
"displayName-count-one": "Salomonen-Dollar",
"displayName-count-other": "Salomonen-Dollar",
"symbol": "SBD",
"symbol-alt-narrow": "$"
},
"SCR": {
"displayName": "Seychellen-Rupie",
"displayName-count-one": "Seychellen-Rupie",
"displayName-count-other": "Seychellen-Rupien",
"symbol": "SCR"
},
"SDD": {
"displayName": "Sudanesischer Dinar (1992-2007)",
"displayName-count-one": "Sudanesischer Dinar (1992-2007)",
"displayName-count-other": "Sudanesische Dinar (1992-2007)",
"symbol": "SDD"
},
"SDG": {
"displayName": "Sudanesisches Pfund",
"displayName-count-one": "Sudanesisches Pfund",
"displayName-count-other": "Sudanesische Pfund",
"symbol": "SDG"
},
"SDP": {
"displayName": "Sudanesisches Pfund (1957-1998)",
"displayName-count-one": "Sudanesisches Pfund (1957-1998)",
"displayName-count-other": "Sudanesische Pfund (1957-1998)",
"symbol": "SDP"
},
"SEK": {
"displayName": "Schwedische Krone",
"displayName-count-one": "Schwedische Krone",
"displayName-count-other": "Schwedische Kronen",
"symbol": "SEK",
"symbol-alt-narrow": "kr"
},

```

```

"SGD": {
  "displayName": "Singapur-Dollar",
  "displayName-count-one": "Singapur-Dollar",
  "displayName-count-other": "Singapur-Dollar",
  "symbol": "SGD",
  "symbol-alt-narrow": "$"
},
"SHP": {
  "displayName": "St. Helena-Pfund",
  "displayName-count-one": "St. Helena-Pfund",
  "displayName-count-other": "St. Helena-Pfund",
  "symbol": "SHP",
  "symbol-alt-narrow": "£"
},
"SIT": {
  "displayName": "Slowenischer Tolar",
  "displayName-count-one": "Slowenischer Tolar",
  "displayName-count-other": "Slowenische Tolar",
  "symbol": "SIT"
},
"SKK": {
  "displayName": "Slowakische Krone",
  "displayName-count-one": "Slowakische Kronen",
  "displayName-count-other": "Slowakische Kronen",
  "symbol": "SKK"
},
"SLL": {
  "displayName": "Sierra-leonischer Leone",
  "displayName-count-one": "Sierra-leonischer Leone",
  "displayName-count-other": "Sierra-leonische Leones",
  "symbol": "SLL"
},
"SOS": {
  "displayName": "Somalia-Schilling",
  "displayName-count-one": "Somalia-Schilling",
  "displayName-count-other": "Somalia-Schilling",
  "symbol": "SOS"
},
"SRD": {
  "displayName": "Suriname-Dollar",
  "displayName-count-one": "Suriname-Dollar",
  "displayName-count-other": "Suriname-Dollar",
  "symbol": "SRD",
  "symbol-alt-narrow": "$"
},
"SRG": {
  "displayName": "Suriname Gulden",
  "displayName-count-one": "Suriname-Gulden",
  "displayName-count-other": "Suriname-Gulden",
  "symbol": "SRG"
},
"SSP": {
  "displayName": "Südsudanesisches Pfund",
  "displayName-count-one": "Südsudanesisches Pfund",
  "displayName-count-other": "Südsudanesische Pfund",
  "symbol": "SSP",
  "symbol-alt-narrow": "£"
}

```

```

},
"STD": {
  "displayName": "São-toméischer Dobra (1977-2017)",
  "displayName-count-one": "São-toméischer Dobra (1977-2017)",
  "displayName-count-other": "São-toméische Dobra (1977-2017)",
  "symbol": "STD"
},
"STN": {
  "displayName": "São-toméischer Dobra",
  "displayName-count-one": "São-toméischer Dobra",
  "displayName-count-other": "São-toméische Dobras",
  "symbol": "STN",
  "symbol-alt-narrow": "Db"
},
"SUR": {
  "displayName": "Sowjetischer Rubel",
  "displayName-count-one": "Sowjetische Rubel",
  "displayName-count-other": "Sowjetische Rubel",
  "symbol": "SUR"
},
"SVC": {
  "displayName": "El Salvador Colon",
  "displayName-count-one": "El Salvador-Colon",
  "displayName-count-other": "El Salvador-Colon",
  "symbol": "SVC"
},
"SYP": {
  "displayName": "Syrisches Pfund",
  "displayName-count-one": "Syrisches Pfund",
  "displayName-count-other": "Syrische Pfund",
  "symbol": "SYP",
  "symbol-alt-narrow": "SYP"
},
"SZL": {
  "displayName": "Swasiländischer Lilangeni",
  "displayName-count-one": "Swasiländischer Lilangeni",
  "displayName-count-other": "Swasiländische Emalangeni",
  "symbol": "SZL"
},
"THB": {
  "displayName": "Thailändischer Baht",
  "displayName-count-one": "Thailändischer Baht",
  "displayName-count-other": "Thailändische Baht",
  "symbol": "฿",
  "symbol-alt-narrow": "฿"
},
"TJR": {
  "displayName": "Tadschikistan Rubel",
  "displayName-count-one": "Tadschikistan-Rubel",
  "displayName-count-other": "Tadschikistan-Rubel",
  "symbol": "TJR"
},
"TJS": {
  "displayName": "Tadschikistan-Somoni",
  "displayName-count-one": "Tadschikistan-Somoni",
  "displayName-count-other": "Tadschikistan-Somoni",
  "symbol": "TJS"
}

```



```

},
"TMM": {
  "displayName": "Turkmenistan-Manat (1993-2009)",
  "displayName-count-one": "Turkmenistan-Manat (1993-2009)",
  "displayName-count-other": "Turkmenistan-Manat (1993-2009)",
  "symbol": "TMM"
},
"TMT": {
  "displayName": "Turkmenistan-Manat",
  "displayName-count-one": "Turkmenistan-Manat",
  "displayName-count-other": "Turkmenistan-Manat",
  "symbol": "TMT"
},
"TND": {
  "displayName": "Tunesischer Dinar",
  "displayName-count-one": "Tunesischer Dinar",
  "displayName-count-other": "Tunesische Dinar",
  "symbol": "TND"
},
"TOP": {
  "displayName": "Tongaischer Paʻanga",
  "displayName-count-one": "Tongaischer Paʻanga",
  "displayName-count-other": "Tongaische Paʻanga",
  "symbol": "TOP",
  "symbol-alt-narrow": "T$"
},
"TPE": {
  "displayName": "Timor-Escudo",
  "displayName-count-one": "Timor-Escudo",
  "displayName-count-other": "Timor-Escudo",
  "symbol": "TPE"
},
"TRL": {
  "displayName": "Türkische Lira (1922-2005)",
  "displayName-count-one": "Türkische Lira (1922-2005)",
  "displayName-count-other": "Türkische Lira (1922-2005)",
  "symbol": "TRL"
},
"TRY": {
  "displayName": "Türkische Lira",
  "displayName-count-one": "Türkische Lira",
  "displayName-count-other": "Türkische Lira",
  "symbol": "TRY",
  "symbol-alt-narrow": "₺",
  "symbol-alt-variant": "TL"
},
"TTD": {
  "displayName": "Trinidad und Tobago-Dollar",
  "displayName-count-one": "Trinidad und Tobago-Dollar",
  "displayName-count-other": "Trinidad und Tobago-Dollar",
  "symbol": "TTD",
  "symbol-alt-narrow": "$"
},
"USD": {
  "displayName": "Neuer Taiwan-Dollar",
  "displayName-count-one": "Neuer Taiwan-Dollar",
  "displayName-count-other": "Neue Taiwan-Dollar",

```

```

"symbol": "NT$",
"symbol-alt-narrow": "NT$"
},
"TZS": {
"display": "Tansania-Schilling",
"display-count-one": "Tansania-Schilling",
"display-count-other": "Tansania-Schilling",
"symbol": "TZS"
},
"UAH": {
"display": "Ukrainische Hrywnja",
"display-count-one": "Ukrainische Hrywnja",
"display-count-other": "Ukrainische Hrywen",
"symbol": "UAH",
"symbol-alt-narrow": "₴"
},
"UAK": {
"display": "Ukrainischer Karbovanetz",
"display-count-one": "Ukrainische Karbovanetz",
"display-count-other": "Ukrainische Karbovanetz",
"symbol": "UAK"
},
"UGS": {
"display": "Uganda-Schilling (1966-1987)",
"display-count-one": "Uganda-Schilling (1966-1987)",
"display-count-other": "Uganda-Schilling (1966-1987)",
"symbol": "UGS"
},
"UGX": {
"display": "Uganda-Schilling",
"display-count-one": "Uganda-Schilling",
"display-count-other": "Uganda-Schilling",
"symbol": "UGX"
},
"USD": {
"display": "US-Dollar",
"display-count-one": "US-Dollar",
"display-count-other": "US-Dollar",
"symbol": "$",
"symbol-alt-narrow": "$"
},
"USN": {
"display": "US Dollar (Nächster Tag)",
"display-count-one": "US-Dollar (Nächster Tag)",
"display-count-other": "US-Dollar (Nächster Tag)",
"symbol": "USN"
},
"USS": {
"display": "US Dollar (Gleicher Tag)",
"display-count-one": "US-Dollar (Gleicher Tag)",
"display-count-other": "US-Dollar (Gleicher Tag)",
"symbol": "USS"
},
"UYI": {
"display": "Uruguayischer Peso (Indexierte Rechnungseinheiten)",
"display-count-one": "Uruguayischer Peso (Indexierte
Rechnungseinheiten)",

```

```

"displayName-count-other": "Uruguayische Pesos (Indexierte
Rechnungseinheiten)",
"symbol": "UYI"
},
"UYP": {
"displayName": "Uruguayischer Peso (1975-1993)",
"displayName-count-one": "Uruguayischer Peso (1975-1993)",
"displayName-count-other": "Uruguayische Pesos (1975-1993)",
"symbol": "UYP"
},
"UYU": {
"displayName": "Uruguayischer Peso",
"displayName-count-one": "Uruguayischer Peso",
"displayName-count-other": "Uruguayische Pesos",
"symbol": "UYU",
"symbol-alt-narrow": "$"
},
"UYW": {
"displayName": "UYW",
"symbol": "UYW"
},
"UZS": {
"displayName": "Usbekistan-Sum",
"displayName-count-one": "Usbekistan-Sum",
"displayName-count-other": "Usbekistan-Sum",
"symbol": "UZS"
},
"VEB": {
"displayName": "Venezolanischer Bolívar (1871-2008)",
"displayName-count-one": "Venezolanischer Bolívar (1871-2008)",
"displayName-count-other": "Venezolanische Bolívares (1871-2008)",
"symbol": "VEB"
},
"VEF": {
"displayName": "Venezolanischer Bolívar (2008-2018)",
"displayName-count-one": "Venezolanischer Bolívar (2008-2018)",
"displayName-count-other": "Venezolanische Bolívares (2008-2018)",
"symbol": "VEF",
"symbol-alt-narrow": "Bs"
},
"VES": {
"displayName": "Venezolanischer Bolívar",
"displayName-count-one": "Venezolanischer Bolívar",
"displayName-count-other": "Venezolanische Bolívares",
"symbol": "VES"
},
"VND": {
"displayName": "Vietnamesischer Dong",
"displayName-count-one": "Vietnamesischer Dong",
"displayName-count-other": "Vietnamesische Dong",
"symbol": "₫",
"symbol-alt-narrow": "₫"
},
"VNN": {
"displayName": "Vietnamesischer Dong(1978-1985)",
"displayName-count-one": "Vietnamesischer Dong(1978-1985)",
"displayName-count-other": "Vietnamesische Dong(1978-1985)",

```

```

"symbol": "VNN"
},
"VUV": {
"displayname": "Vanuatu-Vatu",
"displayname-count-one": "Vanuatu-Vatu",
"displayname-count-other": "Vanuatu-Vatu",
"symbol": "VUV"
},
"WST": {
"displayname": "Samoanischer Tala",
"displayname-count-one": "Samoanischer Tala",
"displayname-count-other": "Samoanische Tala",
"symbol": "WST"
},
"XAF": {
"displayname": "CFA-Franc (BEAC)",
"displayname-count-one": "CFA-Franc (BEAC)",
"displayname-count-other": "CFA-Franc (BEAC)",
"symbol": "FCFA"
},
"XAG": {
"displayname": "Unze Silber",
"displayname-count-one": "Unze Silber",
"displayname-count-other": "Unzen Silber",
"symbol": "XAG"
},
"XAU": {
"displayname": "Unze Gold",
"displayname-count-one": "Unze Gold",
"displayname-count-other": "Unzen Gold",
"symbol": "XAU"
},
"XBA": {
"displayname": "Europäische Rechnungseinheit",
"displayname-count-one": "Europäische Rechnungseinheiten",
"displayname-count-other": "Europäische Rechnungseinheiten",
"symbol": "XBA"
},
"XBB": {
"displayname": "Europäische Währungseinheit (XBB)",
"displayname-count-one": "Europäische Währungseinheiten (XBB)",
"displayname-count-other": "Europäische Währungseinheiten (XBB)",
"symbol": "XBB"
},
"XBC": {
"displayname": "Europäische Rechnungseinheit (XBC)",
"displayname-count-one": "Europäische Rechnungseinheiten (XBC)",
"displayname-count-other": "Europäische Rechnungseinheiten (XBC)",
"symbol": "XBC"
},
"XBD": {
"displayname": "Europäische Rechnungseinheit (XBD)",
"displayname-count-one": "Europäische Rechnungseinheiten (XBD)",
"displayname-count-other": "Europäische Rechnungseinheiten (XBD)",
"symbol": "XBD"
},
"XCD": {

```

```

"displayName": "Ostkaribischer Dollar",
"displayName-count-one": "Ostkaribischer Dollar",
"displayName-count-other": "Ostkaribische Dollar",
"symbol": "EC$",
"symbol-alt-narrow": "$"
},
"XDR": {
"displayName": "Sonderziehungsrechte",
"displayName-count-one": "Sonderziehungsrechte",
"displayName-count-other": "Sonderziehungsrechte",
"symbol": "XDR"
},
"XEU": {
"displayName": "Europäische Währungseinheit (XEU)",
"displayName-count-one": "Europäische Währungseinheiten (XEU)",
"displayName-count-other": "Europäische Währungseinheiten (XEU)",
"symbol": "XEU"
},
"XFO": {
"displayName": "Französischer Gold-Franc",
"displayName-count-one": "Französische Gold-Franc",
"displayName-count-other": "Französische Gold-Franc",
"symbol": "XFO"
},
"XFU": {
"displayName": "Französischer UIC-Franc",
"displayName-count-one": "Französische UIC-Franc",
"displayName-count-other": "Französische UIC-Franc",
"symbol": "XFU"
},
"XOF": {
"displayName": "CFA-Franc (BCEAO)",
"displayName-count-one": "CFA-Franc (BCEAO)",
"displayName-count-other": "CFA-Francs (BCEAO)",
"symbol": "CFA"
},
"XPD": {
"displayName": "Unze Palladium",
"displayName-count-one": "Unze Palladium",
"displayName-count-other": "Unzen Palladium",
"symbol": "XPD"
},
"XPF": {
"displayName": "CFP-Franc",
"displayName-count-one": "CFP-Franc",
"displayName-count-other": "CFP-Franc",
"symbol": "CFPF"
},
"XPT": {
"displayName": "Unze Platin",
"displayName-count-one": "Unze Platin",
"displayName-count-other": "Unzen Platin",
"symbol": "XPT"
},
"XRE": {
"displayName": "RINET Funds",
"displayName-count-one": "RINET Funds",

```

```

"displayName-count-other": "RINET Funds",
"symbol": "XRE"
},
"XSU": {
"displayName": "SUCRE",
"displayName-count-one": "SUCRE",
"displayName-count-other": "SUCRE",
"symbol": "XSU"
},
"XTS": {
"displayName": "Testwährung",
"displayName-count-one": "Testwährung",
"displayName-count-other": "Testwährung",
"symbol": "XTS"
},
"XUA": {
"displayName": "Rechnungseinheit der AfEB",
"displayName-count-one": "Rechnungseinheit der AfEB",
"displayName-count-other": "Rechnungseinheiten der AfEB",
"symbol": "XUA"
},
"XXX": {
"displayName": "Unbekannte Währung",
"displayName-count-one": "(unbekannte Währung)",
"displayName-count-other": "(unbekannte Währung)",
"symbol": "XXX"
},
"YDD": {
"displayName": "Jemen-Dinar",
"displayName-count-one": "Jemen-Dinar",
"displayName-count-other": "Jemen-Dinar",
"symbol": "YDD"
},
"YER": {
"displayName": "Jemen-Rial",
"displayName-count-one": "Jemen-Rial",
"displayName-count-other": "Jemen-Rial",
"symbol": "YER"
},
"YUD": {
"displayName": "Jugoslawischer Dinar (1966-1990)",
"displayName-count-one": "Jugoslawischer Dinar (1966-1990)",
"displayName-count-other": "Jugoslawische Dinar (1966-1990)",
"symbol": "YUD"
},
"YUM": {
"displayName": "Jugoslawischer Neuer Dinar (1994-2002)",
"displayName-count-one": "Jugoslawischer Neuer Dinar (1994-2002)",
"displayName-count-other": "Jugoslawische Neue Dinar (1994-2002)",
"symbol": "YUM"
},
"YUN": {
"displayName": "Jugoslawischer Dinar (konvertibel)",
"displayName-count-one": "Jugoslawische Dinar (konvertibel)",
"displayName-count-other": "Jugoslawische Dinar (konvertibel)",
"symbol": "YUN"
},

```

```

"YUR": {
  "displayName": "Jugoslawischer reformierter Dinar (1992-1993)",
  "displayName-count-one": "Jugoslawischer reformierter Dinar (1992-1993)",
  "displayName-count-other": "Jugoslawische reformierte Dinar (1992-1993)",
  "symbol": "YUR"
},
"ZAL": {
  "displayName": "Südafrikanischer Rand (Finanz)",
  "displayName-count-one": "Südafrikanischer Rand (Finanz)",
  "displayName-count-other": "Südafrikanischer Rand (Finanz)",
  "symbol": "ZAL"
},
"ZAR": {
  "displayName": "Südafrikanischer Rand",
  "displayName-count-one": "Südafrikanischer Rand",
  "displayName-count-other": "Südafrikanische Rand",
  "symbol": "ZAR",
  "symbol-alt-narrow": "R"
},
"ZMK": {
  "displayName": "Kwacha (1968-2012)",
  "displayName-count-one": "Kwacha (1968-2012)",
  "displayName-count-other": "Kwacha (1968-2012)",
  "symbol": "ZMK"
},
"ZMW": {
  "displayName": "Kwacha",
  "displayName-count-one": "Kwacha",
  "displayName-count-other": "Kwacha",
  "symbol": "ZMW",
  "symbol-alt-narrow": "K"
},
"ZRN": {
  "displayName": "Zaire-Neuer Zaïre (1993-1998)",
  "displayName-count-one": "Zaire-Neuer Zaïre (1993-1998)",
  "displayName-count-other": "Zaire-Neue Zaïre (1993-1998)",
  "symbol": "ZRN"
},
"ZRZ": {
  "displayName": "Zaire-Zaïre (1971-1993)",
  "displayName-count-one": "Zaire-Zaïre (1971-1993)",
  "displayName-count-other": "Zaire-Zaïre (1971-1993)",
  "symbol": "ZRZ"
},
"ZWD": {
  "displayName": "Simbabwe-Dollar (1980-2008)",
  "displayName-count-one": "Simbabwe-Dollar (1980-2008)",
  "displayName-count-other": "Simbabwe-Dollar (1980-2008)",
  "symbol": "ZWD"
},
"ZWL": {
  "displayName": "Simbabwe-Dollar (2009)",
  "displayName-count-one": "Simbabwe-Dollar (2009)",
  "displayName-count-other": "Simbabwe-Dollar (2009)",
  "symbol": "ZWL"
},
"ZWR": {

```

```

"displayName": "Simbabwe-Dollar (2008)",
"displayName-count-one": "Simbabwe-Dollar (2008)",
"displayName-count-other": "Simbabwe-Dollar (2008)",
"symbol": "ZWR"
}
}
}
}
}
}
}

```

Drucken
 Alle erweitern
 Alles einklappen

Task ID	Task Name	Duration	Start Date	Priority
1	▼ Parent Task 1	€10.00	10/23/2017	Critical
2	▼ Child task 1	€50.00	10/24/2017	Low
3	Child Task 2	€5.00	10/25/2017	Critical
4	Child task 3	€6.00	10/26/2017	High

1 von 2 Seiten (2 Beiträge)

* In the above sample, **Duration** column is formatted by **NumberFormatOptions**.

* By default, **locale** value is **en-US**. In order to change the **en-US** culture to a different culture, set the **SetCulture** method accordingly.

Right to left (RTL)

RTL provides an option to switch the text direction and layout of the Tree 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.). In the below sample **EnableRtl** method is used to enable RTL in the Tree Grid.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" AllowPaging="true" Locale="ar-AE"
IdMapping="TaskId" AllowSelection="true" ParentIdMapping="ParentId"
TreeColumnIndex="1"
Toolbar="@ (new List<string>() { "Print", "ExpandAll", "CollapseAll" })">
<TreeGridPageSettings PageSize="1"></TreeGridPageSettings>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>

```



```

<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
Format="C2"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText=" Start Date" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
[Inject]
IJSRuntime JsRuntime { get; set; }
public List<TreeData> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
protected override void OnAfterRender(bool firstRender)
{
if (firstRender)
{
this.JsRuntime.Sf().EnableRtl(true);
}
}
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<TreeData> GetSelfDataSource()
{
List<TreeData> BusinessObjectCollection = new List<TreeData>();
BusinessObjectCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent
Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null
});
BusinessObjectCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child
task 1", Progress = 80, Priority = "Low", ParentId = 1 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child
Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child
task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent
Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
BusinessObjectCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child
task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5 });
}
}
}

```

```

BusinessObjectCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child
Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5});
BusinessObjectCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child
task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5});
BusinessObjectCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child
task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5});
return BusinessObjectCollection;
}
}
}

```

JSON

```

{
  "ar-AE": {
    "treegrid": {
      "EmptyRecord": "لا سجلات لعرضها",
      "Print": "طباعة",
      "ExpandAll": "توسيع الكل",
      "CollapseAll": "انهيار جميع",
      "FilterButton": "منقي",
      "ClearButton": "واضح",
      "StartsWith": "ابدا ب",
      "EndsWith": "ينتهي مع",
      "Contains": "يحتوي على",
      "Equal": "مساو",
      "NotEqual": "غير متساوي",
      "LessThan": "أقل من",
      "LessThanOrEqual": "اصغر من أو يساوي",
      "GreaterThan": "أكثر من",
      "GreaterThanOrEqual": "أكبر من أو يساوي",
      "ChooseDate": "اختر تاريخا",
      "EnterValue": "أدخل القيمة",
      "FilterMenu": "منقي"
    },
    "pager": {
      "currentPageInfo": "من {0} 1 {صفحة}",
      "totalItemsInfo": "({0} العناصر)",
      "firstPageTooltip": "انتقل إلى الصفحة الأولى",
      "lastPageTooltip": "انتقل إلى الصفحة الأخيرة",
      "nextPageTooltip": "انتقل إلى الصفحة التالية",
      "previousPageTooltip": "انتقل إلى الصفحة السابقة",
      "nextPagerTooltip": "الذهاب إلى بيجر المقبل",
      "previousPagerTooltip": "الذهاب إلى بيجر السابقة"
    },
    "dropdowns": {
      "noRecordsTemplate": "لا توجد سجلات"
    },
    "datepicker": {
      "placeholder": "اختر تاريخا",
      "today": "اليوم"
    }
  }
}

```

<div> <div>طباعة</div> <div>توسيع الكل</div> <div>انهيار جميع</div> </div>				
Priority	Start Date	Duration	Task Name	Task ID
Critical	10/23/2017	\$10.00	Parent Task 1 ▾	1
Low	10/24/2017	\$50.00	Child task 1 ▾	2
Critical	10/25/2017	\$5.00	Child Task 2	3
High	10/26/2017	\$6.00	Child task 3	4
1 من 2 صفحة (2 العناصر)			<div> <div>⏪</div> <div><</div> <div>2</div> <div>1</div> <div>></div> <div>⏩</div> </div>	

Selection in Blazor TreeGrid Component

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 Tree Grid, set the [AllowSelection](#) to false.

To know more about selection feature in Blazor tree grid component, you can check on this video.

{% youtube

"youtube:https://www.youtube.com/watch?v=0RkHA05pS5s"%}

The tree grid supports two types of selection that can be set by using the [Type](#) property in [TreeGridSelectionSettings](#). They are:

- **Single:** The Single value is set by default, and it only allows selection of a single row or a cell.
- **Multiple:** It allows to select multiple rows or cells.

To perform the multi-selection, press and hold CTRL key and click the desired rows or cells. To select range of rows or cells, press and hold the SHIFT key and click the rows or cells.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
AllowSelection="true" ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridSelectionSettings
Type="Syncfusion.Blazor.Grids.SelectionType.Multiple"></TreeGridSelectionSet
tings>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
```

```

<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
}
public static List<BusinessObject> GetSelfDataSource()
{
List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", Duration = 6, Progress = 80, Priority = "Low", ParentId = 1 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5});
return BusinessObjectCollection;
}
}
}

```

Selection mode

The tree grid supports three types of selection mode that can be set by using the [Mode](#) property of the [TreeGridSelectionSettings](#). They are:

- **Row:** The Row value is set by default, and allows to select only rows.
- **Cell:** It allows to select only cells.
- **Both:** It allows to select rows and cells at the same time.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
@using Syncfusion.Blazor.Grids;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
AllowSelection="true" ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridSelectionSettings Type="SelectionType.Multiple"
Mode="Syncfusion.Blazor.Grids.SelectionMode.Cell">
</TreeGridSelectionSettings>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}
```

C#

```
namespace TreeGridComponent.Data {
public class TreeData
{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
}
public static List<BusinessObject> GetSelfDataSource()
{
```

```

List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", Duration = 6, Progress = 80, Priority = "Low", ParentId = 1 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5});
return BusinessObjectCollection;
}
}
}

```

Task ID	Task Name	Duration	Progress	Priority
1	▼ Parent Task 1	10	70	Critical
2	▼ Child task 1	4	80	Low
3	▼ Child Task 2	5	65	Critical
4	Child task 3	6	77	High
5	▼ Parent Task 2	10	70	Critical
6	Child task 1	4	80	Critical
7	Child Task 2	5	65	Low
8	Child task 3	6	77	High
9	Child task 4	6	77	Low

Cell selection

Cell selection can be done through simple mouse down or arrow keys (up, down, left, and right).

The tree grid supports two types of cell selection mode that can be set by using the [CellSelectionMode](#) property of [TreeGridSelectionSettings](#). They are:

- **Flow:** The Flow value is set by default. The range of cells are selected between the start index and end index that includes in between cells of rows.
- **Box:** Range of cells are selected from the start and end column indexes that includes in between cells of rows within the range.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
@using Syncfusion.Blazor.Grids;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
AllowSelection="true" ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridSelectionSettings Type="SelectionType.Multiple"
Mode="Syncfusion.Blazor.Grids.SelectionMode.Cell"
CellSelectionMode="CellSelectionMode.Box"></TreeGridSelectionSettings>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}
```

C#

```
namespace TreeGridComponent.Data {
public class TreeData
{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
}
public static List<BusinessObject> GetSelfDataSource()
{
}
```

```

List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", Progress = 80, Priority = "Low", ParentId = 1 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5});
return BusinessObjectCollection;
}
}
}

```

Task ID	Task Name	Duration	Progress	Priority
1	▼ Parent Task 1	10	70	Critical
2	▼ Child task 1	4	80	Low
3	▼ Child Task 2	5	65	Critical
4	Child task 3	6	77	High
5	▼ Parent Task 2	10	70	Critical
6	Child task 1	4	80	Critical
7	Child Task 2	5	65	Low
8	Child task 3	6	77	High
9	Child task 4	6	77	Low

Cell selection requires the [Mode](#) to be **Cell** or **Both**, and **Type** should be **Multiple**.

Checkbox selection

Checkbox selection provides an option to select multiple tree grid records with the help of checkbox in each row. To render the checkbox in each tree grid row, use the checkbox column with type as **checkbox** using the column [Type](#) property.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
AllowSelection="true" ParentIdMapping="ParentId" TreeColumnIndex="2">
<TreeGridColumn>
<TreeGridColumn Type="Syncfusion.Blazor.Grids.ColumnType.CheckBox"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
public List<TreeData> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}
```


C#

```
namespace TreeGridComponent.Data {
public class TreeData
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<TreeData> GetSelfDataSource()
{
List<TreeData> BusinessObjectCollection = new List<TreeData>();
BusinessObjectCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent
Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null
});
BusinessObjectCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child
task 1", Duration = 6, Progress = 80, Priority = "Low", ParentId = 1 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child
Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
}
```

```

BusinessObjectCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child
task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent
Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null});
BusinessObjectCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child
task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5});
BusinessObjectCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child
Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5});
BusinessObjectCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child
task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5});
BusinessObjectCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child
task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5});
return BusinessObjectCollection;
}
}
}

```

	Task ID	Task Name	Duration	Progress	Priority
<input type="checkbox"/>	1	▼ Parent Task 1	10	70	Critical
<input checked="" type="checkbox"/>	2	▼ Child task 1	50	80	Low
<input checked="" type="checkbox"/>	3	▼ Child Task 2	5	65	Critical
<input type="checkbox"/>	4	Child task 3	6	77	High
<input checked="" type="checkbox"/>	5	▼ Parent Task 2	10	70	Critical
<input checked="" type="checkbox"/>	6	Child task 1	4	80	Critical
<input type="checkbox"/>	7	Child Task 2	5	65	Low
<input checked="" type="checkbox"/>	8	Child task 3	6	77	High
<input type="checkbox"/>	9	Child task 4	6	77	Low

* By default, selection is allowed by clicking a tree grid row or checkbox in that row. To allow selection only through checkbox, set the

[CheckboxOnly](#) property of [TreeGridSelectionSettings](#) to true.

* Selection can be persisted in all the operations using the [PersistSelection](#) property of [TreeGridSelectionSettings](#).

For persisting selection on the tree grid, any one of the columns should be defined as a primary key using the [IsPrimaryKey](#) property of [TreeGridColumn](#).

Checkbox selection Mode

In checkbox selection, selection can also be done by clicking on rows. This selection provides two types of Checkbox Selection mode which can be set by using the following API, [CheckboxMode](#) in [TreeGridSelectionSettings](#). The modes are;

- **Default:** This is the default value of the checkboxMode. In this mode, user can select multiple rows by clicking rows one by one.
- **ResetOnRowClick:** In ResetOnRowClick mode, when user clicks on a row it will reset previously selected row. Also, multiple-selection can be performed in this mode by pressing and holding CTRL key, and clicking the desired rows. To select range of rows, press and hold the SHIFT key and click the rows.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
AllowSelection="true" ParentIdMapping="ParentId" TreeColumnIndex="2">
<TreeGridSelectionSettings
CheckboxMode="CheckboxSelectionMode.ResetOnRowClick"></TreeGridSelectionSettings>
<TreeGridColumns>
<TreeGridColumn Type="Syncfusion.Blazor.Grids.ColumnType.CheckBox"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public List<TreeData> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}
```


C#

```
namespace TreeGridComponent.Data {
public class TreeData
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
}
```

```

public int? ParentId { get; set; }
public static List<TreeData> GetSelfDataSource()
{
    List<TreeData> BusinessObjectCollection = new List<TreeData>();
    BusinessObjectCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
    BusinessObjectCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child task 1", Duration = 6, Progress = 80, Priority = "Low", ParentId = 1 });
    BusinessObjectCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
    BusinessObjectCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
    BusinessObjectCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
    BusinessObjectCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5 });
    BusinessObjectCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
    BusinessObjectCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
    BusinessObjectCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
    return BusinessObjectCollection;
}
}
}

```

	Task ID	Task Name	Duration	Progress	Priority
<input type="checkbox"/>	1	▼ Parent Task 1	10	70	Critical
<input type="checkbox"/>	2	▼ Child task 1	50	80	Low
<input type="checkbox"/>	3	▼ Child Task 2	5	65	Critical
<input type="checkbox"/>	4	Child task 3	6	77	High
<input type="checkbox"/>	5	▼ Parent Task 2	10	70	Critical
<input checked="" type="checkbox"/>	6	Child task 1	4	80	Critical
<input type="checkbox"/>	7	Child Task 2	5	65	Low
<input type="checkbox"/>	8	Child task 3	6	77	High
<input type="checkbox"/>	9	Child task 4	6	77	Low

Toggle Selection

The Toggle selection allows to perform selection and unselection of the particular row or cell. To enable toggle selection, set [EnableToggle](#) property of the TreeGridSelectionSettings as true. If the selected row or cell is clicked, then it will be unselected and vice versa.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeGridData" IdMapping="TaskId"
AllowSelection="true" ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridSelectionSettings EnableToggle="true"
Type="SelectionMode.Multiple"></TreeGridSelectionSettings>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public List<TreeData> TreeGridData { get; set; }
SfTreeGrid<TreeData> TreeGrid;
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<TreeData> GetSelfDataSource()
{
List<TreeData> BusinessObjectCollection = new List<TreeData>();
BusinessObjectCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent
Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null
});
BusinessObjectCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child
task 1", Duration = 6, Progress = 80, Priority = "Low", ParentId = 1 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child
Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child
task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent
Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null});
}
}

```

```

BusinessObjectCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child
task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5});
BusinessObjectCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child
Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5});
BusinessObjectCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child
task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5});
BusinessObjectCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child
task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5});
return BusinessObjectCollection;
}
}
}

```

Task ID	Task Name	Duration	Progress	Priority
1	Parent Task 1	10	70	Critical
2	Child task 1	6	80	Low
3	Child Task 2	5	65	Critical
4	Child task 3	6	77	High
5	Parent Task 2	10	70	Critical
6	Child task 1	4	80	Critical
7	Child Task 2	5	65	Low
8	Child task 3	6	77	High
9	Child task 4	6	77	Low

If multi selection is enabled, then first click on any selected row (without pressing Ctrl key), it will clear the multi selection and in second click on the same row, it will be unselected.

Select row at initial rendering

To select a row at initial rendering, set the [SelectedRowIndex](#) value.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
AllowSelection="true" ParentIdMapping="ParentId" TreeColumnIndex="1"
SelectedRowIndex="1">
<TreeGridColumn>
<TreeGridColumnField Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumnField Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumnField Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumnField Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>

```

```

<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
}
public static List<BusinessObject> GetSelfDataSource()
{
List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", Duration = 6, Progress = 80, Priority = "Low", ParentId = 1 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5});
return BusinessObjectCollection;
}
}
}

```

Task ID	Task Name	Duration	Progress	Priority
1	▼ Parent Task 1	10	70	Critical
2	▼ Child task 1	4	80	Low
3	▼ Child Task 2	5	65	Critical
4	Child task 3	6	77	High
5	▼ Parent Task 2	10	70	Critical
6	Child task 1	4	80	Critical
7	Child Task 2	5	65	Low
8	Child task 3	6	77	High
9	Child task 4	6	77	Low

Get selected row indexes

The selected row indexes is got by using the GetSelectedRowIndexes method.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeGridData" IdMapping="TaskId"
AllowSelection="true" ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridEvents RowSelected="RowSelectHandler"
TValue="TreeData"></TreeGridEvents>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
public List<TreeData> TreeGridData { get; set; }
SfTreeGrid<TreeData> TreeGrid;
public List<double> SelectedRowIndexes { get; set; }
public List<TreeData> SelectedRecords { get; set; }
protected override void OnInitialized()
```



```

{
    this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
private async void RowSelectHandler(RowSelectEventArgs<TreeData> Args)
{
    this.SelectedRowIndex = await this.TreeGrid.GetSelectedRowIndex();    ///
    get the selected row indexes
    string index = JsonConvert.SerializeObject(this.SelectedRowIndex);
    JsRuntime.InvokeAsync<string>("window.alert", index);
    this.SelectedRecords = await this.TreeGrid.GetSelectedRecords();    /// get
    the selected records
    string records = JsonConvert.SerializeObject(this.SelectedRecords);
    JsRuntime.InvokeAsync<string>("window.alert", records);
}
}

```

C#



```

namespace TreeGridComponent.Data {
    public class TreeData
    {
        public int TaskId { get; set; }
        public string TaskName { get; set; }
        public int? Duration { get; set; }
        public int? Progress { get; set; }
        public string Priority { get; set; }
        public int? ParentId { get; set; }
        public static List<TreeData> GetSelfDataSource()
        {
            List<TreeData> BusinessObjectCollection = new List<TreeData>();
            BusinessObjectCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent
            Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null
            });
            BusinessObjectCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child
            task 1", Duration = 6, Progress = 80, Priority = "Low", ParentId = 1 });
            BusinessObjectCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child
            Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
            BusinessObjectCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child
            task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
            BusinessObjectCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent
            Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
            BusinessObjectCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child
            task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5 });
            BusinessObjectCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child
            Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
            BusinessObjectCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child
            task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
            BusinessObjectCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child
            task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
            return BusinessObjectCollection;
        }
    }
}

```

Touch interaction

When a tree grid row is tapped on the touchscreen device, the tapped row is selected. It also shows a

popup  for multi-row selection. To select multiple rows or cells, tap the popup  and then tap the desired rows or cells.

Multi-selection requires the selection [Type](#) to be **Multiple**.

The following screenshot represents a tree grid touch selection in the device.

```
<!-- markdownlint-disable MD033 -->
```

```

```

```
<!-- markdownlint-enable MD033 -->
```

Toolbar in Blazor TreeGrid Component

The Tree Grid provides Toolbar support to handle tree grid actions.

Built-in toolbar items

Built-in toolbar items execute standard actions of the tree grid, and it can be added 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
-----	-----
ExpandAll	Expands all the rows.
CollapseAll	Collapses all the rows.
Add	Adds a new record.
Edit	Edits the selected record.
Update	Updates the edited record.
Delete	Deletes the selected record.
Cancel	Cancels the edit state.
Search	Searches the records by the given key.
Print	Prints the tree grid.
ExcelExport	Exports the tree grid to Excel.
PdfExport	Exports the tree grid to PDF.
WordExport	Exports the tree grid to Word.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
```

```

<SfTreeGrid DataSource="@TreeGridData" Toolbar="@ (new List<string>() {
    "Print", "Search" })" IdMapping="TaskId" AllowSelection="true"
    ParentIdMapping="ParentId" TreeColumnIndex="1">
    <TreeGridColumn>
    <TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
        TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
    <TreeGridColumn Field="TaskName" HeaderText="Task Name"
        Width="160"></TreeGridColumn>
    <TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
        TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
    <TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
        TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
    <TreeGridColumn Field="Priority" HeaderText="Priority"
        Width="80"></TreeGridColumn>
    </TreeGridColumn>
</SfTreeGrid>
@code{
    public List<TreeData.BusinessObject> TreeGridData { get; set; }
    protected override void OnInitialized()
    {
        this.TreeGridData = TreeData.GetSelfDataSource().ToList();
    }
}

```

C#

```

namespace TreeGridComponent.Data {
    public class TreeData
    {
        public class BusinessObject
        {
            public int TaskId { get; set; }
            public string TaskName { get; set; }
            public int? Duration { get; set; }
            public int? Progress { get; set; }
            public string Priority { get; set; }
            public int? ParentId { get; set; }
        }
        public static List<BusinessObject> GetSelfDataSource()
        {
            List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
                "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
                null });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
                "Child task 1", Progress = 80, Priority = "Low", ParentId = 1 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
                "Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
            });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
                "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
                "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
                null });
        }
    }
}

```

```

BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5});
return BusinessObjectCollection;
}
}
}

```

Print		Search		
Task ID	Task Name	Duration	Progress	Priority
1	▼ Parent Task 1	10	70	Critical
2	▼ Child task 1	4	80	Low
3	▼ Child Task 2	5	65	Critical
4	Child task 3	6	77	High
5	▼ Parent Task 2	10	70	Critical
6	Child task 1	4	80	Critical
7	Child Task 2	5	65	Low
8	Child task 3	6	77	High
9	Child task 4	6	77	Low

The [Toolbar](#) has options to define both built-in and custom toolbar items.

Enable/Disable Toolbar Items

The tool bar items can be enabled or disabled by using the [EnableToolbarItems](#) method.

ASPX-CS

```

@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.TreeGrid;
@using Syncfusion.Blazor.Grids;
<div>
<div style="float:left;">
<SfButton id="Enable" Content="Enable" @onclick="Enable"></SfButton>

```

```

</div>
<div style="padding-left: 90px">
<SfButton id="Disable" Content="Disable" @onclick="Disable"></SfButton>
</div>
</div>
@{
var Tool = (new string[] { "ExpandAll", "CollapseAll" });
}
<SfTreeGrid ID="TreeGrid" @ref="TreeGrid" DataSource="TreeGridData"
IdMapping="TaskId" ParentIdMapping="ParentId" TreeColumnIndex="1"
Toolbar="@Tool" Height="350">
<TreeGridEvents TValue="TreeData"
OnToolbarClick="ToolBarClick"></TreeGridEvents>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="145"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority" Width="100"
TextAlign="TextAlign.Right"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid<TreeData> TreeGrid;
public void Enable()
{
this.TreeGrid.EnableToolbarItems(new List<string>() {
"TreeGrid_gridcontrol_ExpandAll", "TreeGrid_gridcontrol_CollapseAll" },
true);
}
public void Disable()
{
this.TreeGrid.EnableToolbarItems(new List<string>() {
"TreeGrid_gridcontrol_ExpandAll", "TreeGrid_gridcontrol_CollapseAll" },
false);
}
public void ToolBarClick(Syncfusion.Blazor.Navigations.ClickEventArgs Args)
{
if (Args.Item.Text == "ExpandAll")
{
this.TreeGrid.ExpandAll();
}
if (Args.Item.Text == "CollapseAll")
{
this.TreeGrid.CollapseAll();
}
}
public List<TreeData> TreeGridData { get; set; }
public class TreeData
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
}
}

```

```
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<TreeData> GetSelfDataSource()
{
    List<TreeData> TreeDataCollection = new List<TreeData>();
    TreeDataCollection.Add(new TreeData(){ TaskId = 1, TaskName = "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
    TreeDataCollection.Add(new TreeData(){ TaskId = 2, TaskName = "Child task 1", Progress = 80, Priority = "Low", Duration = 50, ParentId = 1 });
    TreeDataCollection.Add(new TreeData(){ TaskId = 3, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
    TreeDataCollection.Add(new TreeData(){ TaskId = 4, TaskName = "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
    TreeDataCollection.Add(new TreeData(){ TaskId = 5, TaskName = "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
    TreeDataCollection.Add(new TreeData(){ TaskId = 6, TaskName = "Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5 });
    TreeDataCollection.Add(new TreeData(){ TaskId = 7, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
    TreeDataCollection.Add(new TreeData(){ TaskId = 8, TaskName = "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
    TreeDataCollection.Add(new TreeData(){ TaskId = 9, TaskName = "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
    return TreeDataCollection;
}
}
protected override void OnInitialized()
{
    this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}
```

The following screenshots represent a Tree Grid with Enable/disable toolbar items,

Enable

Disable

ExpandAll

CollapseAll

Task ID	Task Name	Duration	Progress	Priority
1	▼ Parent Task 1	10	70	Critical
2	▼ Child task 1	50	80	Low
3	▼ Child Task 2	5	65	Critical
4	Child task 3	6	77	High
5	▼ Parent Task 2	10	70	Critical
6	Child task 1	4	80	Critical
7	Child Task 2	5	65	Low
8	Child task 3	6	77	High
9	Child task 4	6	77	Low

<!--

Custom toolbar items

Custom toolbar items can be added by defining the [Toolbar](#) as a collection of

ItemModels.

Actions for this customized toolbar items are defined in the [ToolbarClick](#) event.

By default, Custom toolbar items are in position **Left**. You can change the position by using the **Align** property. In the below sample, we have applied position **Right** for the **Quick Filter** toolbar item.

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

Tree Grid have an option to use both built-in and custom toolbar items at same time.

In the below example, **ExpandAll**, **CollapseAll** are built-in toolbar items and **Click** is custom toolbar item.

Enable/disable toolbar items

You can enable/disable toolbar items by using the **enableItems** method.

-->

Excel Export in Blazor TreeGrid Component

The excel export allows exporting Tree Grid data to Excel document. Use the

ExcelExport method for exporting. To enable Excel export in the Tree Grid, set the [AllowExcelExport](#) property as true.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeGridData"
AllowExcelExport="true" IdMapping="TaskId" ParentIdMapping="ParentId"
TreeColumnIndex="1"
Toolbar="@ (new List<string>() { "ExcelExport" }) ">
<TreeGridEvents OnToolbarClick="ToolbarClickHandler"
TValue="TreeData"></TreeGridEvents>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
Format="C2"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText=" Start Date" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid<TreeData> TreeGrid;
public List<TreeData> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
private void
ToolbarClickHandler(Syncfusion.Blazor.Navigations.ClickEventArgs Args)
{
if(Args.Item.Text == "Excel Export")
{
this.TreeGrid.ExcelExport();
}
}
}
```

C#

```
namespace TreeGridComponent.Data {
public class TreeData
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
}
```



```

public int? ParentId { get; set; }
public static List<TreeData> GetSelfDataSource()
{
    List<TreeData> BusinessObjectCollection = new List<TreeData>();
    BusinessObjectCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
    BusinessObjectCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child task 1", Progress = 80, Priority = "Low", ParentId = 1 });
    BusinessObjectCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
    BusinessObjectCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 2 });
    BusinessObjectCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
    BusinessObjectCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5 });
    BusinessObjectCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
    BusinessObjectCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
    BusinessObjectCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
    return BusinessObjectCollection;
}
}
}

```

To customize excel export

The excel export provides an option to customize mapping of the Tree Grid to excel document.

[Export current page](#)

The excel export provides an option to export the current page into excel. To export current page, define **exportType** to **CurrentPage**.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeGridData"
AllowExcelExport="true" IdMapping="TaskId" ParentIdMapping="ParentId"
TreeColumnIndex="1"
Toolbar="@ (new List<string>() { "ExcelExport" }) ">
<TreeGridEvents OnToolbarClick="ToolbarClickHandler"
TValue="TreeData"></TreeGridEvents>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
Format="C2"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText=" Start Date" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date"

```

```

TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid<TreeData> TreeGrid;
public List<TreeData> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
private void
ToolbarClickHandler(Syncfusion.Blazor.Navigations.ClickEventArgs Args)
{
if(Args.Item.Text == "Excel Export")
{
Syncfusion.Blazor.Grids.ExcelExportProperties ExportProperties = new
Syncfusion.Blazor.Grids.ExcelExportProperties();
ExportProperties.ExportType =
Syncfusion.Blazor.Grids.ExportType.CurrentPage;
this.TreeGrid.ExcelExport(ExportProperties);
}
}
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<TreeData> GetSelfDataSource()
{
List<TreeData> BusinessObjectCollection = new List<TreeData>();
BusinessObjectCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent
Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null
});
BusinessObjectCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child
task 1", Progress = 80, Priority = "Low", ParentId = 1 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child
Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child
task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 2 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent
Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null});
BusinessObjectCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child
task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5});
BusinessObjectCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child
Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5});
}
}
}

```

```

BusinessObjectCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child
task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5});
BusinessObjectCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child
task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5});
return BusinessObjectCollection;
}
}
}

```

Export hidden columns

The excel export provides an option to export hidden columns of Tree Grid by defining **includeHiddenColumn** as **true**.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeGridData"
AllowExcelExport="true" IdMapping="TaskId" ParentIdMapping="ParentId"
TreeColumnIndex="1"
Toolbar="@ (new List<string>() { "ExcelExport" })">
<TreeGridEvents OnToolbarClick="ToolbarClickHandler"
TValue="TreeData"></TreeGridEvents>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
Format="C2"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="StartDate" Visible="false" HeaderText=" Start Date"
Format="yMd" Type="Syncfusion.Blazor.Grids.ColumnType.Date"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid<TreeData> TreeGrid;
public List<TreeData> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
private void
ToolbarClickHandler(Syncfusion.Blazor.Navigations.ClickEventArgs Args)
{
if(Args.Item.Text == "Excel Export")
{
Syncfusion.Blazor.Grids.ExcelExportProperties ExportProperties = new
Syncfusion.Blazor.Grids.ExcelExportProperties();
ExportProperties.IncludeHiddenColumn = true;
this.TreeGrid.ExcelExport(ExportProperties);
}
}
}

```

```
}
}
```

C#

```
namespace TreeGridComponent.Data {
public class TreeData
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<TreeData> GetSelfDataSource()
{
List<TreeData> BusinessObjectCollection = new List<TreeData>();
BusinessObjectCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent
Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null
});
BusinessObjectCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child
task 1", Progress = 80, Priority = "Low", ParentId = 1 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child
Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child
task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 2 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent
Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
BusinessObjectCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child
task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child
Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child
task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child
task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
return BusinessObjectCollection;
}
}
}
```

Theme

The excel export provides an option to include theme for exported excel document.

To apply theme in exported Excel, define the **theme** in export properties.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeGridData"
AllowExcelExport="true" IdMapping="TaskId" ParentIdMapping="ParentId"
TreeColumnIndex="1"
Toolbar="@ (new List<string>() { "ExcelExport" }) ">
<TreeGridEvents OnToolbarClick="ToolbarClickHandler"
TValue="TreeData"></TreeGridEvents>
```

```

<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
Format="C2"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText=" Start Date" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
SfTreeGrid<TreeData> TreeGrid;
public List<TreeData> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
private void
ToolbarClickHandler(Syncfusion.Blazor.Navigations.ClickEventArgs Args)
{
if(Args.Item.Text == "Excel Export")
{
Syncfusion.Blazor.Grids.ExcelExportProperties ExportProperties = new
Syncfusion.Blazor.Grids.ExcelExportProperties();
Syncfusion.Blazor.Grids.ExcelTheme Theme = new
Syncfusion.Blazor.Grids.ExcelTheme();
Syncfusion.Blazor.Grids.ExcelStyle ThemeStyle = new
Syncfusion.Blazor.Grids.ExcelStyle()
{
FontName = "Segoe UI",
FontColor = "#666666"
};
Theme.Header = ThemeStyle;
Theme.Record = ThemeStyle;
Theme.Caption = ThemeStyle;
this.TreeGrid.ExcelExport(ExportProperties);
}
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
}
}

```

```

public int? ParentId { get; set; }
public static List<TreeData> GetSelfDataSource()
{
    List<TreeData> BusinessObjectCollection = new List<TreeData>();
    BusinessObjectCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
    BusinessObjectCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child task 1", Progress = 80, Priority = "Low", ParentId = 1 });
    BusinessObjectCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
    BusinessObjectCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 2 });
    BusinessObjectCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
    BusinessObjectCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5 });
    BusinessObjectCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
    BusinessObjectCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
    BusinessObjectCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
    return BusinessObjectCollection;
}
}
}

```

By default, material theme is applied to exported excel document.

File name for exported document

The file name can be assigned for the exported document by defining **fileName** property in excel export properties.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeGridData"
AllowExcelExport="true" IdMapping="TaskId" ParentIdMapping="ParentId"
TreeColumnIndex="1"
Toolbar="@ (new List<string>() { "ExcelExport" }) ">
<TreeGridEvents OnToolbarClick="ToolbarClickHandler"
TValue="TreeData"></TreeGridEvents>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
Format="C2"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText="Start Date" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="100"></TreeGridColumn>

```

```

<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid<TreeData> TreeGrid;
public List<TreeData> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
private void
ToolbarClickHandler(Syncfusion.Blazor.Navigations.ClickEventArgs Args)
{
if(Args.Item.Text == "Excel Export")
{
Syncfusion.Blazor.Grids.ExcelExportProperties ExportProperties = new
Syncfusion.Blazor.Grids.ExcelExportProperties();
ExportProperties.FileName = "New.xlsx";
this.TreeGrid.ExcelExport(ExportProperties);
}
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<TreeData> GetSelfDataSource()
{
List<TreeData> BusinessObjectCollection = new List<TreeData>();
BusinessObjectCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent
Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null
});
BusinessObjectCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child
task 1", Progress = 80, Priority = "Low", ParentId = 1 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child
Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child
task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 2 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent
Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null});
BusinessObjectCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child
task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5});
BusinessObjectCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child
Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child
task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
}
}

```

```

BusinessObjectCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child
task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
return BusinessObjectCollection;
}
}
}

```

To persist collapsed state

The collapsed state can be persisted in the exported document by defining **IsCollapsedStatePersist** property as true in the **TreeGridExcelExportProperties** parameter of the [ExcelExport](#) method.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeGridData"
AllowExcelExport="true" IdMapping="TaskId" ParentIdMapping="ParentId"
TreeColumnIndex="1"
Toolbar="@ (new List<string>() { "ExcelExport" }) ">
<TreeGridEvents OnToolbarClick="ToolbarClickHandler"
TValue="TreeData"></TreeGridEvents>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
Format="C2"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText=" Start Date" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid<TreeData> TreeGrid;
public List<TreeData> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
private void
ToolbarClickHandler(Syncfusion.Blazor.Navigations.ClickEventArgs Args)
{
if(Args.Item.Text == "Excel Export")
{
Syncfusion.Blazor.TreeGrid.TreeGridExcelExportProperties ExportProperties =
new Syncfusion.Blazor.TreeGrid.TreeGridExcelExportProperties();
ExportProperties.IsCollapsedStatePersist = true;
this.TreeGrid.ExcelExport(ExportProperties);
}
}
}

```


C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<TreeData> GetSelfDataSource()
{
List<TreeData> BusinessObjectCollection = new List<TreeData>();
BusinessObjectCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent
Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null
});
BusinessObjectCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child
task 1", Progress = 80, Priority = "Low", ParentId = 1 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child
Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child
task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 2 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent
Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
BusinessObjectCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child
task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child
Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child
task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child
task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
return BusinessObjectCollection;
}
}
}

```

Limitations

Microsoft Excel permits up to seven nested levels in outlines. So that in the Tree Grid we can able to provide only up to seven nested levels and if it exceeds more than seven levels then the document will be exported without outline option. Please refer the [Microsoft Limitation](#).

PDF Export in Blazor TreeGrid Component

PDF export allows exporting Tree Grid data to PDF document. You need to use the

PdfExport method for exporting. To enable PDF export in the Tree Grid, set the [AllowPdfExport](#) as true.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeGridData" AllowPdfExport="true"
IdMapping="TaskId" ParentIdMapping="ParentId" TreeColumnIndex="1"
Toolbar="@ (new List<string>() { "PdfExport" }) ">

```

```

<TreeGridEvents OnToolbarClick="ToolbarClickHandler"
TValue="TreeData"></TreeGridEvents>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
Format="C2"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText=" Start Date" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid<TreeData> TreeGrid;
public List<TreeData> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
private void
ToolbarClickHandler(Syncfusion.Blazor.Navigations.ClickEventArgs Args)
{
if(Args.Item.Text == "PDF Export")
{
this.TreeGrid.PdfExport();
}
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<TreeData> GetSelfDataSource()
{
List<TreeData> BusinessObjectCollection = new List<TreeData>();
BusinessObjectCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent
Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null
});
BusinessObjectCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child
task 1", Progress = 80, Priority = "Low", ParentId = 1 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child
Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
}
}

```

```

BusinessObjectCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 2 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
BusinessObjectCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
return BusinessObjectCollection;
}
}
}

```

To customize PDF export

PDF export provides an option to customize mapping of Tree Grid to exported PDF document.

File name for exported document

The file name can be assigned for the exported document by defining **fileName** property in **PdfExportProperties**.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeGridData" AllowPdfExport="true"
IdMapping="TaskId" ParentIdMapping="ParentId" TreeColumnIndex="1"
Toolbar="@ (new List<string>() { "PdfExport" })">
<TreeGridEvents OnToolbarClick="ToolbarClickHandler"
TValue="TreeData"></TreeGridEvents>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
Format="C2"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText=" Start Date" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
SfTreeGrid<TreeData> TreeGrid;
public List<TreeData> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}

```

```
private void
ToolbarClickHandler(Syncfusion.Blazor.Navigations.ClickEventArgs Args)
{
    if (Args.Item.Text == "PDF Export")
    {
        Syncfusion.Blazor.Grids.PdfExportProperties ExportProperties = new
        Syncfusion.Blazor.Grids.PdfExportProperties();
        ExportProperties.FileName = "test.pdf";
        this.TreeGrid.PdfExport(ExportProperties);
    }
}
}
```

C#

```
namespace TreeGridComponent.Data {
    public class TreeData
    {
        public int TaskId { get; set; }
        public string TaskName { get; set; }
        public int? Duration { get; set; }
        public int? Progress { get; set; }
        public string Priority { get; set; }
        public int? ParentId { get; set; }
        public static List<TreeData> GetSelfDataSource()
        {
            List<TreeData> BusinessObjectCollection = new List<TreeData>();
            BusinessObjectCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent
            Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null
            });
            BusinessObjectCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child
            task 1", Progress = 80, Priority = "Low", ParentId = 1 });
            BusinessObjectCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child
            Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
            BusinessObjectCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child
            task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 2 });
            BusinessObjectCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent
            Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
            BusinessObjectCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child
            task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5 });
            BusinessObjectCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child
            Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
            BusinessObjectCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child
            task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
            BusinessObjectCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child
            task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
            return BusinessObjectCollection;
        }
    }
}
```

How to change page orientation

Page orientation can be changed Landscape(Default Portrait) for the exported document using the export properties.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeGridData" AllowPdfExport="true"
IdMapping="TaskId" ParentIdMapping="ParentId" TreeColumnIndex="1"
Toolbar="@(<new List<string>() { "PdfExport" })">
<TreeGridEvents OnToolbarClick="ToolbarClickHandler"
TValue="TreeData"></TreeGridEvents>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
Format="C2"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText=" Start Date" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
SfTreeGrid<TreeData> TreeGrid;
public List<TreeData> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
private void
ToolbarClickHandler(Syncfusion.Blazor.Navigations.ClickEventArgs Args)
{
if(Args.Item.Text == "PDF Export")
{
Syncfusion.Blazor.Grids.PdfExportProperties ExportProperties = new
Syncfusion.Blazor.Grids.PdfExportProperties();
ExportProperties.PageOrientation =
Syncfusion.Blazor.Grids.PageOrientation.Landscape;
this.TreeGrid.PdfExport(ExportProperties);
}
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
}
}

```

```

public int? ParentId { get; set; }
public static List<TreeData> GetSelfDataSource()
{
    List<TreeData> BusinessObjectCollection = new List<TreeData>();
    BusinessObjectCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
    BusinessObjectCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child task 1", Progress = 80, Priority = "Low", ParentId = 1 });
    BusinessObjectCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
    BusinessObjectCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 2 });
    BusinessObjectCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
    BusinessObjectCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5 });
    BusinessObjectCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
    BusinessObjectCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
    BusinessObjectCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
    return BusinessObjectCollection;
}
}
}

```

[How to change page size](#)

Page size can be customized for the exported document using the export properties.

Supported page sizes are:

- Letter
- Note
- Legal
- A0
- A1
- A2
- A3
- A5
- A6
- A7
- A8
- A9
- B0
- B1
- B2
- B3
- B4
- B5
- Archa

- Archb
- Archc
- Archd
- Arche
- Flsa
- HalfLetter
- Letter11x17
- Ledger

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeGridData" AllowPdfExport="true"
IdMapping="TaskId" ParentIdMapping="ParentId" TreeColumnIndex="1"
Toolbar="@ (new List<string>() { "PdfExport" }) ">
<TreeGridEvents OnToolbarClick="ToolbarClickHandler"
TValue="TreeData"></TreeGridEvents>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
Format="C2"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText=" Start Date" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid<TreeData> TreeGrid;
public List<TreeData> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
private void
ToolbarClickHandler(Syncfusion.Blazor.Navigations.ClickEventArgs Args)
{
if(Args.Item.Text == "PDF Export")
{
Syncfusion.Blazor.Grids.PdfExportProperties ExportProperties = new
Syncfusion.Blazor.Grids.PdfExportProperties();
ExportProperties.PageSize = Syncfusion.Blazor.Grids.PdfPageSize.Letter;
this.TreeGrid.PdfExport(ExportProperties);
}
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<TreeData> GetSelfDataSource()
{
List<TreeData> BusinessObjectCollection = new List<TreeData>();
BusinessObjectCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent
Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null
});
BusinessObjectCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child
task 1", Progress = 80, Priority = "Low", ParentId = 1 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child
Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child
task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 2 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent
Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null});
BusinessObjectCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child
task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5});
BusinessObjectCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child
Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5});
BusinessObjectCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child
task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5});
BusinessObjectCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child
task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5});
return BusinessObjectCollection;
}
}
}

```

[Export current page](#)

PDF export provides an option to export the current page into PDF. To export current page, define the **exportType** to **CurrentPage**.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeGridData" AllowPdfExport="true"
IdMapping="TaskId" ParentIdMapping="ParentId" TreeColumnIndex="1"
Toolbar="@ (new List<string>() { "PdfExport" }) ">
<TreeGridEvents OnToolbarClick="ToolbarClickHandler"
TValue="TreeData"></TreeGridEvents>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>

```



```

<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
Format="C2"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText=" Start Date" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid<TreeData> TreeGrid;
public List<TreeData> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
private void
ToolbarClickHandler(Syncfusion.Blazor.Navigations.ClickEventArgs Args)
{
if(Args.Item.Text == "PDF Export")
{
Syncfusion.Blazor.Grids.PdfExportProperties ExportProperties = new
Syncfusion.Blazor.Grids.PdfExportProperties();
ExportProperties.ExportType =
Syncfusion.Blazor.Grids.ExportType.CurrentPage;
this.TreeGrid.PdfExport(ExportProperties);
}
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<TreeData> GetSelfDataSource()
{
List<TreeData> BusinessObjectCollection = new List<TreeData>();
BusinessObjectCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent
Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null
});
BusinessObjectCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child
task 1", Progress = 80, Priority = "Low", ParentId = 1 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child
Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child
task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 2 });
}
}

```

```

BusinessObjectCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
BusinessObjectCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
return BusinessObjectCollection;
}
}
}

```

Export hidden columns

PDF export provides an option to export hidden columns of the Tree Grid by defining the **includeHiddenColumn** as **true**.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeGridData" AllowPdfExport="true"
IdMapping="TaskId" ParentIdMapping="ParentId" TreeColumnIndex="1"
Toolbar="@ (new List<string>() { "PdfExport" }) ">
<TreeGridEvents OnToolbarClick="ToolbarClickHandler"
TValue="TreeData"></TreeGridEvents>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
Format="C2"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="StartDate" Visible="false" HeaderText=" Start Date"
Format="yMd" Type="Syncfusion.Blazor.Grids.ColumnType.Date"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
SfTreeGrid<TreeData> TreeGrid;
public List<TreeData> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
private void
ToolbarClickHandler(Syncfusion.Blazor.Navigations.ClickEventArgs Args)
{
if(Args.Item.Text == "PDF Export")
{

```

```

Syncfusion.Blazor.Grids.PdfExportProperties ExportProperties = new
Syncfusion.Blazor.Grids.PdfExportProperties();
ExportProperties.IncludeHiddenColumn = true;
this.TreeGrid.PdfExport(ExportProperties);
}
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<TreeData> GetSelfDataSource()
{
List<TreeData> BusinessObjectCollection = new List<TreeData>();
BusinessObjectCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent
Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null
});
BusinessObjectCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child
task 1", Progress = 80, Priority = "Low", ParentId = 1 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child
Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child
task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 2 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent
Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
BusinessObjectCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child
task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child
Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child
task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
BusinessObjectCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child
task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
return BusinessObjectCollection;
}
}
}

```

Theme

PDF export provides an option to include theme for exported PDF document.

To apply theme in exported PDF, define the **theme** in export properties.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;

```

```

<SfTreeGrid @ref="TreeGrid" DataSource="@TreeGridData" AllowPdfExport="true"
IdMapping="TaskId" ParentIdMapping="ParentId" TreeColumnIndex="1"
Toolbar="@ (new List<string>() { "PdfExport" }) ">
<TreeGridEvents OnToolbarClick="ToolbarClickHandler"
TValue="TreeData"></TreeGridEvents>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
Format="C2"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText=" Start Date" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
SfTreeGrid<TreeData> TreeGrid;
public List<TreeData> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
private void
ToolbarClickHandler(Syncfusion.Blazor.Navigations.ClickEventArgs Args)
{
if(Args.Item.Text == "PDF Export")
{
Syncfusion.Blazor.Grids.PdfExportProperties ExportProperties = new
Syncfusion.Blazor.Grids.PdfExportProperties();
Syncfusion.Blazor.Grids.PdfTheme Theme = new
Syncfusion.Blazor.Grids.PdfTheme();
Syncfusion.Blazor.Grids.PdfBorder HeaderBorder = new
Syncfusion.Blazor.Grids.PdfBorder();
HeaderBorder.Color = "#64FA50";
Syncfusion.Blazor.Grids.PdfThemeStyle HeaderThemeStyle = new
Syncfusion.Blazor.Grids.PdfThemeStyle()
{
FontColor = "#64FA50",
FontName = "Calibri",
FontSize = 17,
Bold = true,
Border = HeaderBorder
};
Theme.Header = HeaderThemeStyle;
Syncfusion.Blazor.Grids.PdfThemeStyle RecordThemeStyle = new
Syncfusion.Blazor.Grids.PdfThemeStyle()
{
FontColor = "#64FA50",
FontName = "Calibri",
FontSize = 17
};
};
}

```

```

Theme.Record = RecordThemeStyle;
Syncfusion.Blazor.Grids.PdfThemeStyle CaptionThemeStyle = new
Syncfusion.Blazor.Grids.PdfThemeStyle()
{
    FontColor = "#64FA50",
    FontName = "Calibri",
    FontSize = 17,
    Bold = true
};
Theme.Caption = CaptionThemeStyle;
ExportProperties.Theme = Theme;
this.TreeGrid.PdfExport(ExportProperties);
}
}
}

```

C#

```

namespace TreeGridComponent.Data {
    public class TreeData
    {
        public int TaskId { get; set; }
        public string TaskName { get; set; }
        public int? Duration { get; set; }
        public int? Progress { get; set; }
        public string Priority { get; set; }
        public int? ParentId { get; set; }
        public static List<TreeData> GetSelfDataSource()
        {
            List<TreeData> BusinessObjectCollection = new List<TreeData>();
            BusinessObjectCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent
Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null
});
            BusinessObjectCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child
task 1", Progress = 80, Priority = "Low", ParentId = 1 });
            BusinessObjectCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child
Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
            BusinessObjectCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child
task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 2 });
            BusinessObjectCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent
Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null});
            BusinessObjectCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child
task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5});
            BusinessObjectCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child
Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5});
            BusinessObjectCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child
task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5});
            BusinessObjectCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child
task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5});
            return BusinessObjectCollection;
        }
    }
}

```

By default, material theme is applied to exported PDF document.

Print in Blazor TreeGrid Component

To print the Tree Grid, use the [Print](#) method from the tree grid instance. The print option can be displayed on the [Toolbar](#) by adding the **Print** toolbar item.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" Toolbar="@ (new List<string>() {
    "Print" })" IdMapping="TaskId" AllowSelection="true"
    ParentIdMapping="ParentId" TreeColumnIndex="1">
    <TreeGridColumns>
    <TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
    TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
    <TreeGridColumn Field="TaskName" HeaderText="Task Name"
    Width="160"></TreeGridColumn>
    <TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
    TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
    <TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
    TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
    <TreeGridColumn Field="Priority" HeaderText="Priority"
    Width="80"></TreeGridColumn>
    </TreeGridColumns>
</SfTreeGrid>
@code{
    public List<TreeData.BusinessObject> TreeGridData { get; set; }
    protected override void OnInitialized()
    {
        this.TreeGridData = TreeData.GetSelfDataSource().ToList();
    }
}
```

C#

```
namespace TreeGridComponent.Data {
    public class TreeData
    {
        public class BusinessObject
        {
            public int TaskId { get; set; }
            public string TaskName { get; set; }
            public int? Duration { get; set; }
            public int? Progress { get; set; }
            public string Priority { get; set; }
            public int? ParentId { get; set; }
        }
        public static List<BusinessObject> GetSelfDataSource()
        {
            List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
                "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
                null });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
                "Child task 1", Progress = 80, Priority = "Low", ParentId = 1 });
        }
    }
}
```

```

BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
null});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5});
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5});
return BusinessObjectCollection;
}
}
}

```

Print				
Task ID	Task Name	Duration	Progress	Priority
1	▼ Parent Task 1	10	70	Critical
2	▼ Child task 1	4	80	Low
3	▼ Child Task 2	5	65	Critical
4	Child task 3	6	77	High
5	▼ Parent Task 2	10	70	Critical
6	Child task 1	4	80	Critical
7	Child Task 2	5	65	Low
8	Child task 3	6	77	High
9	Child task 4	6	77	Low

Page setup

Some of the print options cannot be configured through JavaScript code. So the layout, paper size, and margin options have to be customized using the browser page setup dialog. Please 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 tree grid from an external button, invoke the [Print](#) method.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<input type="button" @onclick="onClick" value="Print Tree" />
<SfTreeGrid DataSource="@TreeGridData" @ref="TreeGrid" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnName="1">
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
SfTreeGrid<TreeData.BusinessObject> TreeGrid;
public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
private void onClick()
{
TreeGrid.Print();
}
}
```

C#

```
namespace TreeGridComponent.Data {
public class TreeData
{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
}
}
```



```
public static List<BusinessObject> GetSelfDataSource()
{
    List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
    "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
    null });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
    "Child task 1", Progress = 80, Priority = "Low", ParentId = 1 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
    "Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
    });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
    "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
    "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
    null });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
    "Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
    5 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
    "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
    "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
    BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
    "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
    return BusinessObjectCollection;
}
}
```

Print Tree

Task ID	Task Name	Duration	Progress	Priority
1	▼ Parent Task 1	10	70	Critical
2	▼ Child task 1	4	80	Low
3	▼ Child Task 2	5	65	Critical
4	Child task 3	6	77	High
5	▼ Parent Task 2	10	70	Critical
6	Child task 1	4	80	Critical
7	Child Task 2	5	65	Low
8	Child task 3	6	77	High
9	Child task 4	6	77	Low

Print the visible page

By default, the tree grid prints all the pages. To print the current page alone, set the [PrintMode](#) to **CurrentPage**.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" Toolbar="@((new List<string>()) {
    "Print" })" PrintMode="Syncfusion.Blazor.Grids.PrintMode.CurrentPage"
AllowPaging="true" IdMapping="TaskId" ParentIdMapping="ParentId"
TreeColumnIndex="1">
<TreeGridPageSettings PageSize="2"
PageSizeMode="PageSizeMode.Root"></TreeGridPageSettings>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
```

```

public List<TreeData.BusinessObject> TreeGridData { get; set; }
protected override void OnInitialized()
{
    this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}


```

C#

```

namespace TreeGridComponent.Data {
    public class TreeData
    {
        public class BusinessObject
        {
            public int TaskId { get; set; }
            public string TaskName { get; set; }
            public int? Duration { get; set; }
            public int? Progress { get; set; }
            public string Priority { get; set; }
            public int? ParentId { get; set; }
        }
        public static List<BusinessObject> GetSelfDataSource()
        {
            List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
                "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
                null });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
                "Child task 1", Progress = 80, Priority = "Low", ParentId = 1 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
                "Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2
            });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
                "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
                "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId =
                null });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
                "Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId =
                5 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
                "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
                "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
                "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
            return BusinessObjectCollection;
        }
    }
}

```

 Print				
Task ID	Task Name	Duration	Progress	Priority
1	▼ Parent Task 1	10	70	Critical
2	▼ Child task 1	4	80	Low
3	▼ Child Task 2	5	65	Critical
4	Child task 3	6	77	High
5	▼ Parent Task 2	10	70	Critical
6	Child task 1	4	80	Critical
7	Child Task 2	5	65	Low
8	Child task 3	6	77	High
9	Child task 4	6	77	Low
I< < 1 2 3 > >I				1 of 3 pages (6 items)

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 tree 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 the print option panel based on the content size.



<!-- Show or Hide columns while Printing

You can show a hidden column or hide a visible column while printing the tree grid using [ToolbarClick](#) events.

In [ToolbarClick](#) event, based on **args.Item.Text** as **print**. We can show or hide columns by setting [Visible](#) of [TreeGridColumn](#) property to **true** or **false** respectively.

In [PrintComplete](#) event, We have reversed the state back to the previous state.

In the below example, we have **Duration** as a hidden column in the tree grid. While printing, we have changed **Duration** to visible column and **StartDate** as hidden column.

-->

Limitations of printing large data

When tree 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.

If printing of all the data is still needed, we suggest to Export the tree grid to **Excel** or **CSV** or **Pdf** and then print it from another non-web based application.

Clipboard in Blazor TreeGrid Component

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 Tree 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.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrids DataSource="@TreeData" IdMapping="TaskID"
ParentIdMapping="ParentID" TreeColumnIndex="1">
<TreeGridColumn>
<TreeGridColumn Field="TaskID" HeaderText="Task ID" Width="60"
TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name" Width="80">
</TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText="Start Date" Format="d"
Type=ColumnType.Date Width="90" TextAlign="TextAlign.Right">
</TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="80"
TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="80">
</TreeGridColumn>
</TreeGridColumn>
</SfTreeGrids>
@code {
public List<BusinessObject> TreeData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = BusinessObject.GetSelfDataSource().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public DateTime? StartDate { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<BusinessObject> GetSelfDataSource()
{
List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", StartDate = new DateTime(2017, 10, 23), Duration =
10, Progress = 70, Priority = "Critical", ParentId = null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", StartDate = new DateTime(2017, 10, 23), Duration = 12, Progress
= 80, Priority = "Low", ParentId = 1 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", StartDate = new DateTime(2017, 10, 24), Duration = 5, Progress
= 65, Priority = "Critical", ParentId = 2 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", StartDate = new DateTime(2017, 10, 25), Duration = 6, Priority
= "High", Progress = 77, ParentId = 3 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Child Task 5", StartDate = new DateTime(2017, 10, 26), Duration = 9,
Progress = 25, ParentId = 4, Priority = "Normal" });
}
}

```

```

BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child Task 6",StartDate = new DateTime(2017, 10, 27), Duration = 9,
Progress = 7, ParentId = 5, Priority = "Normal" });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Parent Task 3",StartDate = new DateTime(2017, 10, 28), Duration = 4,
Progress = 45, ParentId = null, Priority = "High" });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child Task 7",StartDate = new DateTime(2017, 10, 29), Duration = 3,
Progress = 38, ParentId = 7, Priority = "Critical" });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child Task 8",StartDate = new DateTime(2017, 10, 30), Duration = 7,
Progress = 70, ParentId = 7, Priority = "Low" });
return BusinessObjectCollection;
}
}
}

```

Copy to clipboard by external buttons

To copy the data of the selected rows or cells into the clipboard with help of external buttons, invoke the `copy` method.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
@using Syncfusion.Blazor.Buttons;
<SfButton OnClick="Copy">Copy</SfButton>
<SfButton OnClick="CopyHeader">Copy With Header</SfButton>
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" AllowTextWrap="true">
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText="Start Date" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="100"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code {
SfTreeGrid<BusinessObject> TreeGrid;
public List<BusinessObject> TreeData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = BusinessObject.GetSelfDataSource().ToList();
}
public async void Copy()
{
await this.TreeGrid.Copy();
}
}

```

```

}
public async void CopyHeader()
{
    await this.TreeGrid.Copy(true);
}
}

```

C#

```

namespace TreeGridComponent.Data {
    public class BusinessObject
    {
        public int TaskId { get; set; }
        public string TaskName { get; set; }
        public DateTime? StartDate { get; set; }
        public int? Duration { get; set; }
        public int? Progress { get; set; }
        public string Priority { get; set; }
        public int? ParentId { get; set; }
        public static List<BusinessObject> GetSelfDataSource()
        {
            List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1,
            TaskName = "Parent Task 1", StartDate = new DateTime(2017, 10, 23),
            Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2,
            TaskName = "Child task 1", StartDate = new DateTime(2017, 10, 23),
            Duration = 12, Progress = 80, Priority = "Low", ParentId = 1 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3,
            TaskName = "Child Task 2", StartDate = new DateTime(2017, 10, 24),
            Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4,
            TaskName = "Child task 3", StartDate = new DateTime(2017, 10, 25),
            Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5,
            TaskName = "Child Task 5", StartDate = new DateTime(2017, 10, 26),
            Duration = 9, Progress = 25, ParentId = 4, Priority = "Normal" });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6,
            TaskName = "Child Task 6", StartDate = new DateTime(2017, 10, 27),
            Duration = 9, Progress = 7, ParentId = 5, Priority = "Normal" });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7,
            TaskName = "Parent Task 3", StartDate = new DateTime(2017, 10, 28),
            Duration = 4, Progress = 45, ParentId = null, Priority = "High" });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8,
            TaskName = "Child Task 7", StartDate = new DateTime(2017, 10, 29),
            Duration = 3, Progress = 38, ParentId = 7, Priority = "Critical" });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9,
            TaskName = "Child Task 8", StartDate = new DateTime(2017, 10, 30),
            Duration = 7, Progress = 70, ParentId = 7, Priority = "Low" });
            return BusinessObjectCollection;
        }
    }
}

```


Copy Hierarchy Modes

Tree Grid provides support for a set of copy modes with [CopyHierarchyMode](#) property. The below are the type of filter mode available in the Tree Grid.

- **Parent** : This is the default copy hierarchy mode in the Tree Grid. Clipboard value will have the selected records with its parent records, if the selected records does not have any parent record then the selected record will be in clipboard.
- **Child** : Clipboard value will have the selected records with its child record. If the selected records do not have any child record then the selected records will be in clipboard.
- **Both** : Clipboard value will have the selected records with its both parent and child record. If the selected records do not have any parent and child record then the selected records alone are copied to the clipboard.
- **None** : Only the selected records will be in the clipboard.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
@using Syncfusion.Blazor.DropDowns;
<SfDropDownList TValue="string" TItem="DropDownData" @bind-Value="@CopyMode"
DataSource="@CopyModes">
<DropDownListEvents TValue="string"
ValueChange="OnTypeChange"></DropDownListEvents>
<DropDownListFieldSettings Text="Mode"
Value="Id"></DropDownListFieldSettings>
</SfDropDownList>
<SfTreeGrids @ref="TreeGrid" CopyHierarchyMode="@CopyType"
DataSource="@TreeData" IdMapping="TaskID" ParentIdMapping="ParentID"
TreeColumnIndex="1">
<TreeGridColumn>
<TreeGridColumn Field="TaskID" HeaderText="Task ID" Width="60"
TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText="Start Date" Format="d"
Type=ColumnType.Date Width="90"
TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="80"
TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress"
Width="80"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrids>
@code {
SfTreeGrids<BusinessObject> TreeGrid;
public string CopyMode { get; set; } = "Parent";
public CopyHierarchyType CopyType { get; set; } = CopyHierarchyType.Parent;
public List<BusinessObject> TreeData { get; set; }
public List<DropDownData> CopyModes { get; set; } = new
List<DropDownData>();
public class DropDownData
{
public string Id { get; set; }
public string Mode { get; set; }
}
```

```

}
protected override void OnInitialized()
{
    this.TreeData = BusinessObject.GetSelfDataSource().ToList();
    this.CopyModes.Add(new DropdownData() { Id = "Parent", Mode = "Parent" });
    this.CopyModes.Add(new DropdownData() { Id = "Child", Mode = "Child" });
    this.CopyModes.Add(new DropdownData() { Id = "Both", Mode = "Both" });
    this.CopyModes.Add(new DropdownData() { Id = "None", Mode = "None" });
}
private async void
OnChange(Syncfusion.Blazor.DropDowns.ChangeEventArgs<string> Args)
{
    if (Args.Value == "Parent")
    {
        CopyType = CopyHierarchyType.Parent;
    }
    else if (Args.Value == "Child")
    {
        CopyType = CopyHierarchyType.Child;
    }
    else if (Args.Value == "Both")
    {
        CopyType = CopyHierarchyType.Both;
    }
    else if (Args.Value == "None")
    {
        CopyType = CopyHierarchyType.None;
    }
}
}

```

C#

```

namespace TreeGridComponent.Data {
    public class BusinessObject
    {
        public int TaskId { get; set; }
        public string TaskName { get; set; }
        public DateTime? StartDate { get; set; }
        public int? Duration { get; set; }
        public int? Progress { get; set; }
        public string Priority { get; set; }
        public int? ParentId { get; set; }
        public static List<BusinessObject> GetSelfDataSource()
        {
            List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
                "Parent Task 1", StartDate = new DateTime(2017, 10, 23), Duration =
                10, Progress = 70, Priority = "Critical", ParentId = null });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
                "Child task 1", StartDate = new DateTime(2017, 10, 23), Duration = 12, Progress =
                80, Priority = "Low", ParentId = 1 });
            BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
                "Child Task 2", StartDate = new DateTime(2017, 10, 24), Duration = 5, Progress =
                65, Priority = "Critical", ParentId = 2 });
        }
    }
}

```

```

BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", StartDate = new DateTime(2017, 10, 25), Duration = 6, Priority
= "High", Progress = 77, ParentId = 3 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Child Task 5", StartDate = new DateTime(2017, 10, 26), Duration = 9,
Progress = 25, ParentId = 4, Priority = "Normal" });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child Task 6", StartDate = new DateTime(2017, 10, 27), Duration = 9,
Progress = 7, ParentId = 5, Priority = "Normal" });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Parent Task 3", StartDate = new DateTime(2017, 10, 28), Duration = 4,
Progress = 45, ParentId = null, Priority = "High" });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child Task 7", StartDate = new DateTime(2017, 10, 29), Duration = 3,
Progress = 38, ParentId = 7, Priority = "Critical" });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child Task 8", StartDate = new DateTime(2017, 10, 30), Duration = 7,
Progress = 70, ParentId = 7, Priority = "Low" });
return BusinessObjectCollection;
}
}
}

```

Paste

The content of a cell or a group of cells can be copied 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.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" AllowPaging="true"
Toolbar="@ (new List<string>() { "Add", "Delete", "Update", "Cancel" })">
<TreeGridPageSettings PageSize="2"></TreeGridPageSettings>
<TreeGridSelectionSettings
Type="Syncfusion.Blazor.Grids.SelectionType.Multiple"
Mode="Syncfusion.Blazor.Grids.SelectionMode.Cell"
CellSelectionMode="Syncfusion.Blazor.Grids.CellSelectionMode.Box"></TreeGrid
SelectionSettings>
<TreeGridEditSettings AllowAdding="true" AllowEditing="true"
AllowDeleting="true"
Mode="Syncfusion.Blazor.TreeGrid.EditMode.Batch"></TreeGridEditSettings>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="60"
IsPrimaryKey="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="155"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText="Start Date" Format="d"
Type=Syncfusion.Blazor.Grids.ColumnType.Date Width="85"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
EditType=Syncfusion.Blazor.Grids.EditType.DatePickerEdit></TreeGridColumn>

```

```

<TreeGridColumn Field="Duration" HeaderText="Duration" Width="70"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="70"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority" Width="70"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code {
public List<BusinessObject> TreeData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = BusinessObject.GetSelfDataSource().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public DateTime? StartDate { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<BusinessObject> GetSelfDataSource()
{
List<BusinessObject> BusinessObjectCollection = new List<BusinessObject>();
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 1, TaskName =
"Parent Task 1", StartDate = new DateTime(2017, 10, 23), Duration =
10, Progress = 70, Priority = "Critical", ParentId = null });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 2, TaskName =
"Child task 1", StartDate = new DateTime(2017, 10, 23), Duration = 12, Progress
= 80, Priority = "Low", ParentId = 1 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 3, TaskName =
"Child Task 2", StartDate = new DateTime(2017, 10, 24), Duration = 5, Progress
= 65, Priority = "Critical", ParentId = 2 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 4, TaskName =
"Child task 3", StartDate = new DateTime(2017, 10, 25), Duration = 6, Priority
= "High", Progress = 77, ParentId = 3 });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 5, TaskName =
"Child Task 5", StartDate = new DateTime(2017, 10, 26), Duration = 9,
Progress = 25, ParentId = 4, Priority = "Normal" });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 6, TaskName =
"Child Task 6", StartDate = new DateTime(2017, 10, 27), Duration = 9,
Progress = 7, ParentId = 5, Priority = "Normal" });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 7, TaskName =
"Parent Task 3", StartDate = new DateTime(2017, 10, 28), Duration = 4,
Progress = 45, ParentId = null, Priority = "High" });
BusinessObjectCollection.Add(new BusinessObject() { TaskId = 8, TaskName =
"Child Task 7", StartDate = new DateTime(2017, 10, 29), Duration = 3,
Progress = 38, ParentId = 7, Priority = "Critical" });
}
}

```

```

BusinessObjectCollection.Add(new BusinessObject() { TaskId = 9, TaskName =
"Child Task 8", StartDate = new DateTime(2017, 10, 30), Duration = 7,
Progress = 70, ParentId = 7, Priority = "Low" });
return BusinessObjectCollection;
}
}
}

```

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 be displayed as **NaN**. For date type cells, when the copied string format cells are pasted to date type cells then it will be displayed as an **empty cell**.

Accessibility in Blazor TreeGrid Component

Accessibility is achieved in the Tree Grid component through WAI-ARIA standard and keyboard navigations. The Tree Grid 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.

The following ARIA attributes are used in the Tree Grid:

- grid (role)
- row (role)
- gridcell (role)
- aria-selected (attribute)
- aria-expanded (attribute)
- aria-sort (attribute)
- aria-busy (attribute)
- aria-invalid (attribute)
- aria-grabbed (attribute)
- aria-owns (attribute)
- aria-label (attribute)

Keyboard navigation

Tree Grid functionalities can be interactive with keyboard shortcuts.

The following keyboard shortcuts are supported by Tree Grid.

Interaction Keys | Description

PageDown | Goes to the next page.

PageUp | Goes to the previous page.

Ctrl + Alt + PageDown | Goes to the last page.

Ctrl + Alt + PageUp | Goes to the first page.

Alt + PageDown | Goes to the next page.

Alt + PageUp | Goes to the previous page.

Home | Goes to the first cell.

End | Goes to the last cell.

Ctrl + Home | Goes to the first row.

Ctrl + End | Goes to the last row.

DownArrow | Moves the cell focus downward.

UpArrow | Moves the cell focus upward.

LeftArrow | Moves the cell focus left side.

RightArrow | Moves the cell focus right side.

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.

Enter | Moves the row/cell selection downward. If current cell is in edit state, then completes the editing. If the current cell is a header then performs sorting.

Shift + Enter | Moves the row/cell selection upward. If the current cell is a header then clears sorting for the selected column.

Ctrl + Enter | If the current cell is a header then performs multi-sorting.

Tab | Moves the cell selection right side.

Shift + Tab | Moves the cell selection left side.

Esc | Deselects all the rows/cells.

Ctrl + A | Selects all the rows/cells.

UpArrow | Moves up a row/cell selection.

DownArrow | Moves down a row/cell selection.

RightArrow | Moves to the right cell selection.

LeftArrow | Moves to the left cell selection.

Alt + DownArrow | Expands the selected group.

Ctrl + DownArrow | Expands all the visible groups.

Alt + UpArrow | Collapses the selected group.

Ctrl + UpArrow | Collapses all the visible groups.

Ctrl + P | Prints the Tree Grid.

Context Menu in Blazor TreeGrid Component

The Tree Grid has options to show the context menu when right clicked on it. To enable this feature, define either default or custom item in the [ContextMenuItems](#) property.

The following table lists the default context menu items,

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 Tree Grid data as Pdf document.

ExcelExport | Export the Tree Grid data as Excel document.

CsvExport | Export the Tree Grid data as CSV document.

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.

AddRow | Add new row to the Tree Grid.

The following sample code demonstrates enabling context menu with its default items,

ASPX-CS

```
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeData" IdMapping="TaskId" AllowPaging="true"
AllowExcelExport="true" AllowPdfExport="true" AllowSorting="true"
ParentIdMapping="ParentId" ContextMenuItems="@ (new List<object>() {
"AutoFit", "AutoFitAll", "SortAscending", "SortDescending", "Copy", "Edit",
"Delete", "Save", "Cancel", "PdfExport", "ExcelExport", "CsvExport",
"FirstPage", "PrevPage", "LastPage", "NextPage"})" TreeColumnIndex="1">
<TreeGridEditSettings AllowAdding="true" AllowEditing="true"
AllowDeleting="true" Mode="EditMode.Row"></TreeGridEditSettings>
```

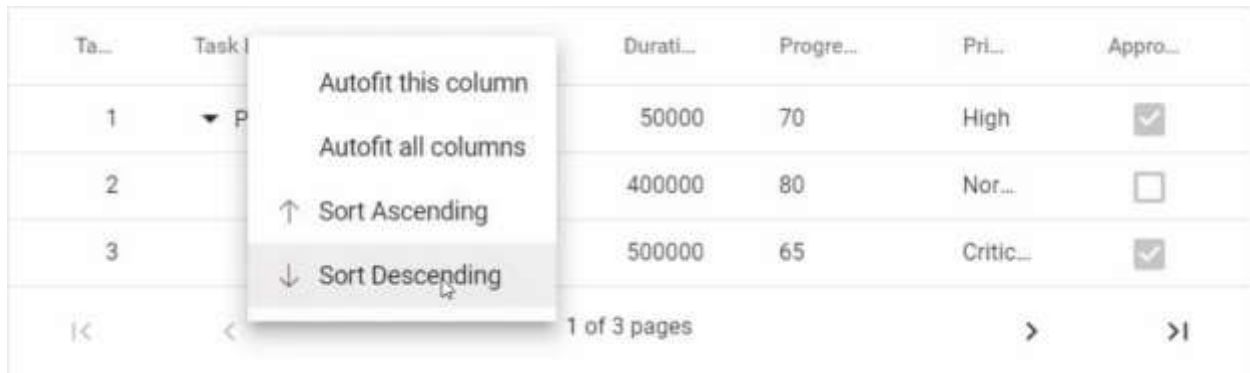
```

<TreeGridPageSettings PageSize="1"></TreeGridPageSettings>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" IsPrimaryKey="true"
Width="80" TextAlign="
Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="220"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign=" Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
EditType="Syncfusion.Blazor.Grids.EditType.NumericEdit" TextAlign="
Syncfusion.Blazor.Grids.TextAlign.Left"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Approved" HeaderText="Approved"
Type="Syncfusion.Blazor.Grids.ColumnType.Boolean"
EditType="Syncfusion.Blazor.Grids.EditType.BooleanEdit" Width="100"
DisplayAsCheckBox="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public Boolean Approved { get; set; }
}
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
= false });
TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
Approved = true });
}

```



```
TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
Approved = true });
}
```



Custom context menu items

The custom context menu items can be added by defining the [ContextMenuItems](#) as a collection of [ContextMenuModel](#). Actions for these customized items can be defined in the [ContextMenuClicked](#) event.

The following sample code demonstrates defining custom context menu item and its corresponding action in the [ContextMenuClicked](#) event,

ASPX-CS

```
@using Syncfusion.Blazor.Grids
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeData" @ref="TreeGrid" IdMapping="TaskId"
ParentIdMapping="ParentId" ContextMenuItems="@ (new
List<ContextMenuModel>() { new ContextMenuItemModel { Text = "Copy with
headers", Target = ".e-content", Id = "copywithheader" } })"
TreeColumnIndex="1">
<TreeGridEvents ContextMenuItemClicked="OnContextMenuClick"
TValue="BusinessObject"></TreeGridEvents>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" IsPrimaryKey="true"
Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
```

```
SfTreeGrid<BusinessObject> TreeGrid;
public class BusinessObject
{
    public int TaskId { get; set; }
    public string TaskName { get; set; }
    public int Duration { get; set; }
    public int Progress { get; set; }
    public string Priority { get; set; }
    public int? ParentId { get; set; }
}
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
    TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
    Duration = 50000, Progress = 70, ParentId = null, Priority = "High" });
    TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
    Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal" });
    TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
    Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical" });
    TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
    Duration = 609890, Progress = 77, ParentId = null, Priority = "Low" });
    TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
    Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal" });
    TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
    Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal" });
    TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
    Duration = 478708, Progress = 45, ParentId = null, Priority = "High" });
    TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
    Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical" });
    TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
    Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low" });
}
public void OnContextMenuClick(ContextMenuClickEventArgs args)
{
    if (args.Item.Id == "copywithheader")
    {
        this.TreeGrid.Copy(true);
    }
}
}
```

Task ID	Task Name	Duration	Progress	Priority
1	▼ Parent Task 1	50000	70	High
2	Child task 1	400000	80	Normal
3	Child Task 2	500000	65	Critical
4	▼ Parent Task 2	609890	77	Low
5	▼ Child Task 5	9778686	25	Normal
6	Child Task 6	954359	7	Normal
7	▼ Parent Task 3	478708	45	High
8	Child Task 7	36786979	38	Critical
9	Child Task 8	778907897	70	Low

Styling and Appearance in Blazor TreeGrid Component

To modify the Tree Grid appearance, override the default CSS of Tree Grid. Please find the list of CSS classes and its corresponding section in Tree Grid.

| Section | CSS class | Purpose of CSS class |

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

| **Root** | e-treegrid | This classes are in this root element div of the Tree Grid control. |

| **Header** | e-gridheader | This class is added in the root element of header element. In this class, thin line between header and content of the Tree Grid can be overridden. |

| | e-table | This class is added at the **table** of the Tree Grid header. This CSS class makes table width as 100 %. |

| | e-columnheader | This class is added at **tr** of the Tree Grid header. |

| | e-headercell | This class is added in **th** element of Tree Grid header. The background color of header and border color can be overridden. |

| | e-headercelldiv | This class is added in div which is present in **th** element in the header. Use the e-headercelldiv to override skeleton of header. |

| **Body** | e-gridcontent | This class is added at root of the body content. This is to override background color of the body. |

| | e-table | This class is added to table of the content. This CSS class makes table width as 100 %. |

| | e-altrow | This class is added to the alternate rows of Tree Grid. This is to override alternate row color of the Tree Grid. |

| | e-rowcell | This class is added to all cells in the Tree Grid. This is to override cells appearance and styling. |

| | e-groupcaption | This class is added to the **td** of group caption which is to change the background color of caption cell. |

| | e-selectionbackground | This class is added to rowcell's of the Tree Grid. This is override selection. |

| | e-hover | This class adds to row of Tree Grid, while hovering the Tree Grid rows. |

| **Pager** | e-pager | This class is added to root element of the pager. This is to change appearance of the background color and color of font. |

| | e-pagercontainer | This class is added to numeric items of the pager. |

| | e-parentmsgbar | This class is added to pager info of the pager. |

| **Summary** | e-gridfooter | This class is added to root of the summary div. |

| | e-summaryrow | This class is added to rows of Tree Grid summary. |

| | e-summarycell | This class is added to cells of summary row. This to override background color of summary. |

Events in Blazor TreeGrid Component

In this section, the list of events of the Tree Grid component is provided which will be triggered for appropriate Tree Grid actions.

The events should be provided to the Tree Grid using **TreeGridEvents** component. When using events of Tree Grid, **TValue** must be provided in the **TreeGridEvents** component.

OnActionBegin

[OnActionBegin](#) event triggers when the Tree Grid actions such as sorting, filtering, paging etc., starts.

ASPX-CS

```
@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" Toolbar="@ (new List<string>()
{ "Add", "Edit", "Delete", "Update", "Cancel" })">
<TreeGridEditSettings AllowAdding="true" AllowEditing="true"
AllowDeleting="true"
Mode="Syncfusion.Blazor.TreeGrid.EditMode.Row"></TreeGridEditSettings>
<TreeGridEvents TValue="BusinessObject"
OnActionBegin="ActionBeginHandler"></TreeGridEvents>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" IsPrimaryKey="true"
Width="80" TextAlign="
Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="220"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign=" Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
EditType="Syncfusion.Blazor.Grids.EditType.NumericEdit" TextAlign="
Syncfusion.Blazor.Grids.TextAlign.Left"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Approved" HeaderText="Approved"
Type="Syncfusion.Blazor.Grids.ColumnType.Boolean"
EditType="Syncfusion.Blazor.Grids.EditType.BooleanEdit" Width="100"
```

```

DisplayAsCheckBox="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public Boolean Approved { get; set; }
}
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
= false });
TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
Approved = true });
}
public void ActionBeginHandler(ActionEventArgs<BusinessObject> args)
{
// Here you can customize your code
}
}

```

OnActionComplete

[OnActionComplete](#) event triggers when the Tree Grid actions such as sorting, filtering, paging etc. are completed.

ASPX-CS

```

@using Syncfusion.Blazor.Grids
@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" Toolbar="@ (new List<string>()
{ "Add", "Edit", "Delete", "Update", "Cancel" })">
<TreeGridEditSettings AllowAdding="true" AllowEditing="true"
AllowDeleting="true"
Mode="Syncfusion.Blazor.TreeGrid.EditMode.Row"></TreeGridEditSettings>
<TreeGridEvents TValue="BusinessObject"
OnActionComplete="ActionCompleteHandler"></TreeGridEvents>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" IsPrimaryKey="true"
Width="80" TextAlign="
Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="220"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign=" Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
EditType="Syncfusion.Blazor.Grids.EditType.NumericEdit" TextAlign="
Syncfusion.Blazor.Grids.TextAlign.Left"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Approved" HeaderText="Approved"
Type="Syncfusion.Blazor.Grids.ColumnType.Boolean"
EditType="Syncfusion.Blazor.Grids.EditType.BooleanEdit" Width="100"
DisplayAsCheckBox="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public Boolean Approved { get; set; }
}
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
Approved = true });
}

```

```

TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
= false });
TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
Approved = true });
}
public void ActionCompleteHandler(ActionEventArgs<BusinessObject> args)
{
// Here you can customize your code
}
}

```

OnActionFailure

[OnActionFailure](#) event triggers when any Tree Grid action fails to achieve the desired results. By using this event the error details and its cause are achieved. In the below sample, the wrong field name is provided for the IdMapping property. So that it will throw the OnActionFailure event.

ASPX-CS

```

@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeData" IdMapping="TaskIDD"
ParentIdMapping="ParentId" TreeColumnIndex="1" Toolbar="@ (new List<string>()
{ "Add", "Edit", "Delete", "Update", "Cancel" })">
<TreeGridEditSettings AllowAdding="true" AllowEditing="true"
AllowDeleting="true"
Mode="Syncfusion.Blazor.TreeGrid.EditMode.Row"></TreeGridEditSettings>
<TreeGridEvents TValue="BusinessObject"
OnActionFailure="ActionFailureHandler"></TreeGridEvents>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" IsPrimaryKey="true"
Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="220"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Left"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Approved" HeaderText="Approved"
Type="Syncfusion.Blazor.Grids.ColumnType.Boolean" Width="100"

```

```

DisplayAsCheckBox="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public Boolean Approved { get; set; }
}

public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
= false });
TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
Approved = true });
}
public void ActionFailureHandler(FailureEventArgs args)
{
// Here you can get the error details in the args
}
}

```

Created

[Created](#) event triggers when the Tree Grid component is created. The Tree Grid properties can be modified by using this event.

ASPX-CS

```

@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridEvents TValue="BusinessObject"
Created="CreatedHandler"></TreeGridEvents>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80" TextAlign="
Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="220"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Left"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Approved" HeaderText="Approved"
Type="Syncfusion.Blazor.Grids.ColumnType.Boolean" Width="100"
DisplayAsCheckBox="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid<BusinessObject> TreeGrid;
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public Boolean Approved { get; set; }
}
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
= false });
}

```

```

TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
Approved = true });
}
public void CreatedHandler(object args)
{
// Here you can customize your code
}
}

```

OnLoad

[OnLoad](#) event triggers before the rendering process starts which allows customization of the Tree Grid properties before the Tree Grid rendering.

ASPX-CS

```

@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridEvents TValue="BusinessObject"
OnLoad="LoadHandler"></TreeGridEvents>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80" TextAlign="
Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="220"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Left"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Approved" HeaderText="Approved"
Type="Syncfusion.Blazor.Grids.ColumnType.Boolean" Width="100"
DisplayAsCheckBox="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid<BusinessObject> TreeGrid;
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public Boolean Approved { get; set; }
}
}

```

```

public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
    TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
    Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
    Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
    Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
    Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
    Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
    Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
    = false });
    TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
    Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
    Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
    Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
    Approved = true });
}
public void LoadHandler(object args)
{
    // Here you can customize your code
}
}

```

Destroyed

[Destroyed](#) event triggers when the Tree Grid component is destroyed. By using this event, it is confirmed that the Tree Grid gets completely destroyed.

ASPX-CS

```

@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridEvents TValue="BusinessObject"
Destroyed="DestroyHandler"></TreeGridEvents>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80" TextAlign="
Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="220"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>

```

```

<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Left"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Approved" HeaderText="Approved"
Type="Syncfusion.Blazor.Grids.ColumnType.Boolean" Width="100"
DisplayAsCheckBox="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid<BusinessObject> TreeGrid;
public class BusinessObject
{
    public int TaskId { get; set; }
    public string TaskName { get; set; }
    public int Duration { get; set; }
    public int Progress { get; set; }
    public string Priority { get; set; }
    public int? ParentId { get; set; }
    public Boolean Approved { get; set; }
}
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
    TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
    Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
    Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
    Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
    Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
    Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
    Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
    = false });
    TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
    Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
    Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
    Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
    Approved = true });
}
public void DestroyHandler(object args)
{
    // Here you can customize your code
}

```

```
}

```

OnDataBound

[OnDataBound](#) event triggers before the data is bound to Tree Grid.

ASPX-CS

```
@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridEvents TValue="BusinessObject"
OnDataBound="DataBoundHandler"></TreeGridEvents>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="220"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Left"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Approved" HeaderText="Approved"
Type="Syncfusion.Blazor.Grids.ColumnType.Boolean" Width="100"
DisplayAsCheckBox="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid<BusinessObject> TreeGrid;
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public Boolean Approved { get; set; }
}
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
Duration = 40000, Progress = 80, ParentId = 1, Priority = "Normal",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
Duration = 50000, Progress = 65, ParentId = 1, Priority = "Critical",
Approved = true });
}
```

```

TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
= false });
TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
Approved = true });
}
public void DataBoundHandler(BeforeDataBoundArgs<BusinessObject> args)
{
// Here you can customize your code
}
}

```

DataBound

[DataBound](#) event triggers when the data source is populated in the Tree Grid. The code can be customized.

ASPX-CS

```

@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridEvents TValue="BusinessObject"
DataBound="DataBoundHandler"></TreeGridEvents>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="220"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Left"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Approved" HeaderText="Approved"
Type="Syncfusion.Blazor.Grids.ColumnType.Boolean" Width="100"
DisplayAsCheckBox="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
SfTreeGrid<BusinessObject> TreeGrid;
public class BusinessObject

```

```

{
    public int TaskId { get; set; }
    public string TaskName { get; set; }
    public int Duration { get; set; }
    public int Progress { get; set; }
    public string Priority { get; set; }
    public int? ParentId { get; set; }
    public Boolean Approved { get; set; }
}

public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
    TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
    Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
    Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
    Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
    Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
    Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
    Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
    = false });
    TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
    Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
    Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
    Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
    Approved = true });
}

public void DataBoundHandler(object args)
{
    // Here you can customize your code
}
}

```

RowDataBound

[RowDataBound](#) event triggers every time a request is made to access row information, element, or data and also before the row element is appended to the Tree Grid element.

ASPX-CS

```

@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1">

```

```

<TreeGridEvents TValue="BusinessObject"
RowDataBound="RowDataBoundHandler"></TreeGridEvents>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="220"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Left"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Approved" HeaderText="Approved"
Type="Syncfusion.Blazor.Grids.ColumnType.Boolean" Width="100"
DisplayAsCheckBox="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid<BusinessObject> TreeGrid;
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public Boolean Approved { get; set; }
}

public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
= false });
TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
Approved = true });
}

```



```

TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
Approved = true });
}
public void RowDataBoundHandler(RowDataBoundEventArgs<BusinessObject> args)
{
// Here you can customize your code
}
}

```

DetailDataBound

The [DetailDataBound](#) event triggers after detail row expands.

ASPX-CS

```

@using Syncfusion.Blazor.TreeGrid
@using Syncfusion.Blazor.Grids
@inject Microsoft.AspNetCore.Components.NavigationManager UriHelper
<SfTreeGrid Height="400" DataSource="@TreeData" IdMapping="EmployeeID"
ParentIdMapping="ParentId" TreeColumnIndex="0">
<TreeGridEvents DetailDataBound="DetailDataBoundHandler"
TValue="Employee"></TreeGridEvents>
<TreeGridTemplates>
<DetailTemplate>
<div style="position: relative; display: inline-block; float: left; font-
weight: bold; width: 10%;padding:5px 4px 2px 27px; ">

</div>
<div style="padding-left: 10px; display: inline-block; width: 66%; text-
wrap: normal;font-size:13px;font-family:'Segoe UI';">
<div class="e-description" style="word-wrap: break-word;">
<b>@((context as Employee).Name)</b> was lives at @((context as
Employee).Address), @((context as Employee).Country). @((context as
Employee).Name) holds a position of <b>@((context as
Employee).Designation)</b> in our WA department, (Seattle USA).
</div>
<div class="e-description" style="word-wrap: break-word;margin-top:5px;">
<b style="margin-right:10px;">Contact:</b>@((context as Employee).Contact)
</div>
</div>
</DetailTemplate>
</TreeGridTemplates>
<TreeGridColumns>
<TreeGridColumn Field="Name" HeaderText="Name" Width="160"></TreeGridColumn>
<TreeGridColumn Field="DOB" HeaderText="DOB" Width="10"
Type="Syncfusion.Blazor.Grids.ColumnType.Date"
Format="yMd"></TreeGridColumn>
<TreeGridColumn Field="Designation" HeaderText="Designation"
Width="120"></TreeGridColumn>
<TreeGridColumn Field="EmpID" HeaderText="Employee ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>

```

```

<TreeGridColumn Field="Country" HeaderText="Country"
Width="100"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public class Employee
{
public string Name { get; set; }
public string FullName { get; set; }
public DateTime? DOB { get; set; }
public string Designation { get; set; }
public string EmpID { get; set; }
public int? EmployeeID { get; set; }
public string Country { get; set; }
public string Address { get; set; }
public string Contact { get; set; }
public int? ParentId { get; set; }
public static List<Employee> GetTemplateData()
{
List<Employee> DataCollection = new List<Employee>();
DataCollection.Add(new Employee { Name = "Robert King", FullName =
"RobertKing", Designation = "Chief Executive Officer", EmployeeID = 1, EmpID =
"EMP001", Address = "507 - 20th Ave. E.Apt. 2A, Seattle", Contact = "(206)
555-9857", Country = "USA", DOB = new DateTime(1963, 2, 15), ParentId = null
});
DataCollection.Add(new Employee { Name = "David william", FullName =
"DavidWilliam", Designation = "Vice President", EmployeeID = 2, EmpID =
"EMP004", Address = "722 Moss Bay Blvd., Kirkland", Contact = "(206) 555-
3412", Country = "USA", DOB = new DateTime(1971, 5, 20), ParentId = 1 });
DataCollection.Add(new Employee { Name = "Nancy Davolio", FullName =
"NancyDavolio", Designation = "Marketing Executive", EmployeeID = 3, EmpID =
"EMP035", Address = "4110 Old Redmond Rd., Redmond", Contact = "(206) 555-
8122", Country = "USA", DOB = new DateTime(1966, 3, 19), ParentId = 1 });
DataCollection.Add(new Employee { Name = "Andrew Fuller", FullName =
"AndrewFuller", Designation = "Sales Representative", EmployeeID = 4, EmpID =
"EMP045", Country = "UK", DOB = new DateTime(1980, 9, 20), ParentId = 1});
return DataCollection;
}
}
public Employee model = new Employee();
public IEnumerable<Employee> TreeData { get; set; }
protected override void OnInitialized()
{
this.TreeData = Employee.GetTemplateData();
}
public void DetailDataBoundHandler(DetailDataBoundEventArgs<Employee> args)
{
// Here you can customize your code
}
}

```

HeaderCellInfo

[HeaderCellInfo](#) event triggers during the rendering of every header cells in the Tree Grid so that the header cells can be customized.

ASPX-CS

```

@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridEvents TValue="BusinessObject"
HeaderCellInfo="HeaderCellInfoHandler"></TreeGridEvents>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="220"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Left"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Approved" HeaderText="Approved"
Type="Syncfusion.Blazor.Grids.ColumnType.Boolean" Width="100"
DisplayAsCheckBox="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
SfTreeGrid<BusinessObject> TreeGrid;
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public Boolean Approved { get; set; }
}
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
= false });
}

```

```

TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
Approved = true });
}
public void HeaderCellInfoHandler(HeaderCellInfoEventArgs args)
{
// Here you can customize your code
}
}

```

QueryCellInfo

[QueryCellInfo](#) event triggers every time a request is made to access cell information, element, or data and also before the cell element is appended to the Tree Grid element.

ASPX-CS

```

@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridEvents TValue="BusinessObject"
QueryCellInfo="QueryCellInfoHandler"></TreeGridEvents>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="220"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Left"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Approved" HeaderText="Approved"
Type="Syncfusion.Blazor.Grids.ColumnType.Boolean" Width="100"
DisplayAsCheckBox="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid<BusinessObject> TreeGrid;
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public Boolean Approved { get; set; }
}
}

```

```

}
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
    TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
    Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
    Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
    Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
    Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
    Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
    Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
    = false });
    TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
    Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
    Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
    Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
    Approved = true });
}
public void QueryCellInfoHandler(QueryCellInfoEventArgs<BusinessObject>
args)
{
    // Here you can customize your code
}
}

```

OnBeginEdit

[OnBeginEdit](#) event triggers before the record is to be edit.

ASPX-CS

```

@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1">
    <TreeGridEvents TValue="BusinessObject"
    OnBeginEdit="BeginEditHandler"></TreeGridEvents>
    <TreeGridEditSettings AllowAdding="true" AllowEditing="true"
    AllowDeleting="true"
    Mode="Syncfusion.Blazor.TreeGrid.EditMode.Row"></TreeGridEditSettings>
    <TreeGridColumns>

```

```

<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
IsPrimaryKey="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="220"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Left"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Approved" HeaderText="Approved"
Type="Syncfusion.Blazor.Grids.ColumnType.Boolean" Width="100"
DisplayAsCheckBox="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid<BusinessObject> TreeGrid;
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public Boolean Approved { get; set; }
}
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
= false });
TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
Approved = false });
}

```

```

TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
Approved = true });
}
public void BeginEditHandler(BeginEditArgs<BusinessObject> args)
{
// Here you can customize your code
}
}

```

OnBatchAdd

[OnBatchAdd](#) event triggers before the records are added in the batch mode.

ASPX-CS

```

@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" Toolbar="@ (new List<string>()
{ "Add", "Delete", "Update", "Cancel" })">
<TreeGridEvents TValue="BusinessObject"
OnBatchAdd="BatchAddHandler"></TreeGridEvents>
<TreeGridEditSettings AllowAdding="true" AllowEditing="true"
AllowDeleting="true"
Mode="Syncfusion.Blazor.TreeGrid.EditMode.Batch"></TreeGridEditSettings>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
IsPrimaryKey="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="220"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Left"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Approved" HeaderText="Approved"
Type="Syncfusion.Blazor.Grids.ColumnType.Boolean" Width="100"
DisplayAsCheckBox="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
SfTreeGrid<BusinessObject> TreeGrid;
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public Boolean Approved { get; set; }
}
public List<BusinessObject> TreeData = new List<BusinessObject>();

```

```
protected override void OnInitialized()
{
    TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
    Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
    Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
    Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
    Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
    Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
    Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
    = false });
    TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
    Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
    Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
    Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
    Approved = true });
}
public void BatchAddHandler(BeforeBatchAddArgs<BusinessObject> args)
{
    // Here you can customize your code
}
}
```

OnBatchSave

[OnBatchSave](#) event triggers before the records are saved in the batch mode.

ASPX-CS

```
@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" Toolbar="@ (new List<string>()
{ "Add", "Delete", "Update", "Cancel" })">
<TreeGridEvents OnBatchSave="BatchSaveHandler"
TValue="BusinessObject"></TreeGridEvents>
<TreeGridEditSettings AllowAdding="true" AllowEditing="true"
AllowDeleting="true"
Mode="Syncfusion.Blazor.TreeGrid.EditMode.Batch"></TreeGridEditSettings>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
IsPrimaryKey="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
```



```

<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="220"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Left"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Approved" HeaderText="Approved"
Type="Syncfusion.Blazor.Grids.ColumnType.Boolean" Width="100"
DisplayAsCheckBox="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid<BusinessObject> TreeGrid;
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public Boolean Approved { get; set; }
}
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
= false });
TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
Approved = true });
}

```

```
public void BatchSaveHandler(BeforeBatchSaveArgs<BusinessObject> args)
{
    // Here you can customize your code
}
}
```

OnBatchDelete

[OnBatchDelete](#) event triggers before the records are deleted in the batch mode.

ASPX-CS

```
@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" Toolbar="@ (new List<string>()
{ "Add", "Delete", "Update", "Cancel" })">
    <TreeGridEvents OnBatchDelete="BatchDeleteHandler"
    TValue="BusinessObject"></TreeGridEvents>
    <TreeGridEditSettings AllowAdding="true" AllowEditing="true"
    AllowDeleting="true"
    Mode="Syncfusion.Blazor.TreeGrid.EditMode.Batch"></TreeGridEditSettings>
    <TreeGridColumns>
        <TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
        IsPrimaryKey="true"
        TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
        <TreeGridColumn Field="TaskName" HeaderText="Task Name"
        Width="220"></TreeGridColumn>
        <TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
        TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
        <TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
        TextAlign="Syncfusion.Blazor.Grids.TextAlign.Left"></TreeGridColumn>
        <TreeGridColumn Field="Priority" HeaderText="Priority"
        Width="80"></TreeGridColumn>
        <TreeGridColumn Field="Approved" HeaderText="Approved"
        Type="Syncfusion.Blazor.Grids.ColumnType.Boolean" Width="100"
        DisplayAsCheckBox="true"
        TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center"></TreeGridColumn>
    </TreeGridColumns>
</SfTreeGrid>
@code{
    SfTreeGrid<BusinessObject> TreeGrid;
    public class BusinessObject
    {
        public int TaskId { get; set; }
        public string TaskName { get; set; }
        public int Duration { get; set; }
        public int Progress { get; set; }
        public string Priority { get; set; }
        public int? ParentId { get; set; }
        public Boolean Approved { get; set; }
    }
    public List<BusinessObject> TreeData = new List<BusinessObject>();
    protected override void OnInitialized()
    {
```

```

TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
= false });
TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
Approved = true });
}
public void BatchDeleteHandler(BeforeBatchDeleteArgs<BusinessObject> args)
{
// Here you can customize your code
}
}

```

OnCellEdit

[OnCellEdit](#) event triggers when the cell is being edited.

ASPX-CS

```

@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridEvents TValue="BusinessObject"
OnCellEdit="CellEditHandler"></TreeGridEvents>
<TreeGridEditSettings AllowAdding="true" AllowEditing="true"
AllowDeleting="true"
Mode="Syncfusion.Blazor.TreeGrid.EditMode.Cell"></TreeGridEditSettings>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
IsPrimaryKey="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="220"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>

```

```

<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Left"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Approved" HeaderText="Approved"
Type="Syncfusion.Blazor.Grids.ColumnType.Boolean" Width="100"
DisplayAsCheckBox="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid<BusinessObject> TreeGrid;
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public Boolean Approved { get; set; }
}
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
= false });
TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
Approved = true });
}
public void CellEditHandler(CellEditArgs<BusinessObject> args)
{
// Here you can customize your code
}

```

```
}

```

OnCellSave

[OnCellSave](#) event triggers when the cell is saved.

ASPX-CS

```
@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridEvents TValue="BusinessObject"
OnCellSave="CellSaveHandler"></TreeGridEvents>
<TreeGridEditSettings AllowAdding="true" AllowEditing="true"
AllowDeleting="true"
Mode="Syncfusion.Blazor.TreeGrid.EditMode.Cell"></TreeGridEditSettings>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
IsPrimaryKey="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="220"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Left"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Approved" HeaderText="Approved"
Type="Syncfusion.Blazor.Grids.ColumnType.Boolean" Width="100"
DisplayAsCheckBox="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid<BusinessObject> TreeGrid;
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public Boolean Approved { get; set; }
}
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
Approved = false });
}
```

```

TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
= false });
TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
Approved = true });
}
public void CellSaveHandler(CellSaveArgs<BusinessObject> args)
{
// Here you can customize your code
}
}

```

CellSaved

[CellSaved](#) event triggers when the cell is saved.

ASPX-CS

```

@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridEvents TValue="BusinessObject"
CellSaved="CellSavedHandler"></TreeGridEvents>
<TreeGridEditSettings AllowAdding="true" AllowEditing="true"
AllowDeleting="true"
Mode="Syncfusion.Blazor.TreeGrid.EditMode.Cell"></TreeGridEditSettings>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
IsPrimaryKey="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="220"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Left"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Approved" HeaderText="Approved"
Type="Syncfusion.Blazor.Grids.ColumnType.Boolean" Width="100"

```

```

DisplayAsCheckBox="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid<BusinessObject> TreeGrid;
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public Boolean Approved { get; set; }
}
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
= false });
TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
Approved = true });
}
public void CellSavedHandler(CellSaveArgs<BusinessObject> args)
{
// Here you can customize your code
}
}

```

RowSelecting

[RowSelecting](#) event triggers before the row selection occurs.

ASPX-CS

```

@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;

<SfTreeGrid @ref="TreeGrid" DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1">
  <TreeGridEvents RowSelecting="RowSelectingHandler"
  TValue="BusinessObject"></TreeGridEvents>
  <TreeGridColumns>
    <TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
    TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
    <TreeGridColumn Field="TaskName" HeaderText="Task Name"
    Width="220"></TreeGridColumn>
    <TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
    TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
    <TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
    TextAlign="Syncfusion.Blazor.Grids.TextAlign.Left"></TreeGridColumn>
    <TreeGridColumn Field="Priority" HeaderText="Priority"
    Width="80"></TreeGridColumn>
    <TreeGridColumn Field="Approved" HeaderText="Approved"
    Type="Syncfusion.Blazor.Grids.ColumnType.Boolean" Width="100"
    DisplayAsCheckBox="true"
    TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center"></TreeGridColumn>
  </TreeGridColumns>
</SfTreeGrid>

@code{
SfTreeGrid<BusinessObject> TreeGrid;
public class BusinessObject
{
    public int TaskId { get; set; }
    public string TaskName { get; set; }
    public int Duration { get; set; }
    public int Progress { get; set; }
    public string Priority { get; set; }
    public int? ParentId { get; set; }
    public Boolean Approved { get; set; }
}
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
    TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
    Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
    Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
    Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
    Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
    Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
    Approved = true });
}

```



```

TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
= false });
TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
Approved = true });
}
public void RowSelectingHandler(RowSelectingEventArgs<BusinessObject> args)
{
// Here you can customize your code
}
}

```

RowSelected

[RowSelected](#) event triggers when a row is selected.

ASPX-CS

```

@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridEvents RowSelected="RowSelectedHandler"
TValue="BusinessObject"></TreeGridEvents>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="220"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Left"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Approved" HeaderText="Approved"
Type="Syncfusion.Blazor.Grids.ColumnType.Boolean" Width="100"
DisplayAsCheckBox="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid<BusinessObject> TreeGrid;
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
}
}

```

```

public int? ParentId { get; set; }
public Boolean Approved { get; set; }
}
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
    TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
    Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
    Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
    Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
    Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
    Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
    Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
    = false });
    TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
    Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
    Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
    Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
    Approved = true });
}
public void RowSelectedHandler(RowSelectEventArgs<BusinessObject> args)
{
    // Here you can customize your code
}
}

```

RowDeselecting

[RowDeselecting](#) event triggers before a selected row is being deselected.

ASPX-CS

```

@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridEvents RowDeselecting="RowDeselectingHandler"
TValue="BusinessObject"></TreeGridEvents>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="220"></TreeGridColumn>

```

```

<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Left"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Approved" HeaderText="Approved"
Type="Syncfusion.Blazor.Grids.ColumnType.Boolean" Width="100"
DisplayAsCheckBox="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid<BusinessObject> TreeGrid;
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public Boolean Approved { get; set; }
}
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
= false });
TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
Approved = true });
}
public void RowDeselectingHandler(RowDeselectEventArgs<BusinessObject> args)
{

```

```
// Here you can customize your code
}
}
```

RowDeselected

[RowDeselected](#) event triggers when a selected row is deselected.

ASPX-CS

```
@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridEvents RowDeselected="RowDeselectHandler"
TValue="BusinessObject"></TreeGridEvents>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="220"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Left"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Approved" HeaderText="Approved"
Type="Syncfusion.Blazor.Grids.ColumnType.Boolean" Width="100"
DisplayAsCheckBox="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
SfTreeGrid<BusinessObject> TreeGrid;
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public Boolean Approved { get; set; }
}
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
Approved = true });
}
```

```

TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
= false });
TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
Approved = true });
}
public void RowDeselectHandler(RowDeselectEventArgs<BusinessObject> args)
{
// Here you can customize your code
}
}

```

CellSelecting

[CellSelecting](#) event triggers before the cell selection occurs.

ASPX-CS

```

@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridEvents CellSelecting="CellSelectingHandler"
TValue="BusinessObject"></TreeGridEvents>
<TreeGridSelectionSettings
Mode=SelectionMode.Cell></TreeGridSelectionSettings>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="220"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Left"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Approved" HeaderText="Approved"
Type="Syncfusion.Blazor.Grids.ColumnType.Boolean" Width="100"
DisplayAsCheckBox="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{

```

```

SfTreeGrid<BusinessObject> TreeGrid;
public class BusinessObject
{
    public int TaskId { get; set; }
    public string TaskName { get; set; }
    public int Duration { get; set; }
    public int Progress { get; set; }
    public string Priority { get; set; }
    public int? ParentId { get; set; }
    public Boolean Approved { get; set; }
}
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
    TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
    Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
    Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
    Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
    Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
    Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
    Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
    = false });
    TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
    Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
    Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
    Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
    Approved = true });
}
public void CellSelectingHandler(CellSelectingEventArgs<BusinessObject>
args)
{
    // Here you can customize your code
}
}

```

CellSelected

[CellSelected](#) event triggers after a cell is selected.

ASPX-CS

```

@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;

```

```

<SfTreeGrid @ref="TreeGrid" DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1">
  <TreeGridEvents CellSelected="CellSelectedHandler"
  TValue="BusinessObject"></TreeGridEvents>
  <TreeGridSelectionSettings
  Mode=SelectionMode.Cell></TreeGridSelectionSettings>
  <TreeGridColumn>
    <TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
    TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
    <TreeGridColumn Field="TaskName" HeaderText="Task Name"
    Width="220"></TreeGridColumn>
    <TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
    TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
    <TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
    TextAlign="Syncfusion.Blazor.Grids.TextAlign.Left"></TreeGridColumn>
    <TreeGridColumn Field="Priority" HeaderText="Priority"
    Width="80"></TreeGridColumn>
    <TreeGridColumn Field="Approved" HeaderText="Approved"
    Type="Syncfusion.Blazor.Grids.ColumnType.Boolean" Width="100"
    DisplayAsCheckBox="true"
    TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center"></TreeGridColumn>
  </TreeGridColumn>
</SfTreeGrid>
@code{
SfTreeGrid<BusinessObject> TreeGrid;
public class BusinessObject
{
  public int TaskId { get; set; }
  public string TaskName { get; set; }
  public int Duration { get; set; }
  public int Progress { get; set; }
  public string Priority { get; set; }
  public int? ParentId { get; set; }
  public Boolean Approved { get; set; }
}
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
  TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
  Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
  Approved = true });
  TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
  Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
  Approved = false });
  TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
  Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
  Approved = true });
  TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
  Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
  Approved = false });
  TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
  Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
  Approved = true });
  TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
  Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
  = false });
}

```

```

TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
Approved = true });
}
public void CellSelectedHandler(CellSelectEventArgs<BusinessObject> args)
{
// Here you can customize your code
}
}

```

CellDeselecting

[CellDeselecting](#) event triggers before a cell is deselected.

ASPX-CS

```

@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridEvents CellDeselecting="CellDeselectingHandler"
TValue="BusinessObject"></TreeGridEvents>
<TreeGridSelectionSettings
Mode=SelectionMode.Cell></TreeGridSelectionSettings>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="220"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Left"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Approved" HeaderText="Approved"
Type="Syncfusion.Blazor.Grids.ColumnType.Boolean" Width="100"
DisplayAsCheckBox="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
SfTreeGrid<BusinessObject> TreeGrid;
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
}
}

```



```

public Boolean Approved { get; set; }
}
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
    TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
    Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
    Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
    Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
    Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
    Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
    Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
    = false });
    TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
    Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
    Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
    Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
    Approved = true });
}
public void CellDeselectingHandler(CellDeselectEventArgs<BusinessObject>
args)
{
    // Here you can customize your code
}
}

```

CellDeselected

[CellDeselected](#) event triggers after a cell is deselected.

ASPX-CS

```

@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridEvents CellDeselected="CellDeselectedHandler"
TValue="BusinessObject"></TreeGridEvents>
<TreeGridSelectionSettings
Mode=SelectionMode.Cell></TreeGridSelectionSettings>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>

```

```

<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="220"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Left"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Approved" HeaderText="Approved"
Type="Syncfusion.Blazor.Grids.ColumnType.Boolean" Width="100"
DisplayAsCheckBox="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid<BusinessObject> TreeGrid;
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public Boolean Approved { get; set; }
}
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
= false });
TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
Approved = true });
}

```

```
public void CellDeselectedHandler(CellDeselectEventArgs<BusinessObject>
args)
{
// Here you can customize your code
}
}
```

OnRecordDoubleClick

[OnRecordDoubleClick](#) event triggers when a record is double clicked.

ASPX-CS

```
@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridEvents OnRecordDoubleClick="RecordDoubleClickHandler"
TValue="BusinessObject"></TreeGridEvents>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="220"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Left"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Approved" HeaderText="Approved"
Type="Syncfusion.Blazor.Grids.ColumnType.Boolean" Width="100"
DisplayAsCheckBox="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid<BusinessObject> TreeGrid;
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public Boolean Approved { get; set; }
}
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
Approved = false });
}
```

```

TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
= false });
TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
Approved = true });
}
public void
RecordDoubleClickHandler(RecordDoubleClickEventArgs<BusinessObject> args)
{
// Here you can customize your code
}
}

```

OnToolbarClick

[OnToolbarClick](#) event triggers when a toolbar item is clicked.

ASPX-CS

```

@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" Toolbar="@ (new List<string>()
{ "Add", "Edit", "Delete", "Cancel", "Update" })">
<TreeGridEvents OnToolbarClick="ToolbarClickHandler"
TValue="BusinessObject"></TreeGridEvents>
<TreeGridEditSettings AllowAdding="true" AllowEditing="true"
AllowDeleting="true" Mode="EditMode.Row"></TreeGridEditSettings>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
IsPrimaryKey="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="220"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Left"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Approved" HeaderText="Approved"
Type="Syncfusion.Blazor.Grids.ColumnType.Boolean" Width="100"

```

```

DisplayAsCheckBox="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid<BusinessObject> TreeGrid;
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public Boolean Approved { get; set; }
}
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
= false });
TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
Approved = true });
}
public void ToolbarClickHandler(Syncfusion.Blazor.Navigations.ClickEventArgs
Args)
{
// Here you can customize your code
}
}

```

CommandClicked

[CommandClicked](#) event triggers when the command button is clicked. It provides the row data of the currently clicked row.

ASPX-CS

```
@using Syncfusion.Blazor.TreeGrid;
@using Syncfusion.Blazor.Grids;
<SfTreeGrid DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridEvents CommandClicked="OnCommandClicked"
TValue="BusinessObject"></TreeGridEvents>
<TreeGridEditSettings AllowEditing="true" AllowAdding="true"
AllowDeleting="true" Mode="Syncfusion.Blazor.TreeGrid.EditMode.Row" />
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" IsPrimaryKey="true"
Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn HeaderText="Manage Records" Width="150">
<TreeGridCommandColumns>
<TreeGridCommandColumn Type="CommandButtonType.Edit" ButtonOption="@ (new
CommandButtonOptions() {IconCss="e-icons e-edit", CssClass="e-flat"
}) "></TreeGridCommandColumn>
<TreeGridCommandColumn Type="CommandButtonType.Delete" ButtonOption="@ (new
CommandButtonOptions() {IconCss="e-icons e-delete", CssClass="e-flat"
}) "></TreeGridCommandColumn>
<TreeGridCommandColumn Type="CommandButtonType.Save" ButtonOption="@ (new
CommandButtonOptions() {IconCss="e-icons e-save", CssClass="e-flat"
}) "></TreeGridCommandColumn>
<TreeGridCommandColumn Type="CommandButtonType.Cancel" ButtonOption="@ (new
CommandButtonOptions() {IconCss="e-icons e-cancel-icon", CssClass="e-flat"
}) "></TreeGridCommandColumn>
</TreeGridCommandColumns>
</TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
SfTreeGrid<BusinessObject> TreeGrid;
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public Boolean Approved { get; set; }
}
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{

```

```

TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
= false });
TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
Approved = true });
}
public void OnCommandClicked(CommandClickEventArgs<BusinessObject> args)
{
// Perform required operations here
}
}

```

ColumnMenuItemClicked

[ColumnMenuItemClicked](#) event triggers when the column menu is clicked.

ASPX-CS

```

@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" ShowColumnMenu="true"
AllowFiltering="true" AllowMultiSorting="true" AllowPaging="true">
<TreeGridEvents ColumnMenuItemClicked="ColumnMenuItemClickedHandler"
TValue="BusinessObject"></TreeGridEvents>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
Format="C2"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText="Start Date" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date"

```

```

TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public Boolean Approved { get; set; }
}
SfTreeGrid<BusinessObject> TreeGrid;
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
= false });
TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
Approved = true });
}
public void ColumnMenuItemClickedHandler(ColumnMenuClickEventArgs args)
{
// Here you can customize your code
}
}

```


ContextMenuItemClicked

[ContextMenuItemClicked](#) event triggers when the context menu is clicked.

ASPX-CS

```
@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeData" IdMapping="TaskId" AllowPaging="true"
AllowExcelExport="true" AllowPdfExport="true" AllowSorting="true"
ParentIdMapping="ParentId" ContextMenuItems="@ (new List<object>() {
"AutoFit", "AutoFitAll", "SortAscending", "SortDescending", "Copy", "Edit",
"Delete", "Save", "Cancel", "PdfExport", "ExcelExport", "CsvExport",
"FirstPage", "PrevPage", "LastPage", "NextPage"})" TreeColumnIndex="1">
<TreeGridEditSettings AllowAdding="true" AllowEditing="true"
AllowDeleting="true"
Mode="Syncfusion.Blazor.TreeGrid.EditMode.Row"></TreeGridEditSettings>
<TreeGridPageSettings PageSize="1"></TreeGridPageSettings>
<TreeGridEvents ContextMenuItemClicked="ContextMenuItemClickedHandler"
TValue="BusinessObject"></TreeGridEvents>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" IsPrimaryKey="true"
Width="80" TextAlign="
Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="220"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign=" Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
EditType="Syncfusion.Blazor.Grids.EditType.NumericEdit" TextAlign="
Syncfusion.Blazor.Grids.TextAlign.Left"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Approved" HeaderText="Approved"
Type="Syncfusion.Blazor.Grids.ColumnType.Boolean"
EditType="Syncfusion.Blazor.Grids.EditType.BooleanEdit" Width="100"
DisplayAsCheckBox="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public Boolean Approved { get; set; }
}
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
Approved = true });
}
```

```

TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
= false });
TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
Approved = true });
}
public void
ContextMenuClickedHandler(ContextMenuClickEventArgs<BusinessObject>
args)
{
// Here you can customize your code
}
}

```

ContextMenuOpen

[ContextMenuOpen](#) event triggers before opening the context menu.

ASPX-CS

```

@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeData" IdMapping="TaskId" AllowPaging="true"
AllowExcelExport="true" AllowPdfExport="true" AllowSorting="true"
ParentIdMapping="ParentId" ContextMenuItems="@ (new List<object>() {
"AutoFit", "AutoFitAll", "SortAscending", "SortDescending", "Copy", "Edit",
"Delete", "Save", "Cancel", "PdfExport", "ExcelExport", "CsvExport",
"FirstPage", "PrevPage", "LastPage", "NextPage"})" TreeColumnIndex="1">
<TreeGridEditSettings AllowAdding="true" AllowEditing="true"
AllowDeleting="true"
Mode="Syncfusion.Blazor.TreeGrid.EditMode.Row"></TreeGridEditSettings>
<TreeGridPageSettings PageSize="1"></TreeGridPageSettings>
<TreeGridEvents ContextMenuOpen="ContextMenuOpenHandler"
TValue="BusinessObject"></TreeGridEvents>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" IsPrimaryKey="true"
Width="80" TextAlign="
Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>

```

```

<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="220"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign=" Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
EditType="Syncfusion.Blazor.Grids.EditType.NumericEdit" TextAlign="
Syncfusion.Blazor.Grids.TextAlign.Left"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
<TreeGridColumn Field="Approved" HeaderText="Approved"
Type="Syncfusion.Blazor.Grids.ColumnType.Boolean"
EditType="Syncfusion.Blazor.Grids.EditType.BooleanEdit" Width="100"
DisplayAsCheckBox="true"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Center"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public Boolean Approved { get; set; }
}
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
= false });
TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
Approved = true });
}

```

```

}
public void ContextMenuOpenHandler(ContextMenuOpenEventArgs<BusinessObject>
args)
{
// Here you can customize your code
}
}

```

OnPdfExport

[OnPdfExport](#) event triggers before the Tree Grid data is exported to PDF document.

ASPX-CS

```

@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeData" AllowPdfExport="true"
IdMapping="TaskId" ParentIdMapping="ParentId" TreeColumnIndex="1"
Toolbar="@ (new List<string>() { "PdfExport" })">
<TreeGridEvents OnPdfExport="PdfExportHandler"
OnToolbarClick="ToolbarClickHandler"
TValue="BusinessObject"></TreeGridEvents>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
Format="C2"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText=" Start Date" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public Boolean Approved { get; set; }
}
SfTreeGrid<BusinessObject> TreeGrid;
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
Approved = true });
}
}

```

```

TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
= false });
TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
Approved = true });
}
public async Task
ToolbarClickHandler(Syncfusion.Blazor.Navigations.ClickEventArgs Args)
{
if (Args.Item.Text == "PDF Export")
{
await this.TreeGrid.PdfExport();
}
}
public void PdfExportHandler(object args)
{
// Here you can customize your code
}
}

```

[PdfQueryCellInfoEvent](#)

[PdfQueryCellInfoEvent](#) event triggers before the Tree Grid data is exported to PDF document. It can be used to customize the Tree Grid content in PDF document.

ASPX-CS

```

@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeData" AllowPdfExport="true"
IdMapping="TaskId" ParentIdMapping="ParentId" TreeColumnIndex="1"
Toolbar="@ (new List<string>() { "PdfExport" })">
<TreeGridEvents PdfQueryCellInfoEvent="PdfQueryCellInfoHandler"
OnToolbarClick="ToolbarClickHandler"
TValue="BusinessObject"></TreeGridEvents>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>

```

```

<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
Format="C2"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText=" Start Date" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public Boolean Approved { get; set; }
}
SfTreeGrid<BusinessObject> TreeGrid;
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
= false });
TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
Approved = true });
}

```

```
private async Task
ToolbarClickHandler(Syncfusion.Blazor.Navigations.ClickEventArgs Args)
{
    if (Args.Item.Text == "PDF Export")
    {
        await this.TreeGrid.PdfExport();
    }
}

public void
PdfQueryCellInfoHandler(PdfQueryCellInfoEventArgs<BusinessObject> args)
{
    // Here you can customize your code
}
}
```

OnExcelExport

[OnExcelExport](#) event triggers before the Tree Grid data is exported to the excel file.

ASPX-CS

```
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeData" AllowExcelExport="true"
IdMapping="TaskId" ParentIdMapping="ParentId" TreeColumnIndex="1"
Toolbar="@ (new List<string>() { "Excel Export" })">
<TreeGridEvents OnExcelExport="ExcelExportHandler"
OnToolbarClick="ToolbarClickHandler"
TValue="BusinessObject"></TreeGridEvents>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
Format="C2"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText=" Start Date" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public class BusinessObject
{
    public int TaskId { get; set; }
    public string TaskName { get; set; }
    public int Duration { get; set; }
    public int Progress { get; set; }
    public string Priority { get; set; }
    public int? ParentId { get; set; }
    public Boolean Approved { get; set; }
}
SfTreeGrid<BusinessObject> TreeGrid;
public List<BusinessObject> TreeData = new List<BusinessObject>();
```

```
protected override void OnInitialized()
{
    TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
    Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
    Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
    Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
    Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
    Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
    Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
    = false });
    TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
    Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
    Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
    Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
    Approved = true });
}
private async Task
ToolbarClickHandler(Syncfusion.Blazor.Navigations.ClickEventArgs Args)
{
    if (Args.Item.Text == "Excel Export")
    {
        await this.TreeGrid.ExcelExport();
    }
}
public void ExcelExportHandler(object args)
{
    // Here you can customize your code
}
}
```

ExcelQueryCellInfoEvent

[ExcelQueryCellInfoEvent](#) event triggers before the Tree Grid data is exported to the Excel file. It can be used to customize the Tree Grid content in Excel file.

ASPX-CS

```
@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeData" AllowExcelExport="true"
    IdMapping="TaskId" ParentIdMapping="ParentId" TreeColumnIndex="1"
    Toolbar="@ (new List<string>() { "Excel Export" }) ">
```



```

<TreeGridEvents ExcelQueryCellInfoEvent="ExcelQueryCellInfoHandler"
OnToolbarClick="ToolbarClickHandler"
TValue="BusinessObject"></TreeGridEvents>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
Format="C2"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText=" Start Date" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public Boolean Approved { get; set; }
}
SfTreeGrid<BusinessObject> TreeGrid;
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
= false });
TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
Approved = true });
}

```

```

TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
Approved = true });
}
private async Task
ToolbarClickHandler(Syncfusion.Blazor.Navigations.ClickEventArgs Args)
{
if (Args.Item.Text == "Excel Export")
{
await this.TreeGrid.ExcelExport();
}
}
public void
ExcelQueryCellInfoHandler(ExcelQueryCellInfoEventArgs<BusinessObject> args)
{
// Here you can customize your code
}
}

```

OnResizeStart

[OnResizeStart](#) event triggers when the column resize starts.

ASPX-CS

```

@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" AllowResizing="true">
<TreeGridEvents OnResizeStart="OnResizeStartHanlder"
TValue="BusinessObject"></TreeGridEvents>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
Format="C2"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText=" Start Date" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
}
}

```

```

public string Priority { get; set; }
public int? ParentId { get; set; }
public Boolean Approved { get; set; }
}
SfTreeGrid<BusinessObject> TreeGrid;
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
    TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
    Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
    Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
    Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
    Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
    Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
    Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
    = false });
    TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
    Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
    Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
    Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
    Approved = true });
}
public void OnResizeStartHanlder(ResizeArgs args)
{
    // Here you can customize your code
}
}

```

ResizeStopped

[ResizeStopped](#) event triggers when the column resize ends.

ASPX-CS

```

@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" AllowResizing="true">
<TreeGridEvents ResizeStopped="ResizeStoppedHanlder"
TValue="BusinessObject"></TreeGridEvents>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>

```

```

<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
Format="C2"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText=" Start Date" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public Boolean Approved { get; set; }
}
SfTreeGrid<BusinessObject> TreeGrid;
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
= false });
TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
Approved = true });
}
public void ResizeStoppedHanlder(ResizeArgs args)

```

```
{
// Here you can customize your code
}
```

Expanding

[Expanding](#) event triggers when a row is expanding.

ASPX-CS

```
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridEvents Expanding="ExpandingHandler"
TValue="BusinessObject"></TreeGridEvents>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
Format="C2"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText=" Start Date" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public Boolean Approved { get; set; }
}
SfTreeGrid<BusinessObject> TreeGrid;
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
Approved = true });
```

```

TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
= false });
TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
Approved = true });
TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
Approved = false });
TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
Approved = true });
}
public void ExpandingHandler(RowExpandingEventArgs<BusinessObject> args)
{
// Here you can customize your code
}
}

```

Expanded

[Expanded](#) event triggers when a row is expanded.

ASPX-CS

```

@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridEvents Expanded="ExpandedHandler"
TValue="BusinessObject"></TreeGridEvents>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
Format="C2"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText=" Start Date" Format="yMd"
Type="Syncfusion.Blazor.Grids.ColumnType.Date"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"
Width="100"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
}
}

```

```

public int Duration { get; set; }
public int Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public Boolean Approved { get; set; }
}
SfTreeGrid<BusinessObject> TreeGrid;
public List<BusinessObject> TreeData = new List<BusinessObject>();
protected override void OnInitialized()
{
    TreeData.Add(new BusinessObject() { TaskId = 1, TaskName = "Parent Task 1",
    Duration = 50000, Progress = 70, ParentId = null, Priority = "High",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 2, TaskName = "Child task 1",
    Duration = 400000, Progress = 80, ParentId = 1, Priority = "Normal",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 3, TaskName = "Child Task 2",
    Duration = 500000, Progress = 65, ParentId = 1, Priority = "Critical",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 4, TaskName = "Parent Task 2",
    Duration = 609890, Progress = 77, ParentId = null, Priority = "Low",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 5, TaskName = "Child Task 5",
    Duration = 9778686, Progress = 25, ParentId = 4, Priority = "Normal",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 6, TaskName = "Child Task 6",
    Duration = 954359, Progress = 7, ParentId = 5, Priority = "Normal", Approved
    = false });
    TreeData.Add(new BusinessObject() { TaskId = 7, TaskName = "Parent Task 3",
    Duration = 478708, Progress = 45, ParentId = null, Priority = "High",
    Approved = true });
    TreeData.Add(new BusinessObject() { TaskId = 8, TaskName = "Child Task 7",
    Duration = 36786979, Progress = 38, ParentId = 7, Priority = "Critical",
    Approved = false });
    TreeData.Add(new BusinessObject() { TaskId = 9, TaskName = "Child Task 8",
    Duration = 778907897, Progress = 70, ParentId = 7, Priority = "Low",
    Approved = true });
}
public void ExpandedHandler(RowExpandedEventArgs<BusinessObject> args)
{
    // Here you can customize your code
}
}

```

We are not going to limit Tree Grid with these events, we will be adding new events in future based on the user requests. If the event, you are looking for is not in the list, then please request [here](#).

How To

Show or Hide columns in Dialog editing in Blazor TreeGrid Component

The hidden columns can be shown or visible columns' editor can be hidden in the dialog while editing the Tree Grid record. This can be achieved by **Template**. In the following example, the Tree Grid columns' **Progress** are rendered as hidden column and **Priority** as visible column. In the edit mode, the **Progress** column is changed to visible state and **Priority** column to hidden state.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.Grids
@using Syncfusion.Blazor.TreeGrid
@using Syncfusion.Blazor.DropDowns
@using Syncfusion.Blazor.Inputs
@using Syncfusion.Blazor.Calendars
<SfTreeGrid DataSource="@TreeGridData" AllowPaging="true" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1" Toolbar="@ (new List<string>()
{ "Add", "Edit", "Delete", "Update", "Cancel" }) ">
<TreeGridEvents TValue="TreeData"
OnActionComplete="OnComplete"></TreeGridEvents>
<TreeGridEditSettings AllowEditing="true" AllowAdding="true"
AllowDeleting="true" Mode="Syncfusion.Blazor.TreeGrid.EditMode.Dialog"
NewRowPosition="RowPosition.Child">
<Template>
@{
var employee = (context as TreeData);
}
<div>
<div class="form-row">
<div class="form-group col-md-6">
<SfNumericTextBox ID="TaskId" @bind-Value="@ (employee.TaskId) "
Enabled="@Check" FloatLabelType="FloatLabelType.Always" Placeholder="Task
ID"></SfNumericTextBox>
</div>
<div class="form-group col-md-6">
<SfAutoComplete TItem="TreeData" ID="TaskName" @bind-
Value="@ (employee.TaskName) " TValue="string" DataSource="@TreeGridData"
FloatLabelType="FloatLabelType.Always" Placeholder="Task Name">
<AutoCompleteFieldSettings Value="TaskName"></AutoCompleteFieldSettings>
</SfAutoComplete>
</div>
</div>
<div class="form-row">
<div class="form-group col-md-6">
<SfNumericTextBox ID="Duration" @bind-Value="@ (employee.Duration) "
TValue="int?" FloatLabelType="FloatLabelType.Always"
Placeholder="Duration"></SfNumericTextBox>
</div>
<div class="form-group col-md-6">
<SfNumericTextBox ID="Progress" @bind-Value="@ (employee.Progress) "
TValue="int?" FloatLabelType="FloatLabelType.Always"
Placeholder="Progress"></SfNumericTextBox>
</div>
</div>
</div>
</Template>
</TreeGridEditSettings>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" IsPrimaryKey="true"
Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>

```



```

<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
Visible="false"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
private Boolean Check = false;
public List<TreeData> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
private void OnComplete(ActionEventArgs<TreeData> args)
{
if (args.RequestType.ToString() == "Add")
{
Check = true;
}
else
{
Check = false;
}
}
}
}

```

C#

```

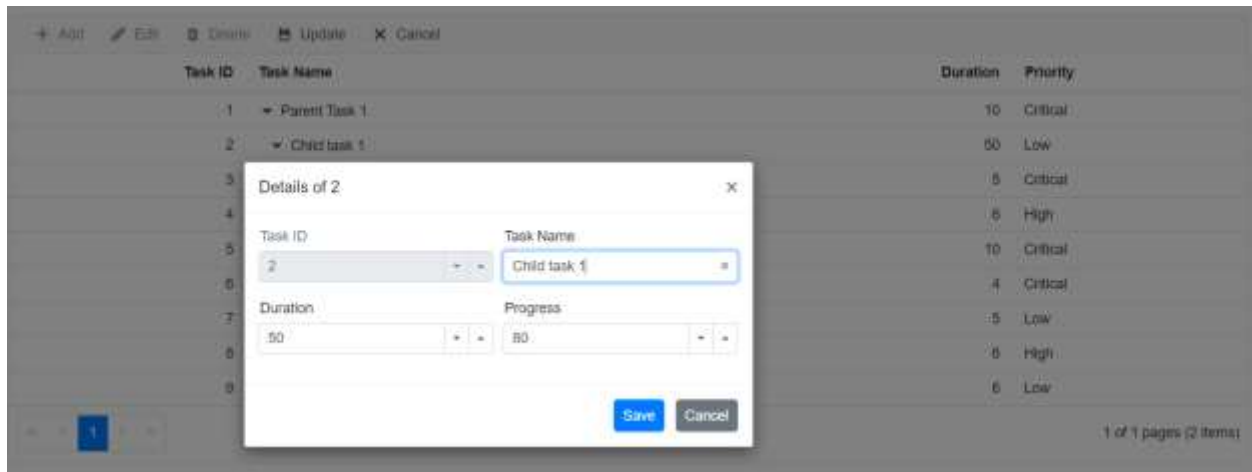
namespace TreeGridComponent.Data {
public class TreeData
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<TreeData> GetSelfDataSource()
{
List<TreeData> TreeDataCollection = new List<TreeData>();
TreeDataCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent Task
1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
TreeDataCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child task
1", Progress = 80, Priority = "Low", Duration = 50, ParentId = 1 });
TreeDataCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child Task
2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
TreeDataCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child task
3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
TreeDataCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent Task
2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
TreeDataCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child task
1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5 });
}
}
}

```

```

TreeDataCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
return TreeDataCollection;
}
}
}

```



Create custom toolbar with drop-down list in Blazor TreeGrid Component

ToolBar items can be created in the Tree Grid. It can be added by defining the [ToolBar](#). Actions for this ToolBar template items are defined in the [ToolBarClick]

Step 1:

Initialize the template for the custom component. Using the following code add the DropDownList component to the ToolBar.

ASPX-CS

```

<SfToolBar>
<ToolBarItems>
<ToolBarItem Type="ItemType.Input">
<Template>
<SfDropDownList TValue="string" TItem="Select" DataSource=@LocalData
Width="200">
<DropDownListFieldSettings Text="text" Value="text">
</DropDownListFieldSettings>
<DropDownListEvents TValue="string" ValueChange="OnChange" TItem="Select">
</DropDownListEvents>
</SfDropDownList>
</Template>
</ToolBarItem>
</ToolBarItems>
</SfToolBar>

```

Step 2:

To render the DropDownList component, use the [DropDownListEvents](#). The Tree Grid row index can be selected based on the selected data in the DropDownList.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.Grids
@using Syncfusion.Blazor.Navigations
@using Syncfusion.Blazor.DropDowns
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeGridData" AllowPaging="true"
IdMapping="TaskId" ParentIdMapping="ParentId" TreeColumnIndex="1"
Height="200">
<SfToolbar>
<ToolbarItems>
<ToolbarItem Type="ItemType.Input">
<Template>
<SfDropDownList TValue="string" TItem="Select" DataSource=@LocalData
Width="200">
<DropDownListFieldSettings Text="text" Value="text">
</DropDownListFieldSettings>
<DropDownListEvents TValue="string" ValueChange="OnChange" TItem="Select">
</DropDownListEvents>
</SfDropDownList>
</Template>
</ToolbarItem>
</ToolbarItems>
</SfToolbar>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" IsPrimaryKey="true"
Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
SfTreeGrid<TreeData> TreeGrid;
public List<TreeData> TreeGridData { get; set; }
public class Select
{
public string text { get; set; }
}
List<Select> LocalData = new List<Select>
{
new Select() { text = "0"},
new Select() { text = "1"},
new Select() { text = "2"},
new Select() { text = "3"},
new Select() { text = "4"},
new Select() { text = "5"},
new Select() { text = "6"},
}
```

```

new Select() { text = "7"},
new Select() { text = "8"},
new Select() { text = "9"},
};
protected override void OnInitialized()
{
    this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
public void OnChange(Syncfusion.Blazor.DropDowns.ChangeEventArgs<string,
Select> args)
{
    this.TreeGrid.SelectRow(int.Parse(args.Value));
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
    public int TaskId { get; set; }
    public string TaskName { get; set; }
    public int? Duration { get; set; }
    public int? Progress { get; set; }
    public string Priority { get; set; }
    public int? ParentId { get; set; }
    public static List<TreeData> GetSelfDataSource()
    {
        List<TreeData> TreeDataCollection = new List<TreeData>();
        TreeDataCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent Task
1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
        TreeDataCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child task
1", Progress = 80, Priority = "Low", Duration = 50, ParentId = 1 });
        TreeDataCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child Task
2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
        TreeDataCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child task
3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
        TreeDataCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent Task
2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
        TreeDataCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child task
1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5 });
        TreeDataCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child Task
2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
        TreeDataCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child task
3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
        TreeDataCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child task
4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
        return TreeDataCollection;
    }
}
}

```



Task ID	Task Name	Duration	Progress	Priority
1	Parent Task 1	10	70	Critical
2	Child task 1	50	80	Low
3	Child Task 2	5	65	Critical
4	Child task 3	6	77	High
5	Parent Task 2	10	70	Critical
6	Child task 1	4	80	Critical

1 of 1 pages (2 items)

Customize Column Styles in Blazor TreeGrid Component

The appearance of the header and content of a particular column can be customized using the [CustomAttributes](#) property.

To customize the Tree Grid column, follow the given steps:

Step 1:

Create a CSS class with custom style to override the default style for row cell and header cell.

CSS

```
.e-attr{
background: #5DADE2;
font-family: "Bell MT";
color: red;
font-size: 5px;
}
```

Step 2:

Add the custom CSS class to the specified column by using the [CustomAttributes](#) property.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId"
TreeColumnIndex="1" AllowPaging="true" Height="200">
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
CustomAttributes="@ (new Dictionary<string, object>() { { "class", "e-attr" }
}) " Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
<style>
```

```

.e-attr {
background: #5DADE2;
font-family: "Bell MT";
color: red;
font-size: 5px;
}
</style>
@code{
SfTreeGrid<TreeData> TreeGrid;
public List<TreeData> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<TreeData> GetSelfDataSource()
{
List<TreeData> TreeDataCollection = new List<TreeData>();
TreeDataCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent Task
1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
TreeDataCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child task
1", Progress = 80, Priority = "Low", Duration = 50, ParentId = 1 });
TreeDataCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child Task
2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
TreeDataCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child task
3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
TreeDataCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent Task
2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
TreeDataCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child task
1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child Task
2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child task
3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child task
4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
return TreeDataCollection;
}
}
}

```

Task ID	Task Name	Duration	Progress	Priority
1	Parent Task 1	10	70	Critical
2	Child task 1	50	80	Low
3	Child Task 2	5	65	Critical
4	Child task 3	6	77	High
5	Parent Task 2	10	70	Critical
6	Child task 4	4	80	Critical

1 of 1 pages (2 items)

Access public methods in Tree Grid in Blazor TreeGrid Component

The public methods available in the Tree Grid component can be accessed by using its reference defined in the component initialization.

This is demonstrated in the below sample code where the [Print](#) method of the Tree Grid component is invoked on button click using the Tree Grid reference,

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfButton OnClick="Print" CssClass="e-primary" IsPrimary="true"
Content="Print data"></SfButton>
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId"
TreeColumnIndex="1" AllowPaging="true" Height="200">
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
SfTreeGrid<TreeData> TreeGrid;
public List<TreeData> TreeGridData { get; set; }
public void Print()
{
this.TreeGrid.Print();
}
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}
```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<TreeData> GetSelfDataSource()
{
List<TreeData> TreeDataCollection = new List<TreeData>();
TreeDataCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent Task
1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
TreeDataCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child task
1", Progress = 80, Priority = "Low", Duration = 50, ParentId = 1 });
TreeDataCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child Task
2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
TreeDataCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child task
3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
TreeDataCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent Task
2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
TreeDataCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child task
1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child Task
2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child task
3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child task
4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
return TreeDataCollection;
}
}
}

```

Similarly all the public methods of the Tree Grid can be invoked. The available public methods can be found in this [link](#).

Change datasource dynamically in Blazor TreeGrid Component

The [DataSource](#) of the Tree Grid component can be changed dynamically through an external button.

This is demonstrated in the below sample code where the [DataSource](#) is dynamically modified using the bounded property.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfButton OnClick="Change">Change data source dynamically</SfButton>
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId"
TreeColumnIndex="1" AllowPaging="true" Height="200">
<TreeGridPageSettings PageSize="8"></TreeGridPageSettings>
<TreeGridColumns>

```



```

<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
SfTreeGrid<TreeData> TreeGrid;
public List<TreeData> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
public void Change()
{
// Data source is modified dynamically
this.TreeGridData = TreeData.GetSelfChangedDataSource().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<TreeData> GetSelfDataSource()
{
List<TreeData> TreeDataCollection = new List<TreeData>();
TreeDataCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent Task
1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
TreeDataCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child task
1", Progress = 80, Priority = "Low", Duration = 50, ParentId = 1 });
TreeDataCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child Task
2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
TreeDataCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child task
3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
TreeDataCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent Task
2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
TreeDataCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child task
1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child Task
2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child task
3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
}
}

```

```

TreeDataCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
return TreeDataCollection;
}
public static List<TreeData> GetSelfChangedDataSource()
{
List<TreeData> TreeDataCollection = new List<TreeData>();
TreeDataCollection.Add(new TreeData() { TaskId = 11, TaskName = "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
TreeDataCollection.Add(new TreeData() { TaskId = 22, TaskName = "Child task 1", Progress = 80, Priority = "Low", Duration = 50, ParentId = 11 });
TreeDataCollection.Add(new TreeData() { TaskId = 13, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 11 });
TreeDataCollection.Add(new TreeData() { TaskId = 14, TaskName = "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = null });
TreeDataCollection.Add(new TreeData() { TaskId = 15, TaskName = "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = 14 });
return TreeDataCollection;
}
}
}

```

Change data source dynamically

Task ID	Task Name	Duration	Progress	Priority
1	Parent Task 1	10	70	Critical
2	Child task 1	50	80	Low
3	Child Task 2	5	65	Critical
4	Child task 3	6	77	High
5	Parent Task 2	10	70	Critical
6	Child task 1	4	80	Critical
7	Child Task 2	5	65	Low
8	Child task 3	7	77	High
9	Child task 4	6	77	Low

1 of 1 pages (2 items)

Custom control in Tree Grid toolbar in Blazor TreeGrid Component

The custom controls can be rendered inside the Tree Grid's toolbar area. This can be achieved by initializing the custom controls within the Template property of the Toolbar component. This toolbar component is defined inside the Tree Grid component.

This is demonstrated in the below sample code where Autocomplete component is rendered inside the Tree Grid's toolbar and is used for performing search operation on the Tree Grid.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
@using Syncfusion.Blazor.DropDowns;
@using Syncfusion.Blazor.Navigations;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId"
TreeColumnIndex="1" AllowPaging="true">

```

```

<TreeGridPageSettings PageSize="8"></TreeGridPageSettings>
<SfToolbar>
<ToolbarItems>
<ToolbarItem Type="ItemType.Input">
<Template>
<SfAutoComplete Placeholder="Search Task Name" TItem="TaskDetails"
TValue="string" DataSource="@TaskNames">
<AutoCompleteEvents ValueChange="OnSearch" TValue="string"
TItem="TaskDetails"></AutoCompleteEvents>
<AutoCompleteFieldSettings Value="Name"></AutoCompleteFieldSettings>
</SfAutoComplete>
</Template>
</ToolbarItem>
</ToolbarItems>
</SfToolbar>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" IsPrimaryKey="true"
Width="70"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name" Width="85">
</TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority" Width="70">
</TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="70"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right">
</TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="70"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid<TreeData> TreeGrid;
public List<TreeData> TreeGridData { get; set; }
public class TaskDetails
{
public string Name { get; set; }
public int Id { get; set; }
}
List<TaskDetails> TaskNames = new List<TaskDetails>
{
new TaskDetails() { Name = "Parent Task 1", Id = 1 },
new TaskDetails() { Name = "Parent Task 2", Id = 2 },
new TaskDetails() { Name = "Child task 1", Id = 3 },
new TaskDetails() { Name = "Child Task 2", Id = 4 },
new TaskDetails() { Name = "Child task 3", Id = 5 },
new TaskDetails() { Name = "Child task 4", Id = 6 }
};
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
public void OnSearch(Syncfusion.Blazor.DropDowns.ChangeEventArgs<string,
TaskDetails> args)
{
this.TreeGrid.Search(args.Value);
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<TreeData> GetSelfDataSource()
{
List<TreeData> TreeDataCollection = new List<TreeData>();
TreeDataCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent Task
1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
TreeDataCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child task
1", Progress = 80, Priority = "Low", Duration = 50, ParentId = 1 });
TreeDataCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child Task
2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
TreeDataCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child task
3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
TreeDataCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent Task
2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
TreeDataCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child task
1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child Task
2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child task
3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child task
4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
return TreeDataCollection;
}
}
}

```

Search Task Name					
Task ID	Task Name	Priority	Duration	Progress	
1	Parent Task 1	Critical	10	70	
2	Child task 1	Low	50	80	
3	Child Task 2	Critical	5	65	
4	Child task 3	High	6	77	
5	Parent Task 2	Critical	10	70	
6	Child task 1	Critical	4	80	
7	Child Task 2	Low	5	65	
8	Child task 3	High	7	77	
9	Child task 4	Low	6	77	

1 of 1 pages (2 items)

Tree Grid customization in Blazor TreeGrid Component

It is possible to customize the default styles of the Tree Grid component. This can be achieved by adding class dynamically to the columns using the `AddClass` method of the [QueryCellInfo](#) event. Then the required styles are added to this class.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1"
AllowPaging="true">
<TreeGridEvents QueryCellInfo="QueryCellInfoHandler"
TValue="TreeData"></TreeGridEvents>
<TreeGridPageSettings PageSize="8"></TreeGridPageSettings>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" IsPrimaryKey="true"
Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name" Width="160">
</TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority" Width="80">
</TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right">
</TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
<style>
.e-treegrid .e-gridcontent .e-rowcell.above-10 {
color: green;
}
.e-treegrid .e-gridcontent .e-rowcell.above-5 {
color: blue;
}
.e-treegrid .e-gridcontent .e-rowcell.below-5 {
color: red;
}
</style>
@code{
SfTreeGrid<TreeData> TreeGrid;
public List<TreeData> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
public void QueryCellInfoHandler(QueryCellInfoEventArgs<TreeData> args)
{
if (args.Data.Duration > 10)
{
args.Cell.AddClass(new string[] { "above-10" });
}
else if (args.Data.Duration > 5 && args.Data.Duration <= 10)
```

```

{
args.Cell.AddClass(new string[] { "above-5" });
}
else
{
args.Cell.AddClass(new string[] { "below-5" });
}
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<TreeData> GetSelfDataSource()
{
List<TreeData> TreeDataCollection = new List<TreeData>();
TreeDataCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent Task
1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
TreeDataCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child task
1", Progress = 80, Priority = "Low", Duration = 50, ParentId = 1 });
TreeDataCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child Task
2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
TreeDataCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child task
3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
TreeDataCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent Task
2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
TreeDataCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child task
1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child Task
2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child task
3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child task
4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
return TreeDataCollection;
}
}
}

```

Task ID	Task Name	Priority	Duration	Progress
1	▼ Parent Task 1	Critical	10	70
2	▼ Child task 1	Low	50	80
3	▼ Child Task 2	Critical	5	65
4	Child task 3	High	6	77
5	▼ Parent Task 2	Critical	10	70
6	Child task 1	Critical	4	80
7	Child Task 2	Low	5	65
8	Child task 3	High	7	77
9	Child task 4	Low	6	77
<< < 1 of 1 pages > >>				

Prevent default Tree Grid action in Blazor TreeGrid Component

The default Tree Grid actions can be prevented by canceling them in the [OnActionBegin](#) event.

This is demonstrated in the below sample code where the **Add** operation is prevented by setting **Cancel** argument value of the [OnActionBegin](#) event to **false**.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId"
TreeColumnIndex="1" AllowPaging="true" Toolbar="@ (new List<string>() {
"Add", "Delete", "Edit", "Update", "Cancel" })">
<TreeGridEvents OnActionBegin="OnActionBegin"
TValue="TreeData"></TreeGridEvents>
<TreeGridEditSettings AllowAdding="true" AllowEditing="true"
AllowDeleting="true"></TreeGridEditSettings>
<TreeGridPageSettings PageSize="8"></TreeGridPageSettings>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" IsPrimaryKey="true"
Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name" Width="160">
</TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority" Width="80">
</TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right">
</TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
```

```

</TreeGridColumn>
</SfTreeGrid>
@code{
SfTreeGrid<TreeData> TreeGrid;
public List<TreeData> TreeGridData { get; set; }
protected override void OnInitialized()
{
    this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
public void OnActionBegin(ActionEventArgs<TreeData> args)
{
    if (args.RequestType == Syncfusion.Blazor.Grids.Action.Add)
    {
        args.Cancel = true;
    }
}
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
    public int TaskId { get; set; }
    public string TaskName { get; set; }
    public int? Duration { get; set; }
    public int? Progress { get; set; }
    public string Priority { get; set; }
    public int? ParentId { get; set; }
    public static List<TreeData> GetSelfDataSource()
    {
        List<TreeData> TreeDataCollection = new List<TreeData>();
        TreeDataCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
        TreeDataCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child task 1", Progress = 80, Priority = "Low", Duration = 50, ParentId = 1 });
        TreeDataCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
        TreeDataCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
        TreeDataCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
        TreeDataCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5 });
        TreeDataCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
        TreeDataCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
        TreeDataCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
        return TreeDataCollection;
    }
}
}
}

```


Get index value of selected row cell in Blazor TreeGrid Component

The index value of a selected row cell or row is got by using the [GetSelectedRowCellIndexes](#) method of the Tree Grid component.

This is demonstrated in the below sample code where the [GetSelectedRowCellIndexes](#) method is called on button click which returns the selected row cell indexes,

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.Buttons
@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfButton OnClick="SelectedRowCellIndex" CssClass="e-primary"
IsPrimary="true" Content="Get selected rowcell index"></SfButton>
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1">
<TreeGridSelectionSettings
Mode=SelectionMode.Cell></TreeGridSelectionSettings>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="160"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="80"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
SfTreeGrid<TreeData> TreeGrid;
public List<TreeData> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
public async Task SelectedRowCellIndex()
{
var value = await this.TreeGrid.GetSelectedRowCellIndexes();
}
}
```

C#

```
namespace TreeGridComponent.Data {
public class TreeData
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
}
```

```

public static List<TreeData> GetSelfDataSource()
{
    List<TreeData> TreeDataCollection = new List<TreeData>();
    TreeDataCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
    TreeDataCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child task 1", Progress = 80, Priority = "Low", Duration = 50, ParentId = 1 });
    TreeDataCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
    TreeDataCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
    TreeDataCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
    TreeDataCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5 });
    TreeDataCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
    TreeDataCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
    TreeDataCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
    return TreeDataCollection;
}
}
}

```

For getting the row cell index value, the [Mode](#) property of the [TreeGridSelectionSettings](#) component should be set as **Cell**.

Select rows based on certain condition in Blazor TreeGrid Component

Specific rows in the Tree Grid can be selected based on some conditions by using the [SelectRows](#) method in the [DataBound](#) event of the Tree Grid component.

This is demonstrated in the below sample code where the index value of Tree Grid rows with **Duration** column value greater than 6 are stored in the [RowDataBound](#) event and then selected in the [DataBound](#) event.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1">
    <TreeGridSelectionSettings
    Type=SelectionMode.Multiple></TreeGridSelectionSettings>
    <TreeGridEvents RowDataBound="OnRowDataBound" DataBound="OnDataBound"
    TValue="TreeData"></TreeGridEvents>
    <TreeGridPageSettings PageSize="8"></TreeGridPageSettings>
    <TreeGridColumn>
    <TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="70"
    TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
    <TreeGridColumn Field="TaskName" HeaderText="Task Name"
    Width="85"></TreeGridColumn>
    <TreeGridColumn Field="Priority" HeaderText="Priority"
    Width="60"></TreeGridColumn>

```

```

<TreeGridColumn Field="Duration" HeaderText="Duration" Width="60"
TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="60"
TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Resources" HeaderText="Resources" Width="70"
TextAlign="TextAlign.Right"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid<TreeData> TreeGrid;
public List<TreeData> TreeGridData { get; set; }
public List<int> SelectedNodeIndex = new List<int>();
protected override void OnInitialized()
{
    this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
public void OnDataBound(object args)
{
    // The filtered index values are selected
    this.TreeGrid.SelectRows(SelectedNodeIndex);
}
public void OnRowDataBound(RowDataBoundEventArgs<TreeData> args)
{
    // Freight values greater than 10 are filtered by comparing the primary
    column values
    if (args.Data.Duration > 6)
    {
        var dataSource = this.TreeGrid.DataSource;
        var index = 0;
        foreach (var data in dataSource)
        {
            if (data.TaskId == args.Data.TaskId)
            {
                SelectedNodeIndex.Add(index);
                break;
            }
            index++;
        }
    }
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
    public int TaskId { get; set; }
    public string TaskName { get; set; }
    public int? Duration { get; set; }
    public int? Progress { get; set; }
    public string Priority { get; set; }
    public int? ParentId { get; set; }
    public static List<TreeData> GetSelfDataSource()
    {
        List<TreeData> TreeDataCollection = new List<TreeData>();
    }
}
}

```

```

TreeDataCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
TreeDataCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child task 1", Progress = 80, Priority = "Low", Duration = 50, ParentId = 1 });
TreeDataCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
TreeDataCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
TreeDataCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
TreeDataCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
return TreeDataCollection;
}
}
}

```

Task ID	Task Name	Priority	Duration	Progress	Resources
1	▼ Parent Task 1	Critical	10	70	1
2	▼ Child task 1	Low	50	80	2
3	▼ Child Task 2	Critical	5	65	3
4	Child task 3	High	6	77	4
5	▼ Parent Task 2	Critical	10	70	5
6	Child task 1	Critical	4	80	6
7	Child Task 2	Low	5	65	7
8	Child task 3	High	7	77	8
9	Child task 4	Low	6	77	5

Customize column menu icon in Blazor TreeGrid Component

The column menu icon can be customized by overriding the default icon class `.e-icons.e-columnmenu` with the `content` property.

CSS

```

.e-grid .e-columnheader .e-icons.e-columnmenu::before {
content: "\e705";
}

```

This is demonstrated in the below sample code,

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid height="315" DataSource="@TreeData" IdMapping="TaskId"
ParentIdMapping="ParentId" AllowPaging="true" TreeColumnIndex="1"
ShowColumnMenu="true" AllowSorting="true">
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="100"
TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="145"></TreeGridColumn>
<TreeGridColumn Field="StartDate" HeaderText="Start Date" Format="d"
Type=ColumnType.Date Width="130"
TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="EndDate" HeaderText="End Date" Format="d"
Type=ColumnType.Date Width="130"
TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="120"
TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="110"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
SfTreeGrid<WrapData> TreeGrid;
public List<WrapData> TreeData { get; set; }
protected override void OnInitialized()
{
this.TreeData = WrapData.GetWrapData().ToList();
}
}
<style>
.e-grid .e-columnheader .e-icons.e-columnmenu::before {
content: "\e705";
}
</style>

```

C#

```

namespace TreeGridComponent.Data {
public class WrapData
{
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 string Progress { get; set; }
public string Priority { get; set; }
public bool Approved { get; set; }
public int Resources { get; set; }
public int? ParentId { get; set; }
public static List<WrapData> GetWrapData()
{
List<WrapData> BusinessObjectCollection = new List<WrapData>();
BusinessObjectCollection.Add(new WrapData()

```

```
{
    TaskId = 1,
    TaskName = "Planning",
    StartDate = new DateTime(2017, 03, 02),
    EndDate = new DateTime(2017, 07, 03),
    Progress = "Open",
    Duration = 5,
    Priority = "Normal",
    Resources = 6,
    Approved = false,
    ParentId = null
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 2,
    TaskName = "Plan timeline",
    StartDate = new DateTime(2017, 03, 04),
    EndDate = new DateTime(2017, 07, 05),
    Progress = "Inprogress",
    Duration = 5,
    Resources = 4,
    Priority = "Normal",
    Approved = false,
    ParentId = 1
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 3,
    TaskName = "Plan budget",
    StartDate = new DateTime(2017, 03, 06),
    EndDate = new DateTime(2017, 07, 07),
    Duration = 5,
    Progress = "Started",
    Approved = true,
    Resources = 6,
    Priority = "Low",
    ParentId = 1
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 4,
    TaskName = "Allocate resources",
    StartDate = new DateTime(2017, 03, 08),
    EndDate = new DateTime(2017, 07, 09),
    Duration = 5,
    Progress = "Open",
    Priority = "Critical",
    ParentId = 1,
    Resources = 3,
    Approved = false
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 5,
    TaskName = "Planning complete",
    StartDate = new DateTime(2017, 07, 10),
    EndDate = new DateTime(2017, 07, 11),
```

```
Duration = 1,
Progress = "Open",
Priority = "Low",
Resources = 5,
ParentId = 1,
Approved = true
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 6,
    TaskName = "Design",
    StartDate = new DateTime(2017, 10, 12),
    EndDate = new DateTime(2017, 02, 13),
    Progress = "Inprogress",
    Duration = 3,
    Priority = "High",
    Resources = 4,
    Approved = false,
    ParentId = null
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 7,
    TaskName = "Software Specification",
    StartDate = new DateTime(2017, 10, 14),
    EndDate = new DateTime(2017, 02, 15),
    Duration = 3,
    Progress = "Started",
    Resources = 3,
    Priority = "Normal",
    ParentId = 6,
    Approved = false
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 8,
    TaskName = "Develop prototype",
    StartDate = new DateTime(2017, 10, 16),
    EndDate = new DateTime(2017, 02, 17),
    Duration = 3,
    Progress = "Inprogress",
    Resources = 2,
    Priority = "Critical",
    ParentId = 6,
    Approved = false
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 9,
    TaskName = "Get approval from customer",
    StartDate = new DateTime(2017, 02, 18),
    EndDate = new DateTime(2017, 02, 19),
    Duration = 2,
    Progress = "Inprogress",
    Resources = 3,
    Priority = "Low",
    Approved = true,
```

```
ParentId = 6
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 10,
    TaskName = "Design complete",
    StartDate = new DateTime(2017, 02, 20),
    EndDate = new DateTime(2017, 02, 21),
    Duration = 1,
    Progress = "Inprogress",
    Resources = 6,
    Priority = "Normal",
    ParentId = 6,
    Approved = true
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 12,
    TaskName = "Implementation Phase",
    StartDate = new DateTime(2017, 02, 22),
    EndDate = new DateTime(2017, 02, 23),
    Priority = "Normal",
    Approved = false,
    Duration = 11,
    Resources = 5,
    Progress = "Started",
    ParentId = null
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 13,
    TaskName = "Phase 1",
    StartDate = new DateTime(2017, 02, 24),
    EndDate = new DateTime(2017, 02, 25),
    Priority = "High",
    Approved = false,
    Duration = 11,
    Progress = "Open",
    Resources = 4,
    ParentId = 12
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 14,
    TaskName = "Implementation Module 1",
    StartDate = new DateTime(2017, 02, 26),
    EndDate = new DateTime(2017, 02, 27),
    Priority = "Normal",
    Duration = 11,
    Progress = "Started",
    Resources = 3,
    Approved = false,
    ParentId = 13
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 15,
```



```

TaskName = "Development Task 1",
StartDate = new DateTime(2017, 06, 18),
EndDate = new DateTime(2017, 06, 19),
Duration = 3,
Progress = "Inprogress",
Priority = "High",
Resources = 2,
ParentId = 14,
Approved = false
});
BusinessObjectCollection.Add(new WrapData()
{
TaskId = 16,
TaskName = "Development Task 2",
StartDate = new DateTime(2017, 02, 13),
EndDate = new DateTime(2017, 03, 01),
Duration = 3,
Progress = "Closed",
Priority = "Low",
Resources = 5,
ParentId = 14,
Approved = true
});
BusinessObjectCollection.Add(new WrapData()
{
TaskId = 17,
TaskName = "Testing",
StartDate = new DateTime(2017, 03, 02),
EndDate = new DateTime(2017, 03, 03),
Duration = 2,
Progress = "Closed",
Priority = "Normal",
ParentId = 14,
Resources = 1,
Approved = true
});
BusinessObjectCollection.Add(new WrapData()
{
TaskId = 18,
TaskName = "Bug fix",
StartDate = new DateTime(2017, 03, 04),
EndDate = new DateTime(2017, 03, 05),
Duration = 2,
Progress = "Validated",
Priority = "Critical",
ParentId = 14,
Resources = 6,
Approved = false
});
BusinessObjectCollection.Add(new WrapData()
{
TaskId = 19,
TaskName = "Customer review meeting",
StartDate = new DateTime(2017, 03, 06),
EndDate = new DateTime(2017, 03, 07),
Duration = 2,
Progress = "Open",

```

```
Priority = "High",
ParentId = 14,
Resources = 6,
Approved = false
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 20,
    TaskName = "Phase 1 complete",
    StartDate = new DateTime(2017, 04, 27),
    EndDate = new DateTime(2017, 07, 27),
    Duration = 2,
    Progress = "Closed",
    Priority = "Low",
    ParentId = 14,
    Resources = 5,
    Approved = true
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 21,
    TaskName = "Phase 2",
    StartDate = new DateTime(2017, 07, 17),
    EndDate = new DateTime(2017, 09, 28),
    Priority = "High",
    Approved = false,
    Progress = "Open",
    ParentId = 12,
    Resources = 3,
    Duration = 12,
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 22,
    TaskName = "Implementation Module 2",
    StartDate = new DateTime(2017, 01, 17),
    EndDate = new DateTime(2017, 02, 28),
    Priority = "Critical",
    Approved = false,
    Progress = "Inprogress",
    ParentId = 21,
    Resources = 3,
    Duration = 12
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 23,
    TaskName = "Development Task 1",
    StartDate = new DateTime(2017, 08, 17),
    EndDate = new DateTime(2017, 09, 20),
    Duration = 4,
    Progress = "Closed",
    Priority = "Normal",
    ParentId = 22,
    Resources = 2,
    Approved = true,
});
```

```
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 24,
    TaskName = "Development Task 2",
    StartDate = new DateTime(2017, 04, 17),
    EndDate = new DateTime(2017, 03, 20),
    Duration = 4,
    Progress = "Closed",
    Priority = "Critical",
    ParentId = 22,
    Resources = 5,
    Approved = true,
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 25,
    TaskName = "Testing",
    StartDate = new DateTime(2017, 01, 21),
    EndDate = new DateTime(2017, 01, 24),
    Duration = 2,
    Progress = "Open",
    Priority = "High",
    ParentId = 22,
    Resources = 3,
    Approved = false,
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 26,
    TaskName = "Bug fix",
    StartDate = new DateTime(2017, 03, 25),
    EndDate = new DateTime(2017, 08, 26),
    Duration = 2,
    Progress = "Validated",
    Priority = "Low",
    Approved = false,
    Resources = 6,
    ParentId = 22
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 27,
    TaskName = "Customer review meeting",
    StartDate = new DateTime(2017, 07, 27),
    EndDate = new DateTime(2017, 06, 28),
    Duration = 2,
    Progress = "Inprogress",
    Priority = "Critical",
    ParentId = 22,
    Resources = 4,
    Approved = true,
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 28,
    TaskName = "Phase 2 complete",
    StartDate = new DateTime(2017, 07, 19),
```

```
EndDate = new DateTime(2017, 05, 28),
Duration = 2,
Priority = "Normal",
Progress = "Open",
ParentId = 22,
Resources = 3,
Approved = false,
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 29,
    TaskName = "Phase 3",
    StartDate = new DateTime(2017, 07, 17),
    EndDate = new DateTime(2017, 02, 12),
    Priority = "Normal",
    Approved = false,
    Duration = 11,
    Progress = "Inprogress",
    Resources = 4,
    ParentId = 12
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 30,
    TaskName = "Implementation Module 3",
    StartDate = new DateTime(2017, 08, 17),
    EndDate = new DateTime(2017, 09, 27),
    Priority = "High",
    Approved = false,
    Duration = 11,
    Resources = 5,
    Progress = "Validated",
    ParentId = 29,
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 31,
    TaskName = "Development Task 1",
    StartDate = new DateTime(2017, 11, 17),
    EndDate = new DateTime(2017, 12, 19),
    Duration = 3,
    Progress = "Closed",
    Priority = "Low",
    Approved = true,
    Resources = 3,
    ParentId = 30
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 32,
    TaskName = "Development Task 2",
    StartDate = new DateTime(2017, 12, 17),
    EndDate = new DateTime(2017, 02, 19),
    Duration = 3,
    Progress = "Closed",
    Priority = "Normal",
    Approved = false,
```

```
Resources = 2,
ParentId = 30
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 33,
    TaskName = "Testing",
    StartDate = new DateTime(2017, 01, 01),
    EndDate = new DateTime(2017, 07, 21),
    Duration = 2,
    Progress = "Closed",
    Priority = "Critical",
    ParentId = 30,
    Resources = 4,
    Approved = true,
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 34,
    TaskName = "Bug fix",
    StartDate = new DateTime(2017, 01, 24),
    EndDate = new DateTime(2017, 01, 25),
    Duration = 2,
    Progress = "Open",
    Priority = "High",
    Approved = false,
    Resources = 3,
    ParentId = 30
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 35,
    TaskName = "Customer review meeting",
    StartDate = new DateTime(2017, 12, 26),
    EndDate = new DateTime(2017, 12, 27),
    Duration = 2,
    Progress = "Inprogress",
    Priority = "Normal",
    ParentId = 30,
    Resources = 6,
    Approved = true,
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 36,
    TaskName = "Phase 3 complete",
    StartDate = new DateTime(2017, 05, 27),
    EndDate = new DateTime(2017, 05, 27),
    Duration = 2,
    Priority = "Critical",
    Progress = "Open",
    Resources = 5,
    ParentId = 30,
    Approved = false,
});
return BusinessObjectCollection;
}
```

```
}

```

Task ID	Task Name	Start Date	End Date	Duration	Priority
1	Autofit all columns	3/2/2017	7/3/2017	5	Normal
2	Autofit this column	3/4/2017	7/5/2017	5	Normal
3	Sort Ascending	3/6/2017	7/7/2017	5	Low
4	Sort Descending	3/8/2017	7/9/2017	6	Critical
5	Columns	7/10/2017	7/11/2017	1	Low
6	Planning complete	10/12/2017	2/13/2017	3	High
7	Design	10/14/2017	2/15/2017	3	Normal
8	Software Specification	10/16/2017	2/17/2017	3	Critical
9	Develop prototype	2/18/2017	2/19/2017	2	Low
	Get approval from cust...				

Calculate column value based on other columns in Blazor TreeGrid

The values for a Tree Grid column can be calculated based on other column values by using the **context** parameter in the [Template](#) property of the [TreeGridColumn](#) component. Inside the [Template](#), the column values can be accessed using the implicit named parameter **context** and then calculate the values for the new column as required.

This is demonstrated in the following sample code where the value for **Resources** column is calculated based on the values of **Duration** and **Progress** columns.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" AllowPaging="true" TreeColumnIndex="1"
AllowSorting="true">
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Width="70"
TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="85"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="60"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="60"
Format="C2" TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="60"
Format="C2" TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Resources" HeaderText="Resources" Width="70"
Format="C2" TextAlign="TextAlign.Right">
<Template>
@{
var value = (context as TreeData);
var finalValue = value.Duration + value.Progress;
<p>${@finalValue}</p>
}
</Template>

```

```

</TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
SfTreeGrid<TreeData> TreeGrid;
public List<TreeData> TreeGridData { get; set; }
protected override void OnInitialized()
{
    this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
    public int TaskId { get; set; }
    public string TaskName { get; set; }
    public int? Duration { get; set; }
    public int? Progress { get; set; }
    public string Priority { get; set; }
    public int? ParentId { get; set; }
    public static List<TreeData> GetSelfDataSource()
    {
        List<TreeData> TreeDataCollection = new List<TreeData>();
        TreeDataCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
        TreeDataCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child task 1", Progress = 80, Priority = "Low", Duration = 50, ParentId = 1 });
        TreeDataCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
        TreeDataCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
        TreeDataCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
        TreeDataCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5 });
        TreeDataCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
        TreeDataCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
        TreeDataCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
        return TreeDataCollection;
    }
}
}

```

Task ID	Task Name	Priority	Duration	Progress	Resources
1	▼ Parent Task 1	Critical	\$10.00	\$70.00	\$80
2	▼ Child task 1	Low	\$50.00	\$80.00	\$130
3	▼ Child Task 2	Critical	\$5.00	\$65.00	\$70
4	Child task 3	High	\$6.00	\$77.00	\$83
5	▼ Parent Task 2	Critical	\$10.00	\$70.00	\$80
6	Child task 1	Critical	\$4.00	\$80.00	\$84
7	Child Task 2	Low	\$5.00	\$65.00	\$70
8	Child task 3	High	\$7.00	\$77.00	\$84
9	Child task 4	Low	\$6.00	\$77.00	\$83

<< < 1 > >>

1 of 1 pages (2 items)

Using dictionary values as datasource in Blazor TreeGrid Component

The dictionary values can be assigned in the Tree Grid's data source by accessing them using **KeyValuePair** data type inside the [Template](#) property of the [TreeGridColumn](#) component

This is demonstrated in the below sample code where **Designation** is defined as Dictionary value and it is accessed inside the template property of the [TreeGridColumn](#) using **KeyValuePair** data type. The key value is compared with the **TaskId** column value and based on that the value is displayed.

ASPX-CS

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" AllowPaging="true"
TreeColumnIndex="1" >
  <TreeGridColumns>
    <TreeGridColumn Field="TaskId" HeaderText="Task ID" IsPrimaryKey="true"
Width="100" TextAlign="TextAlign.Right"></TreeGridColumn>
    <TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="145"></TreeGridColumn>
    <TreeGridColumn Field="Duration" HeaderText="Duation"
Width="90"></TreeGridColumn>
    <TreeGridColumn Field="Priority" HeaderText="Priority"
Width="90"></TreeGridColumn>
    <TreeGridColumn Field="Designation" HeaderText="Designation" Width="90">
      <Template>

```



```

@{
    var Details = context as TaskDetails;
    var level = Details.Designation.Select(kvp => (kvp.Key == Details.TaskId) ?
    kvp.Value.ToString() : "");
    <p>@string.Join("", level)</p>
}
</Template>
</TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
    SfTreeGrid<TaskDetails> TreeGrid;
    public class TaskDetails
    {
        public int TaskId { get; set; }
        public string TaskName { get; set; }
        public int Duration { get; set; }
        public int? Progress { get; set; }
        public string Priority { get; set; }
        public int? ParentId { get; set; }
        public Dictionary<int, string> Designation { get; set; }
    }
    List<TaskDetails> TreeGridData = new List<TaskDetails>
    {
        new TaskDetails { TaskId = 1, TaskName = "Parent Task 1", Duration = 10,
        Progress = 70, Priority = "Critical", ParentId = null, Designation =
        DesignationDetails},
        new TaskDetails { TaskId = 2, TaskName = "Child task 1",Duration = 50,
        Progress = 80, Priority = "Low", ParentId = 1, Designation =
        DesignationDetails},
        new TaskDetails { TaskId = 3, TaskName = "Child Task 2", Duration = 5,
        Progress = 65, Priority = "Critical", ParentId = 2, Designation =
        DesignationDetails},
        new TaskDetails { TaskId = 4, TaskName = "Child task 3", Duration = 6,
        Priority = "High", Progress = 77, ParentId = 3, Designation =
        DesignationDetails},
        new TaskDetails { TaskId = 5, TaskName = "Parent Task 2", Duration = 10,
        Progress = 70, Priority = "Critical", ParentId = null, Designation =
        DesignationDetails},
        new TaskDetails { TaskId = 6, TaskName = "Child task 1", Duration = 4,
        Progress = 80, Priority = "Critical", ParentId = 5, Designation =
        DesignationDetails},
        new TaskDetails { TaskId = 7, TaskName = "Child Task 2", Duration = 5,
        Progress = 65, Priority = "Low", ParentId = 5, Designation =
        DesignationDetails},
        new TaskDetails { TaskId = 8, TaskName = "Child task 3", Duration = 7,
        Progress = 77, Priority = "High", ParentId = 5, Designation =
        DesignationDetails},
        new TaskDetails { TaskId = 9, TaskName = "Child task 4", Duration = 6,
        Progress = 77, Priority = "Low", ParentId = 5, Designation =
        DesignationDetails},
    };
    private static Dictionary<int, string> DesignationDetails = new
    Dictionary<int, string>()
    {
        { 1, "Level1" },
        { 2, "Level2" },
    }
}

```

```
{ 3, "Level3" },
{ 4, "Level4" },
{ 5, "Level5" },
{ 6, "Level6" },
{ 7, "Level7" },
{ 8, "Level8" },
{ 9, "Level9" },
};
}
```

Task ID	Task Name	Duation	Priority	Designation
1	▼ Parent Task 1	10	Critical	Level1
2	▼ Child task 1	50	Low	Level2
3	▼ Child Task 2	5	Critical	Level3
4	Child task 3	6	High	Level4
5	▼ Parent Task 2	10	Critical	Level5
6	Child task 1	4	Critical	Level6
7	Child Task 2	5	Low	Level7
8	Child task 3	7	High	Level8
9	Child task 4	6	Low	Level9

<<
<
1
>
>>

1 of 1 pages (2 items)

Custom toolbar items in Blazor TreeGrid Component

The Custom toolbar items can be created with text name same as default toolbar items (Add,Edit,Delete,etc.). But while creating them, they will be considered as default toolbar items which will cause some issues while clicking on it. To overcome this behavior, it is suggested to define the **Id** property for custom toolbar items.

This is demonstrated in the below sample code where there are custom toolbar items with text same as **Add** and **Delete** buttons. These toolbar buttons will be enabled only when TreeGridEditSettings is defined in TreeGrid. So custom toolbar will be in disabled state considering it as default toolbar item. That behavior must be overcome by defining the Id property.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
@{
    List<Syncfusion.Blazor.Navigations.ItemModel> ToolbarItems = new
    List<Syncfusion.Blazor.Navigations.ItemModel>();
    ToolbarItems.Add(new Syncfusion.Blazor.Navigations.ItemModel() { Text =
    "Add", Id = "add", TooltipText = "Add Record", PrefixIcon = "add" });
    ToolbarItems.Add(new Syncfusion.Blazor.Navigations.ItemModel() { Text =
    "Delete", Id = "delete", TooltipText = "Delete Record", PrefixIcon =
    "delete" });
}
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" AllowPaging="true"
TreeColumnIndex="1" Toolbar="ToolbarItems">
<TreeGridEvents OnToolbarClick="ToolbarClickHandler"
TValue="TreeData"></TreeGridEvents>
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" IsPrimaryKey="true"
Width="70" TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="85"></TreeGridColumn>
<TreeGridColumn Field="Resources" HeaderText="Resource" Width="70"
TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duation" Width="70"
TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="70"
TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="70"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
@code{
    SfTreeGrid<TreeData> TreeGrid;
    public List<TreeData> TreeGridData { get; set; }
    protected override void OnInitialized()
    {
        this.TreeGridData = TreeData.GetSelfDataSource().ToList();
    }
    public void ToolbarClickHandler(Syncfusion.Blazor.Navigations.ClickEventArgs
args)
    {
        if (args.Item.Text == "Add")
        {
            //perform your actions here
        }
        if (args.Item.Text == "Delete")
        {
            //perform your actions here
        }
    }
}

```

C#

```
namespace TreeGridComponent.Data {
```

```
public class TreeData
{
    public int TaskId { get; set; }
    public string TaskName { get; set; }
    public int? Duration { get; set; }
    public int? Progress { get; set; }
    public string Priority { get; set; }
    public int? ParentId { get; set; }
    public static List<TreeData> GetSelfDataSource()
    {
        List<TreeData> TreeDataCollection = new List<TreeData>();
        TreeDataCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
        TreeDataCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child task 1", Progress = 80, Priority = "Low", Duration = 50, ParentId = 1 });
        TreeDataCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
        TreeDataCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
        TreeDataCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
        TreeDataCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5 });
        TreeDataCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
        TreeDataCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
        TreeDataCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
        return TreeDataCollection;
    }
}
```

Add		Delete				
Task ID	Task Name	Resource	Duation	Progress	Priority	
1	▼ Parent Task 1	1	10	70	Critical	
2	▼ Child task 1	2	50	80	Low	
3	▼ Child Task 2	3	5	65	Critical	
4	Child task 3	4	6	77	High	
5	▼ Parent Task 2	5	10	70	Critical	
6	Child task 1	6	4	80	Critical	
7	Child Task 2	7	5	65	Low	
8	Child task 3	8	7	77	High	
9	Child task 4	5	6	77	Low	

<< < 1 > >>
 1 of 1 pages (2 items)

Render Blazor TreeGrid Component inside the Tab with specific height

By default, Tree Grid will occupy the entire space of the parent element when the Tree Grid [Height](#) and [Width](#) property is defined as 100%. But if the similar Tree Grid is rendered inside the Tab control, it will consider the entire page and render the Tree Grid without horizontal scroller.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
@using Syncfusion.Blazor.Navigations
<div style="height:300px">
<SfTab ID="Ej2Tab" Width="100%">
<TabItems>
<TabItem>
<ChildContent>
<TabHeader Text="Tree Grid 1"></TabHeader>
</ChildContent>
<ContentTemplate>
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" AllowPaging="true"
TreeColumnIndex="1" Height="100%" Width="100%">
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" IsPrimaryKey="true"
Width="70" TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="85"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="60"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="70"
Format="C2" TextAlign="TextAlign.Right"></TreeGridColumn>

```

```

<TreeGridColumn Field="Resources" HeaderText="Resource" Width="70"
Format="C2" TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="70"
Format="C2" TextAlign="TextAlign.Right"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
</ContentTemplate>
</TabItem>
<TabItem>
<ChildContent>
<TabHeader Text="Tree Grid 2"></TabHeader>
</ChildContent>
<ContentTemplate>
<SfTreeGrid DataSource="@TreeDataSource" IdMapping="TaskId"
ParentIdMapping="ParentId" AllowPaging="true"
TreeColumnIndex="1" Height="100%" Width="100%">
<TreeGridColumns>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" Visible="false"
Width="100" TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="120"
Format="C2" TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="120"
Format="C2" TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Resources" HeaderText="Resources" Width="120"
Format="C2" TextAlign="TextAlign.Right"></TreeGridColumn>
</TreeGridColumns>
</SfTreeGrid>
</ContentTemplate>
</TabItem>
</TabItems>
</SfTab>
</div>
@code{
SfTreeGrid<TreeData> TreeGrid;
public List<TreeData> TreeGridData { get; set; }
public List<WrapData> TreeDataSource { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
this.TreeDataSource = WrapData.GetWrapData().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public int? Duration { get; set; }
public int? Progress { get; set; }
public string Priority { get; set; }
public int? ParentId { get; set; }
public static List<TreeData> GetSelfDataSource()
{

```

```

List<TreeData> TreeDataCollection = new List<TreeData>();
TreeDataCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent Task
1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
TreeDataCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child task
1", Progress = 80, Priority = "Low", Duration = 50, ParentId = 1 });
TreeDataCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child Task
2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
TreeDataCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child task
3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
TreeDataCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent Task
2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
TreeDataCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child task
1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child Task
2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child task
3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
TreeDataCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child task
4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
return TreeDataCollection;
}
}

public class WrapData
{
    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 string Progress { get; set; }
    public string Priority { get; set; }
    public bool Approved { get; set; }
    public int Resources { get; set; }
    public int? ParentId { get; set; }
    public static List<WrapData> GetWrapData()
    {
        List<WrapData> BusinessObjectCollection = new List<WrapData>();
        BusinessObjectCollection.Add(new WrapData()
        {
            TaskId = 1,
            TaskName = "Planning",
            StartDate = new DateTime(2017, 03, 02),
            EndDate = new DateTime(2017, 07, 03),
            Progress = "Open",
            Duration = 5,
            Priority = "Normal",
            Resources = 6,
            Approved = false,
            ParentId = null
        });
        BusinessObjectCollection.Add(new WrapData()
        {
            TaskId = 2,
            TaskName = "Plan timeline",
            StartDate = new DateTime(2017, 03, 04),
            EndDate = new DateTime(2017, 07, 05),
            Progress = "Inprogress",

```

```
Duration = 5,
Resources = 4,
Priority = "Normal",
Approved = false,
ParentId = 1
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 3,
    TaskName = "Plan budget",
    StartDate = new DateTime(2017, 03, 06),
    EndDate = new DateTime(2017, 07, 07),
    Duration = 5,
    Progress = "Started",
    Approved = true,
    Resources = 6,
    Priority = "Low",
    ParentId = 1
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 4,
    TaskName = "Allocate resources",
    StartDate = new DateTime(2017, 03, 08),
    EndDate = new DateTime(2017, 07, 09),
    Duration = 5,
    Progress = "Open",
    Priority = "Critical",
    ParentId = 1,
    Resources = 3,
    Approved = false
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 5,
    TaskName = "Planning complete",
    StartDate = new DateTime(2017, 07, 10),
    EndDate = new DateTime(2017, 07, 11),
    Duration = 1,
    Progress = "Open",
    Priority = "Low",
    Resources = 5,
    ParentId = 1,
    Approved = true
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 6,
    TaskName = "Design",
    StartDate = new DateTime(2017, 10, 12),
    EndDate = new DateTime(2017, 02, 13),
    Progress = "Inprogress",
    Duration = 3,
    Priority = "High",
    Resources = 4,
    Approved = false,
    ParentId = null
});
```



```
});  
BusinessObjectCollection.Add(new WrapData()  
{  
    TaskId = 7,  
    TaskName = "Software Specification",  
    StartDate = new DateTime(2017, 10, 14),  
    EndDate = new DateTime(2017, 02, 15),  
    Duration = 3,  
    Progress = "Started",  
    Resources = 3,  
    Priority = "Normal",  
    ParentId = 6,  
    Approved = false  
});  
BusinessObjectCollection.Add(new WrapData()  
{  
    TaskId = 8,  
    TaskName = "Develop prototype",  
    StartDate = new DateTime(2017, 10, 16),  
    EndDate = new DateTime(2017, 02, 17),  
    Duration = 3,  
    Progress = "Inprogress",  
    Resources = 2,  
    Priority = "Critical",  
    ParentId = 6,  
    Approved = false  
});  
BusinessObjectCollection.Add(new WrapData()  
{  
    TaskId = 9,  
    TaskName = "Get approval from customer",  
    StartDate = new DateTime(2017, 02, 18),  
    EndDate = new DateTime(2017, 02, 19),  
    Duration = 2,  
    Progress = "Inprogress",  
    Resources = 3,  
    Priority = "Low",  
    Approved = true,  
    ParentId = 6  
});  
BusinessObjectCollection.Add(new WrapData()  
{  
    TaskId = 10,  
    TaskName = "Design complete",  
    StartDate = new DateTime(2017, 02, 20),  
    EndDate = new DateTime(2017, 02, 21),  
    Duration = 1,  
    Progress = "Inprogress",  
    Resources = 6,  
    Priority = "Normal",  
    ParentId = 6,  
    Approved = true  
});  
BusinessObjectCollection.Add(new WrapData()  
{  
    TaskId = 12,  
    TaskName = "Implementation Phase",
```

```
StartDate = new DateTime(2017, 02, 22),
EndDate = new DateTime(2017, 02, 23),
Priority = "Normal",
Approved = false,
Duration = 11,
Resources = 5,
Progress = "Started",
ParentId = null
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 13,
    TaskName = "Phase 1",
    StartDate = new DateTime(2017, 02, 24),
    EndDate = new DateTime(2017, 02, 25),
    Priority = "High",
    Approved = false,
    Duration = 11,
    Progress = "Open",
    Resources = 4,
    ParentId = 12
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 14,
    TaskName = "Implementation Module 1",
    StartDate = new DateTime(2017, 02, 26),
    EndDate = new DateTime(2017, 02, 27),
    Priority = "Normal",
    Duration = 11,
    Progress = "Started",
    Resources = 3,
    Approved = false,
    ParentId = 13
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 15,
    TaskName = "Development Task 1",
    StartDate = new DateTime(2017, 06, 18),
    EndDate = new DateTime(2017, 06, 19),
    Duration = 3,
    Progress = "Inprogress",
    Priority = "High",
    Resources = 2,
    ParentId = 14,
    Approved = false
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 16,
    TaskName = "Development Task 2",
    StartDate = new DateTime(2017, 02, 13),
    EndDate = new DateTime(2017, 03, 01),
    Duration = 3,
    Progress = "Closed",
    Priority = "Low",
```

```
Resources = 5,
ParentId = 14,
Approved = true
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 17,
    TaskName = "Testing",
    StartDate = new DateTime(2017, 03, 02),
    EndDate = new DateTime(2017, 03, 03),
    Duration = 2,
    Progress = "Closed",
    Priority = "Normal",
    ParentId = 14,
    Resources = 1,
    Approved = true
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 18,
    TaskName = "Bug fix",
    StartDate = new DateTime(2017, 03, 04),
    EndDate = new DateTime(2017, 03, 05),
    Duration = 2,
    Progress = "Validated",
    Priority = "Critical",
    ParentId = 14,
    Resources = 6,
    Approved = false
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 19,
    TaskName = "Customer review meeting",
    StartDate = new DateTime(2017, 03, 06),
    EndDate = new DateTime(2017, 03, 07),
    Duration = 2,
    Progress = "Open",
    Priority = "High",
    ParentId = 14,
    Resources = 6,
    Approved = false
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 20,
    TaskName = "Phase 1 complete",
    StartDate = new DateTime(2017, 04, 27),
    EndDate = new DateTime(2017, 07, 27),
    Duration = 2,
    Progress = "Closed",
    Priority = "Low",
    ParentId = 14,
    Resources = 5,
    Approved = true
});
BusinessObjectCollection.Add(new WrapData()
```

```
{
    TaskId = 21,
    TaskName = "Phase 2",
    StartDate = new DateTime(2017, 07, 17),
    EndDate = new DateTime(2017, 09, 28),
    Priority = "High",
    Approved = false,
    Progress = "Open",
    ParentId = 12,
    Resources = 3,
    Duration = 12,
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 22,
    TaskName = "Implementation Module 2",
    StartDate = new DateTime(2017, 01, 17),
    EndDate = new DateTime(2017, 02, 28),
    Priority = "Critical",
    Approved = false,
    Progress = "Inprogress",
    ParentId = 21,
    Resources = 3,
    Duration = 12
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 23,
    TaskName = "Development Task 1",
    StartDate = new DateTime(2017, 08, 17),
    EndDate = new DateTime(2017, 09, 20),
    Duration = 4,
    Progress = "Closed",
    Priority = "Normal",
    ParentId = 22,
    Resources = 2,
    Approved = true,
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 24,
    TaskName = "Development Task 2",
    StartDate = new DateTime(2017, 04, 17),
    EndDate = new DateTime(2017, 03, 20),
    Duration = 4,
    Progress = "Closed",
    Priority = "Critical",
    ParentId = 22,
    Resources = 5,
    Approved = true,
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 25,
    TaskName = "Testing",
    StartDate = new DateTime(2017, 01, 21),
    EndDate = new DateTime(2017, 01, 24),
```

```
Duration = 2,
Progress = "Open",
Priority = "High",
ParentId = 22,
Resources = 3,
Approved = false,
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 26,
    TaskName = "Bug fix",
    StartDate = new DateTime(2017, 03, 25),
    EndDate = new DateTime(2017, 08, 26),
    Duration = 2,
    Progress = "Validated",
    Priority = "Low",
    Approved = false,
    Resources = 6,
    ParentId = 22
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 27,
    TaskName = "Customer review meeting",
    StartDate = new DateTime(2017, 07, 27),
    EndDate = new DateTime(2017, 06, 28),
    Duration = 2,
    Progress = "Inprogress",
    Priority = "Critical",
    ParentId = 22,
    Resources = 4,
    Approved = true,
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 28,
    TaskName = "Phase 2 complete",
    StartDate = new DateTime(2017, 07, 19),
    EndDate = new DateTime(2017, 05, 28),
    Duration = 2,
    Priority = "Normal",
    Progress = "Open",
    ParentId = 22,
    Resources = 3,
    Approved = false,
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 29,
    TaskName = "Phase 3",
    StartDate = new DateTime(2017, 07, 17),
    EndDate = new DateTime(2017, 02, 12),
    Priority = "Normal",
    Approved = false,
    Duration = 11,
    Progress = "Inprogress",
    Resources = 4,
```

```
ParentId = 12
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 30,
    TaskName = "Implementation Module 3",
    StartDate = new DateTime(2017, 08, 17),
    EndDate = new DateTime(2017, 09, 27),
    Priority = "High",
    Approved = false,
    Duration = 11,
    Resources = 5,
    Progress = "Validated",
    ParentId = 29,
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 31,
    TaskName = "Development Task 1",
    StartDate = new DateTime(2017, 11, 17),
    EndDate = new DateTime(2017, 12, 19),
    Duration = 3,
    Progress = "Closed",
    Priority = "Low",
    Approved = true,
    Resources = 3,
    ParentId = 30
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 32,
    TaskName = "Development Task 2",
    StartDate = new DateTime(2017, 12, 17),
    EndDate = new DateTime(2017, 02, 19),
    Duration = 3,
    Progress = "Closed",
    Priority = "Normal",
    Approved = false,
    Resources = 2,
    ParentId = 30
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 33,
    TaskName = "Testing",
    StartDate = new DateTime(2017, 01, 01),
    EndDate = new DateTime(2017, 07, 21),
    Duration = 2,
    Progress = "Closed",
    Priority = "Critical",
    ParentId = 30,
    Resources = 4,
    Approved = true,
});
BusinessObjectCollection.Add(new WrapData()
{
    TaskId = 34,
```

```
TaskName = "Bug fix",
StartDate = new DateTime(2017, 01, 24),
EndDate = new DateTime(2017, 01, 25),
Duration = 2,
Progress = "Open",
Priority = "High",
Approved = false,
Resources = 3,
ParentId = 30
});
BusinessObjectCollection.Add(new WrapData()
{
TaskId = 35,
TaskName = "Customer review meeting",
StartDate = new DateTime(2017, 12, 26),
EndDate = new DateTime(2017, 12, 27),
Duration = 2,
Progress = "Inprogress",
Priority = "Normal",
ParentId = 30,
Resources = 6,
Approved = true,
});
BusinessObjectCollection.Add(new WrapData()
{
TaskId = 36,
TaskName = "Phase 3 complete",
StartDate = new DateTime(2017, 05, 27),
EndDate = new DateTime(2017, 05, 27),
Duration = 2,
Priority = "Critical",
Progress = "Open",
Resources = 5,
ParentId = 30,
Approved = false,
});
return BusinessObjectCollection;
}
}
}
```

Tree Grid 1		Tree Grid 2				
Task ID	Task Name	Priority	Duration	Resource	Progress	
1	▼ Parent Task 1	Critical	\$10.00	\$1.00	\$70.00	
2	▼ Child task 1	Low	\$50.00	\$2.00	\$80.00	
3	▼ Child Task 2	Critical	\$5.00	\$3.00	\$65.00	
4	Child task 3	High	\$6.00	\$4.00	\$77.00	
5	▼ Parent Task 2	Critical	\$10.00	\$5.00	\$70.00	
6	Child task 1	Critical	\$4.00	\$6.00	\$80.00	
7	Child Task 2	Low	\$5.00	\$7.00	\$65.00	
8	Child task 3	High	\$7.00	\$8.00	\$77.00	
9	Child task 4	Low	\$6.00	\$5.00	\$77.00	

1 of 1 pages (2 items)

Single click editing with Batch mode in Blazor TreeGrid Component

A cell is made editable on a single click with a [Batch](#) mode of editing in TreeGrid, by using the [EditCell](#) method.

Set the [Mode](#) property of **TreeGridSelectionSettings** component to **Both** and bind the [CellSelected](#) event to Tree Grid. In the [CellSelected](#) event handler, call the [EditCell](#) method based on the clicked cell.

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" AllowPaging="true" TreeColumnIndex="1"
Toolbar="@ (new List<string>() { "Cancel", "Update" }) ">
<TreeGridEditSettings AllowEditing="true"
Mode="Syncfusion.Blazor.TreeGrid.EditMode.Batch"></TreeGridEditSettings>
<TreeGridSelectionSettings
Mode="Syncfusion.Blazor.Grids.SelectionMode.Both"></TreeGridSelectionSettings>
<TreeGridEvents CellSelected="CellSelectHandler"
TValue="TreeData"></TreeGridEvents>
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" IsPrimaryKey="true"
Width="70" TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name"
Width="85"></TreeGridColumn>
<TreeGridColumn Field="Resources" HeaderText="Resource" Width="70"
TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="70"
TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="70"
TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="70"></TreeGridColumn>
```



```

</TreeGridColumn>
</SfTreeGrid>
@code{
SfTreeGrid<TreeData> TreeGrid;
public List<TreeData> TreeGridData { get; set; }
protected override void OnInitialized()
{
    this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
public async Task CellSelectHandler(CellSelectEventArgs<TreeData> args)
{
    //get selected cell index
    var CellIndexes = await TreeGrid.GetSelectedRowCellIndexes();
    //get the row and cell index
    var CurrentEditRowIndex = CellIndexes[0].Item1;
    var CurrentEditCellIndex = (int)CellIndexes[0].Item2;
    //get the available fields
    var fields = await TreeGrid.GetColumnFieldNames();
    // edit the selected cell using the cell index and column name
    await TreeGrid.EditCell(CurrentEditRowIndex, fields[CurrentEditCellIndex]);
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
    public int TaskId { get; set; }
    public string TaskName { get; set; }
    public int? Duration { get; set; }
    public int? Progress { get; set; }
    public string Priority { get; set; }
    public int? ParentId { get; set; }
    public static List<TreeData> GetSelfDataSource()
    {
        List<TreeData> TreeDataCollection = new List<TreeData>();
        TreeDataCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
        TreeDataCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child task 1", Progress = 80, Priority = "Low", Duration = 50, ParentId = 1 });
        TreeDataCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
        TreeDataCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
        TreeDataCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
        TreeDataCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5 });
        TreeDataCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
        TreeDataCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
        TreeDataCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
        return TreeDataCollection;
    }
}

```

```

}
}
}

```

The following GIF represents the single click edit performed on the Tree Grid with Edit Mode as "Batch",

Task ID	Task Name	Resource	Duration	Progress	Priority
1	Parent Task 1	1	10	70	Critical
2	Child task 1	2	50	80	Low
3	Child Task 2	3	5	65	Critical
4	Child task 3	4	6	77	High
5	Parent Task 2	5	10	70	Critical
6	Child task 1	6	4	80	Critical
7	Child Task 2	7	5	65	Low
8	Child task 3	8	7	77	High
9	Child task 4		6	77	Low

Editing with template column in Blazor TreeGrid Component

A template column value can be edited by defining the [Field](#) property for that particular [TreeGridColumn](#) component.

C#

```

@using TreeGridComponent.Data;
@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
<SfTreeGrid DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" AllowPaging="true"
TreeColumnIndex="1" Toolbar="@ (new List<string>() { "Add", "Edit", "Delete",
"Cancel", "Update" })">
<TreeGridEditSettings AllowEditing="true" AllowAdding="true"
AllowDeleting="true" Mode="Syncfusion.Blazor.TreeGrid.EditMode.Row" />
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" IsPrimaryKey="true"
Width="70" TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name" Width="75">
<Template>
@{
var data = context as TreeData;
<a href="#">@data.TaskName</a>
}
</Template>
</TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority"
Width="60"></TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="70"
Format="C2" TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Resource" HeaderText="Resource" Width="70"
Format="C2" TextAlign="TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="70"
Format="C2" TextAlign="TextAlign.Right"></TreeGridColumn>

```

```

</TreeGridColumn>
</SfTreeGrid>
@code{
SfTreeGrid<TreeData> TreeGrid;
public List<TreeData> TreeGridData { get; set; }
protected override void OnInitialized()
{
    this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}

```

C#

```

namespace TreeGridComponent.Data {
public class TreeData
{
    public int TaskId { get; set; }
    public string TaskName { get; set; }
    public int? Duration { get; set; }
    public int? Progress { get; set; }
    public string Priority { get; set; }
    public int? ParentId { get; set; }
    public static List<TreeData> GetSelfDataSource()
    {
        List<TreeData> TreeDataCollection = new List<TreeData>();
        TreeDataCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
        TreeDataCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child task 1", Progress = 80, Priority = "Low", Duration = 50, ParentId = 1 });
        TreeDataCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
        TreeDataCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
        TreeDataCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
        TreeDataCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5 });
        TreeDataCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
        TreeDataCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
        TreeDataCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
        return TreeDataCollection;
    }
}
}

```

+ Add ✎ Edit 🗑 Delete ✕ Cancel 🔄 Update						
Task ID	Task Name	Priority	Duration	Resource	Progress	
1	▼ Parent Task 1	Critical	\$10.00	\$1.00	\$70.00	
2	▼ Child task 1	Low	\$50.00	\$2.00	\$80.00	
3	▼ Child Task 2	Critical	\$5.00	\$3.00	\$65.00	
4	Child task 3	High	6	4	77	
5	▼ Parent Task 2	Critical	\$10.00	\$5.00	\$70.00	
6	Child task 1	Critical	\$4.00	\$6.00	\$80.00	
7	Child Task 2	Low	\$5.00	\$7.00	\$65.00	
8	Child task 3	High	\$7.00	\$8.00	\$77.00	
9	Child task 4	Low	\$6.00	\$5.00	\$77.00	

1 of 1 pages (2 items)

Hide Tree Grid Header in Blazor TreeGrid Component

The Tree Grid header can be hidden by applying the below styles to Tree Grid component.

HTML

```
<style>
.e-treegrid .e-gridheader .e-columnheader {
display: none;
}
</style>
```

If you want to hide the header for particular Tree Grid, then apply the above styles to that Tree Grid using the ID (#TreeGrid.e-treegrid .e-gridheader .e-columnheader) property value.

1	▼ Parent Task 1	Critical	\$10.00	\$70.00	\$1.00
2	▼ Child task 1	Low	\$50.00	\$80.00	\$2.00
3	▼ Child Task 2	Critical	\$5.00	\$65.00	\$3.00
4	Child task 3	High	\$6.00	\$77.00	\$4.00
5	▼ Parent Task 2	Critical	\$10.00	\$70.00	\$5.00
6	Child task 1	Critical	\$4.00	\$80.00	\$6.00
7	Child Task 2	Low	\$5.00	\$65.00	\$7.00
8	Child task 3	High	\$7.00	\$77.00	\$8.00
9	Child task 4	Low	\$6.00	\$77.00	

1 of 1 pages (2 items)

Display Custom Tooltip in Tree Grid cell in Blazor TreeGrid Component

The custom tooltip in the Tree Grid column can be displayed using the [Column Template](#) feature by rendering the [SfTooltip](#) components inside the template.

This is demonstrated in the below sample code where the tooltip is rendered for **TaskName** column using [Column Template](#).

C#

```
@using TreeGridComponent.Data;
@using Syncfusion.Blazor.Grids;
@using Syncfusion.Blazor.TreeGrid;
@using Syncfusion.Blazor.Popups;
<SfTreeGrid @ref="TreeGrid" DataSource="@TreeGridData" IdMapping="TaskId"
ParentIdMapping="ParentId" TreeColumnIndex="1"
AllowPaging="true" >
<TreeGridColumn>
<TreeGridColumn Field="TaskId" HeaderText="Task ID" IsPrimaryKey="true"
Width="80"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
<TreeGridColumn Field="TaskName" HeaderText="Task Name" Width="160">
<Template>
@{
var taskData = (context as TreeData);
<SfTooltip Target="#txt">
<TooltipTemplates>
<Content>
@taskData.TaskName
</Content>
</TooltipTemplates>
<span id="txt">@taskData.TaskName</span>
</SfTooltip>
}
</Template>
</TreeGridColumn>
<TreeGridColumn Field="Priority" HeaderText="Priority" Width="80" >
</TreeGridColumn>
<TreeGridColumn Field="Duration" HeaderText="Duration" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right">
</TreeGridColumn>
<TreeGridColumn Field="Progress" HeaderText="Progress" Width="100"
TextAlign="Syncfusion.Blazor.Grids.TextAlign.Right"></TreeGridColumn>
</TreeGridColumn>
</SfTreeGrid>
@code{
SfTreeGrid<TreeData> TreeGrid;
public List<TreeData> TreeGridData { get; set; }
protected override void OnInitialized()
{
this.TreeGridData = TreeData.GetSelfDataSource().ToList();
}
}
```

C#

```
namespace TreeGridComponent.Data {
```

```
public class TreeData
{
    public int TaskId { get; set; }
    public string TaskName { get; set; }
    public int? Duration { get; set; }
    public int? Progress { get; set; }
    public string Priority { get; set; }
    public int? ParentId { get; set; }
    public static List<TreeData> GetSelfDataSource()
    {
        List<TreeData> TreeDataCollection = new List<TreeData>();
        TreeDataCollection.Add(new TreeData() { TaskId = 1, TaskName = "Parent Task 1", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
        TreeDataCollection.Add(new TreeData() { TaskId = 2, TaskName = "Child task 1", Progress = 80, Priority = "Low", Duration = 50, ParentId = 1 });
        TreeDataCollection.Add(new TreeData() { TaskId = 3, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Critical", ParentId = 2 });
        TreeDataCollection.Add(new TreeData() { TaskId = 4, TaskName = "Child task 3", Duration = 6, Priority = "High", Progress = 77, ParentId = 3 });
        TreeDataCollection.Add(new TreeData() { TaskId = 5, TaskName = "Parent Task 2", Duration = 10, Progress = 70, Priority = "Critical", ParentId = null });
        TreeDataCollection.Add(new TreeData() { TaskId = 6, TaskName = "Child task 1", Duration = 4, Progress = 80, Priority = "Critical", ParentId = 5 });
        TreeDataCollection.Add(new TreeData() { TaskId = 7, TaskName = "Child Task 2", Duration = 5, Progress = 65, Priority = "Low", ParentId = 5 });
        TreeDataCollection.Add(new TreeData() { TaskId = 8, TaskName = "Child task 3", Duration = 6, Progress = 77, Priority = "High", ParentId = 5 });
        TreeDataCollection.Add(new TreeData() { TaskId = 9, TaskName = "Child task 4", Duration = 6, Progress = 77, Priority = "Low", ParentId = 5 });
        return TreeDataCollection;
    }
}
```

Task ID	Task Name	Priority	Duration	Progress
1	▼ Parent Task 1	Critical	10	70
2	▼ Child task 1	Low	50	80
3	▼ Child Task 2	Critical	5	65
4	Child task 3	High	6	77
5	▼ P Child task 3	Critical	10	70
6	Child task 1	Critical	4	80
7	Child Task 2	Low	5	65
8	Child task 3	High	7	77
9	Child task 4	Low	6	77
<< < 1 of 1 pages > >>				

TreeMap

Blazor TreeMap Component in Server Side App using Visual Studio

This section briefly explains how to include a TreeMap component in the Blazor server-side application. Refer to [Getting Started with Syncfusion Blazor for server-side in Visual Studio](#) page for introduction and configuring common specifications.

Importing Syncfusion Blazor TreeMap component in the application

1. Install **Syncfusion.Blazor.TreeMap** NuGet package in the application using the **NuGet Package Manager**.
2. Add the client-side resources through [CDN](#) or from [NuGet](#) package in the `element` of the `~/Pages/_Host.cshtml` page.

HTML

```
<head>
<link href="_content/Syncfusion.Blazor.Themes/bootstrap4.css"
rel="stylesheet" />
<!--CDN-->
@*<link href="https://cdn.syncfusion.com/blazor/{{ site.blazorversion
}}/styles/bootstrap4.css" rel="stylesheet" />*@
</head>
```

For Internet Explorer 11, kindly refer to the polyfills. Refer the [documentation](#) for more information.

HTML

```
<head>
```

```
<link href="https://cdn.syncfusion.com/blazor/{{ site.blazorversion
}}/styles/bootstrap4.css" rel="stylesheet" />
<script
src="https://github.com/Daddoon/Blazor.Polyfill/releases/download/3.0.1/blaz
or.polyfill.min.js"></script>
</head>
```

Adding component package to the application

Open the `~/_Imports.razor` file and include the **Syncfusion.Blazor.TreeMap** namespace.

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
```

Adding SyncfusionBlazor Service in Startup.cs

Open the **Startup.cs** file and add services required by Syncfusion components using **services.AddSyncfusionBlazor()** method. Add this method in the **ConfigureServices** function as follows.

CSHARP

```
using Syncfusion.Blazor;
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void ConfigureServices(IServiceCollection services)
        {
            ....
            ....
            services.AddSyncfusionBlazor();
        }
    }
}
```

To enable custom client-side source loading from CRG or CDN, please refer to the section about [custom resources in Blazor application](#).

Adding TreeMap component

To initialize the TreeMap component, add the following code to the **Index.razor** view page under **~/Pages** folder. In a new application, if **Index.razor** page has any default content template, then those content can be completely removed and the following code can be added.

Use the [DataSource](#) property to load the car sales details in the TreeMap component. Specify a field name from the data source in the [WeightValuePath](#) property to calculate the TreeMap item size.

ASPX-CS

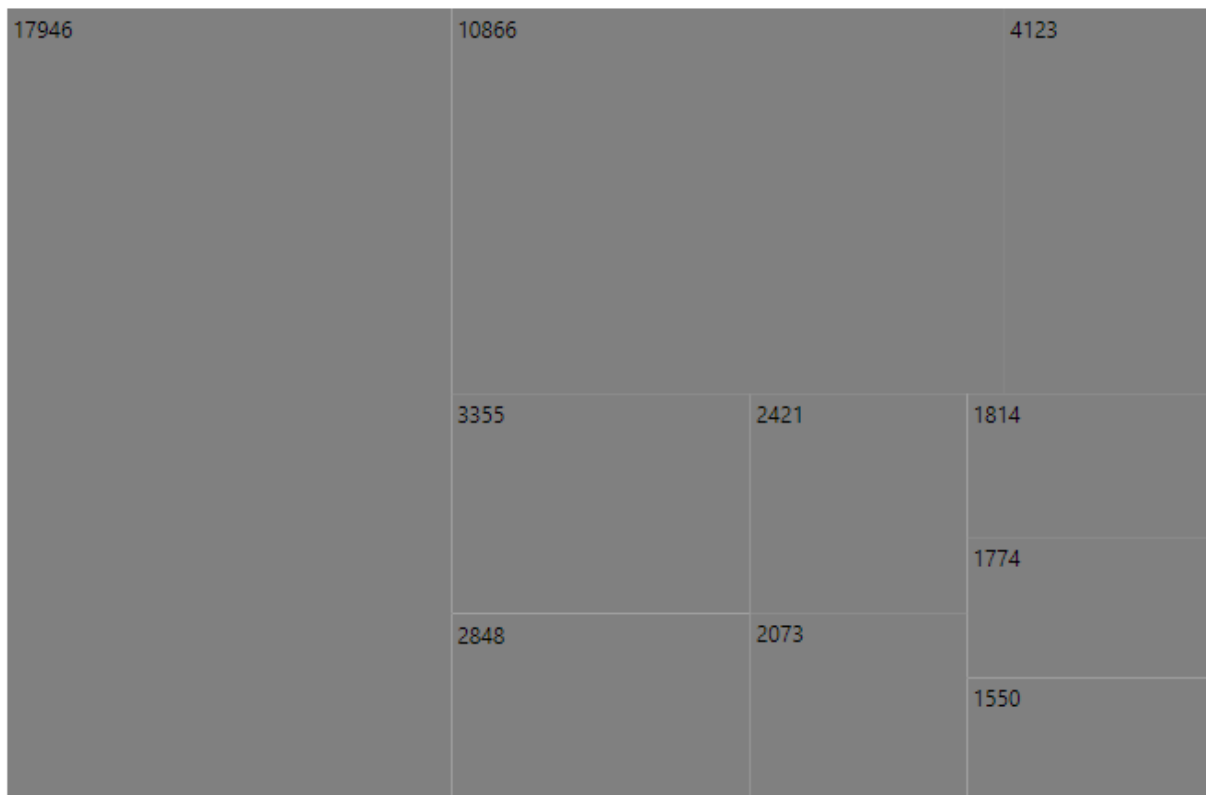
```
<SfTreeMap DataSource="GrowthReport"
WeightValuePath="GDP"
TValue="Country">
</SfTreeMap>
@code {
    public class Country
```



```

{
    public string Name { get; set; }
    public double GDP { get; set; }
}
public List<Country> GrowthReport = new List<Country> {
    new Country {Name="United States", GDP=17946 },
    new Country {Name="China", GDP=10866 },
    new Country {Name="Japan", GDP=4123 },
    new Country {Name="Germany", GDP=3355 },
    new Country {Name="United Kingdom", GDP=2848 },
    new Country {Name="France", GDP=2421 },
    new Country {Name="India", GDP=2073 },
    new Country {Name="Italy", GDP=1814 },
    new Country {Name="Brazil", GDP=1774 },
    new Country {Name="Canada", GDP=1550 }
};
}

```



Adding labels in TreeMap items

Add label text to the leaf items in the TreeMap component by setting the field name from data source in the [LabelPath](#) property in the [TreeMapLeafItemSettings](#), and it provides information to the user about the leaf items.

ASPX-CS

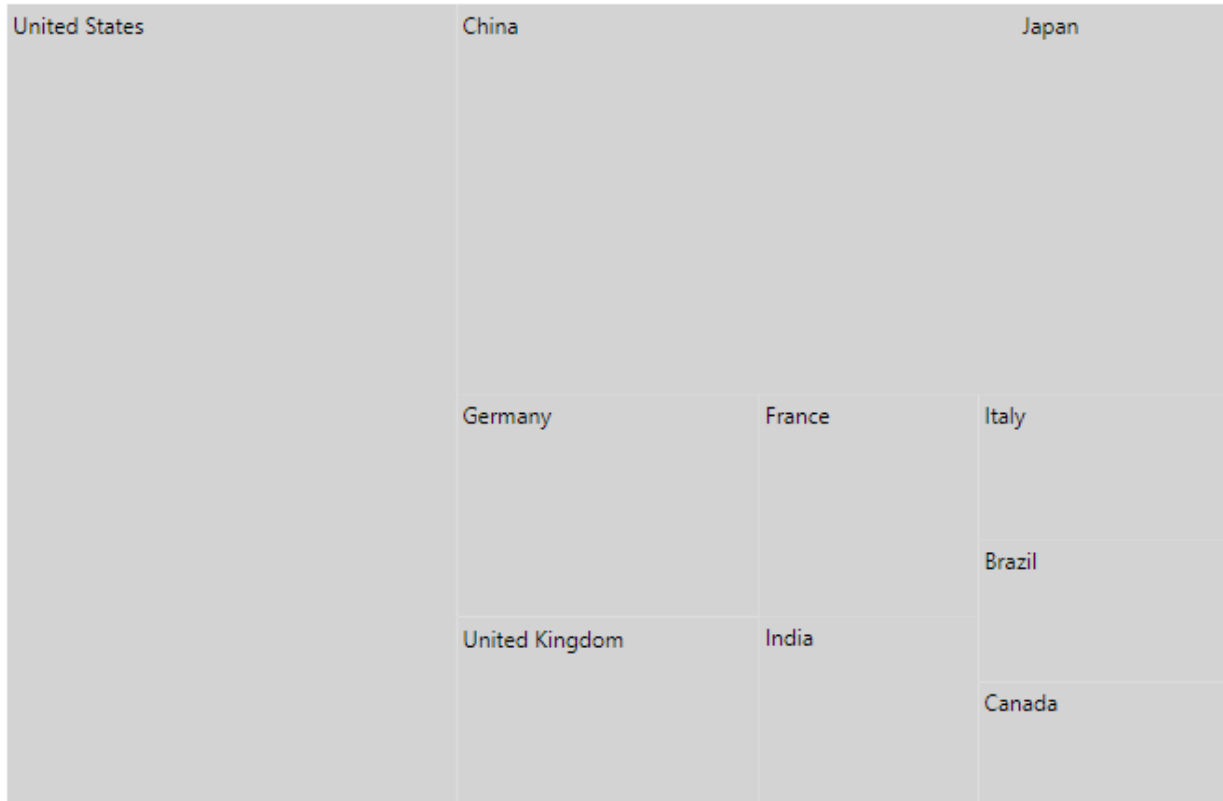
```

<SfTreeMap DataSource="GrowthReport"
    WeightValuePath="GDP"
    TValue="Country">

```

```
<TreeMapLeafItemSettings LabelPath="Name"  
Fill="lightgray"></TreeMapLeafItemSettings>  
</SfTreeMap>
```

Refer to the [code block](#) to know about the property value of the **GrowthReport**.



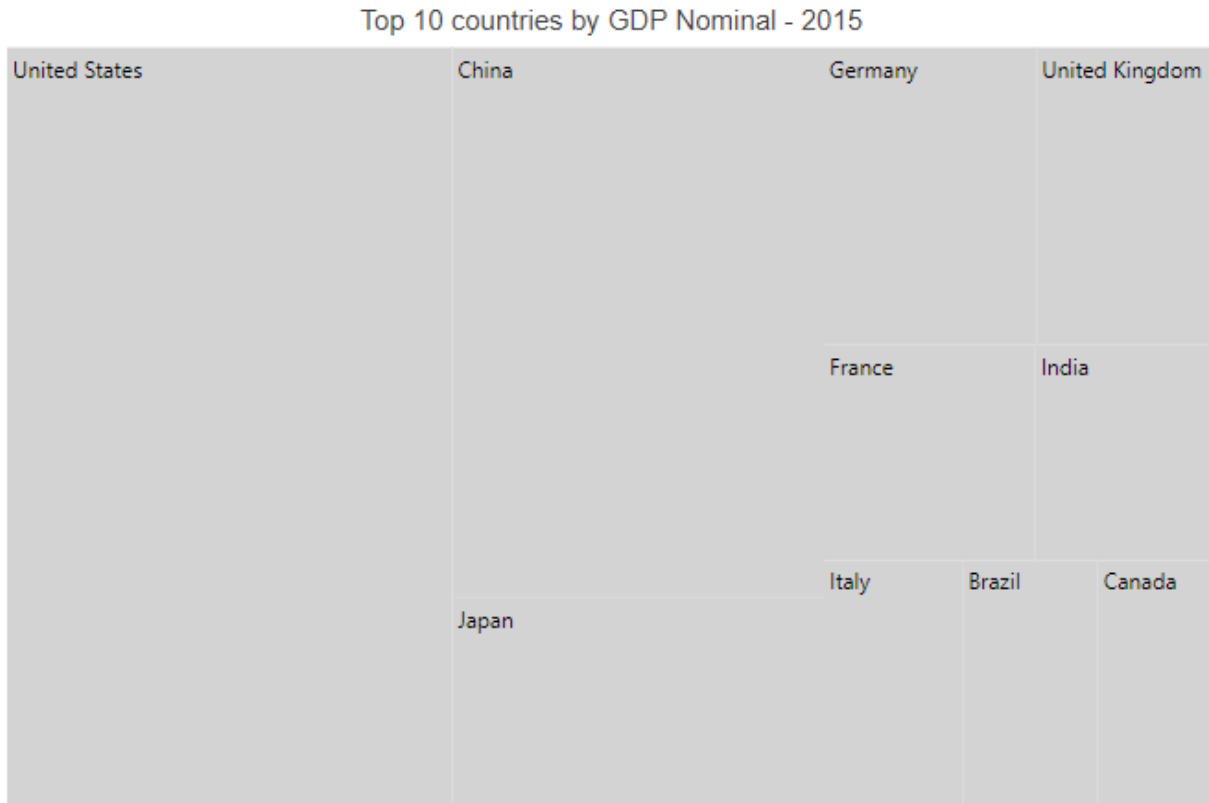
Adding title to TreeMap

Add a title using the [Text](#) property in the [TreeMapTitleSettings](#) to provide quick information to the user about the items rendered in the TreeMap.

ASPX-CS

```
<SfTreeMap DataSource="GrowthReport"  
WeightValuePath="GDP"  
TValue="Country">  
<TreeMapTitleSettings Text="Top 10 countries by GDP Nominal -  
2015"></TreeMapTitleSettings>  
<TreeMapLeafItemSettings LabelPath="Name"  
Fill="lightgray"></TreeMapLeafItemSettings>  
</SfTreeMap>
```

Refer to the [code block](#) to know the property value of the **GrowthReport**.



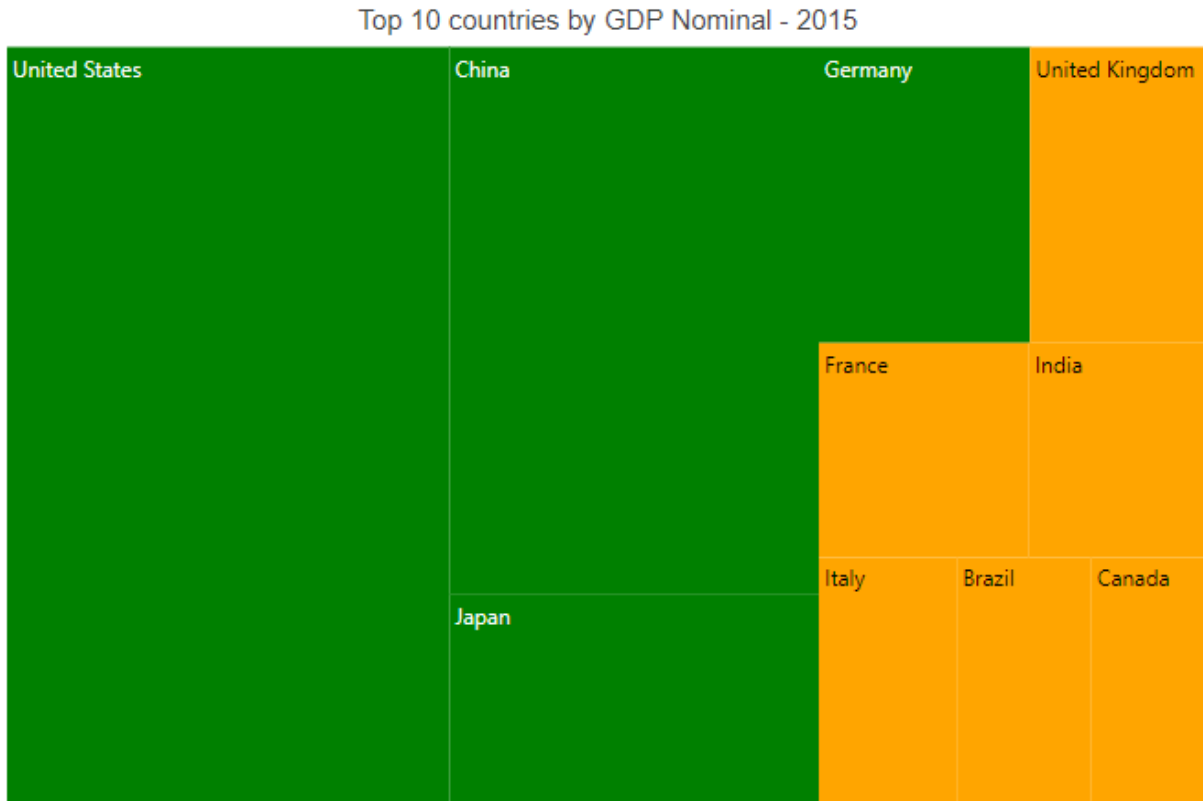
Apply color mapping

The color mapping supports customization of item colors based on the underlying value received from the bound data source. Specify the field name from which the values have to be compared for the items in the [RangeColorValuePath](#) property in [SfTreeMap](#). Also, specify range value and color in the [TreeMapLeafColorMapping](#). Here, in this example, “Orange” is specified for the range “0 - 3000” and “Green” is specified for the range “3000 - 20000”.

ASPX-CS

```
<SfTreeMap DataSource="GrowthReport"
WeightValuePath="GDP"
TValue="Country"
RangeColorValuePath="GDP">
  <TreeMapTitleSettings Text="Top 10 countries by GDP Nominal -
2015"></TreeMapTitleSettings>
  <TreeMapLeafItemSettings LabelPath="Name" Fill="lightgray">
    <TreeMapLeafColorMappings>
      <TreeMapLeafColorMapping From="0" To="3000" Color="@ (new string[] { "Orange"
}) "></TreeMapLeafColorMapping>
      <TreeMapLeafColorMapping From="3000" To="20000" Color="@ (new string[] {
"Green" }) "></TreeMapLeafColorMapping>
    </TreeMapLeafColorMappings>
  </TreeMapLeafItemSettings>
</SfTreeMap>
```

Refer to the [code block](#) to know about the property value of the **GrowthReport**.



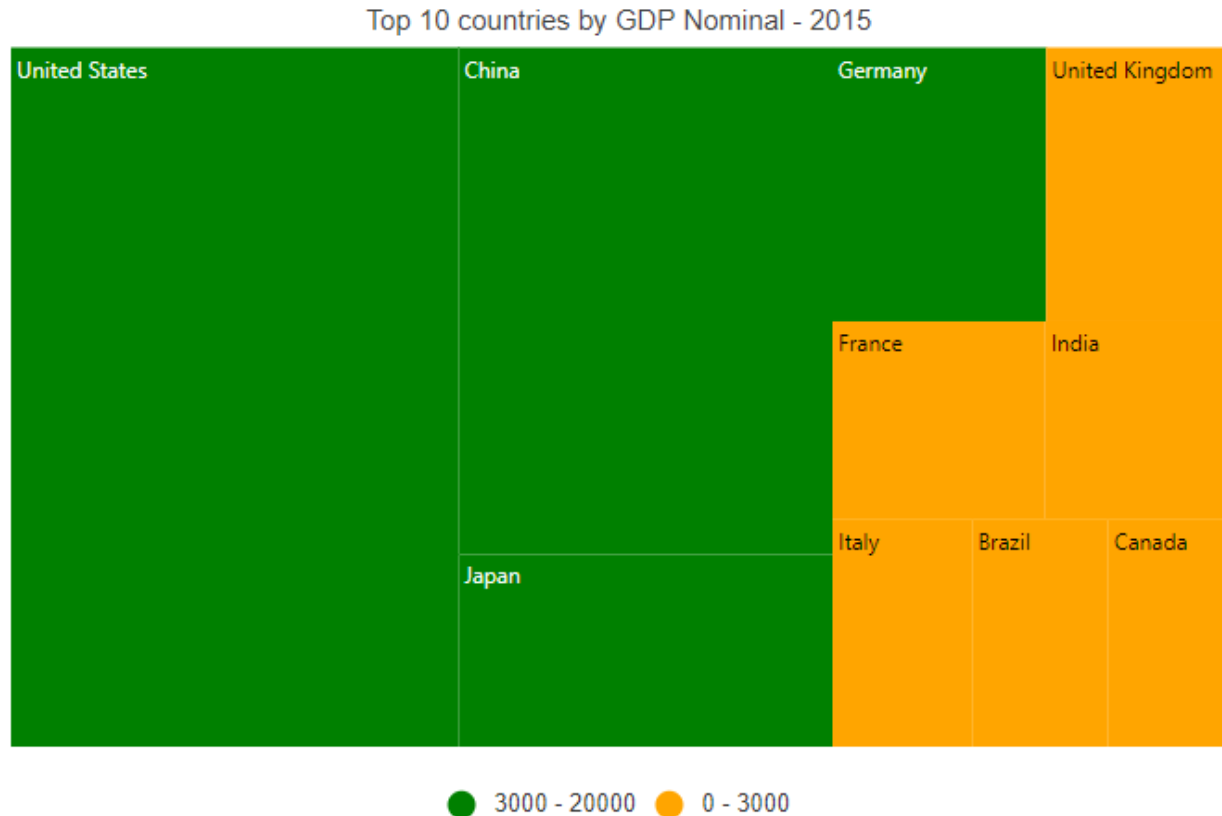
Enable legend

Legend items are used to denote the color mapping categories and show the legend for the TreeMap by setting the [Visible](#) property to **true** in the [TreeMapLegendSettings](#).

ASPX-CS

```
<SfTreeMap DataSource="GrowthReport"
WeightValuePath="GDP"
TValue="Country"
RangeColorValuePath="GDP">
  <TreeMapTitleSettings Text="Top 10 countries by GDP Nominal -
2015"></TreeMapTitleSettings>
  <TreeMapLeafItemSettings LabelPath="Name" Fill="lightgray">
  <TreeMapLeafColorMappings>
    <TreeMapLeafColorMapping From="0" To="3000" Color="@ (new string[] { "Orange"
    }) "></TreeMapLeafColorMapping>
    <TreeMapLeafColorMapping From="3000" To="20000" Color="@ (new string[] {
    "Green" }) "></TreeMapLeafColorMapping>
  </TreeMapLeafColorMappings>
  </TreeMapLeafItemSettings>
  <TreeMapLegendSettings Visible="true"></TreeMapLegendSettings>
</SfTreeMap>
```

Refer to the [code block](#) to know about the property value of the **GrowthReport**.



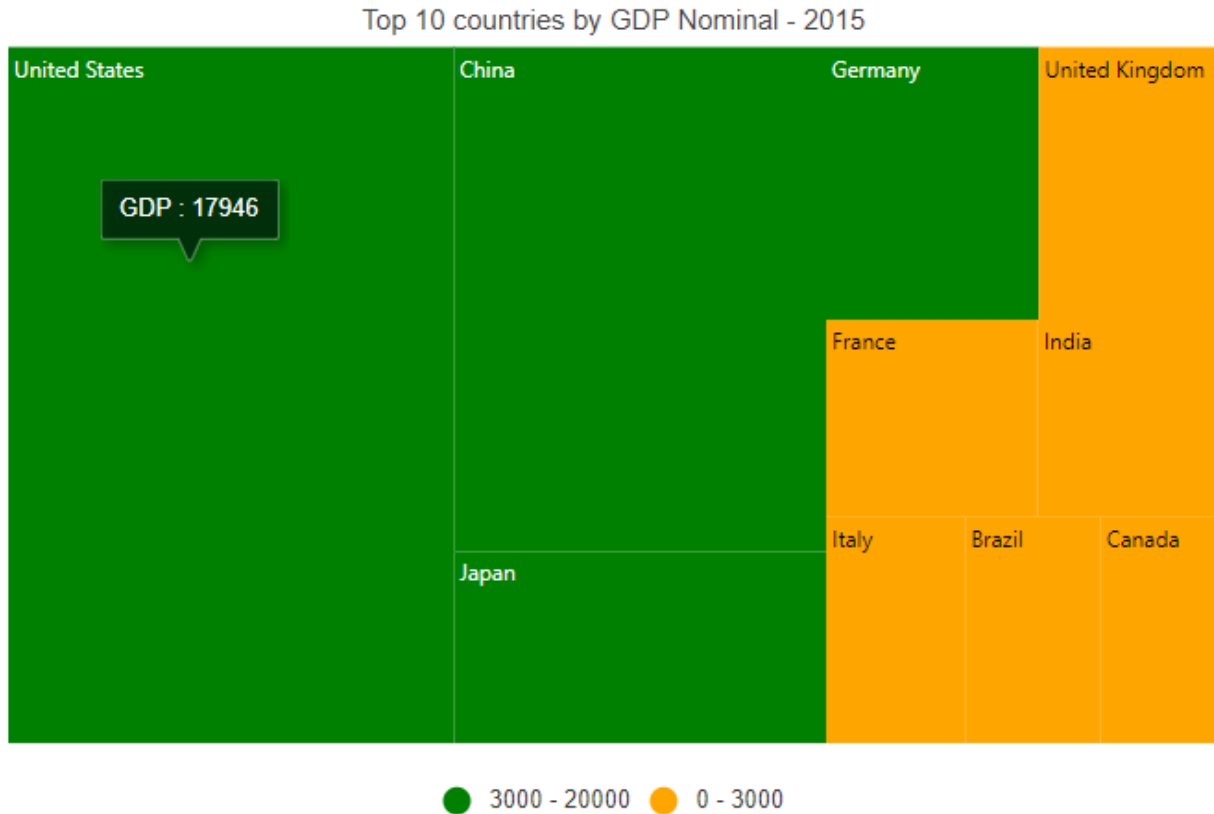
Enable tooltip

When space constraints prevents from displaying information using data labels, the tooltip comes in handy. The tooltip can be enabled by setting the [Visible](#) property to **true** in the [TreeMapTooltipSettings](#).

ASPX-CS

```
<SfTreeMap DataSource="GrowthReport"
WeightValuePath="GDP"
TValue="Country"
RangeColorValuePath="GDP">
  <TreeMapTitleSettings Text="Top 10 countries by GDP Nominal -
  2015"></TreeMapTitleSettings>
  <TreeMapLeafItemSettings LabelPath="Name" Fill="lightgray">
  <TreeMapLeafColorMappings>
  <TreeMapLeafColorMapping From="0" To="3000" Color="@ (new string[] { "Orange"
  }) "></TreeMapLeafColorMapping>
  <TreeMapLeafColorMapping From="3000" To="20000" Color="@ (new string[] {
  "Green" }) "></TreeMapLeafColorMapping>
  </TreeMapLeafColorMappings>
  </TreeMapLeafItemSettings>
  <TreeMapLegendSettings Visible="true"></TreeMapLegendSettings>
  <TreeMapTooltipSettings Visible="true"></TreeMapTooltipSettings>
</SfTreeMap>
```

Refer to the [code block](#) to know about the property value of the **GrowthReport**.



See also

- [Getting Started with Syncfusion Blazor for Client-Side in .NET Core CLI](#)
- [Getting Started with Syncfusion Blazor for Server-Side in Visual Studio](#)
- [Getting Started with Syncfusion Blazor for Server-Side in .NET Core CLI](#)

Data Binding in Blazor TreeMap Component

Populate data

The [DataSource](#) property accepts a collection of values as input. For example, a list of objects can be provided as input. Data can be given as either flat or hierarchical collection to the [DataSource](#) property.

Flat data

The following code example shows, how to bind a flat collection as data source to the TreeMap component.

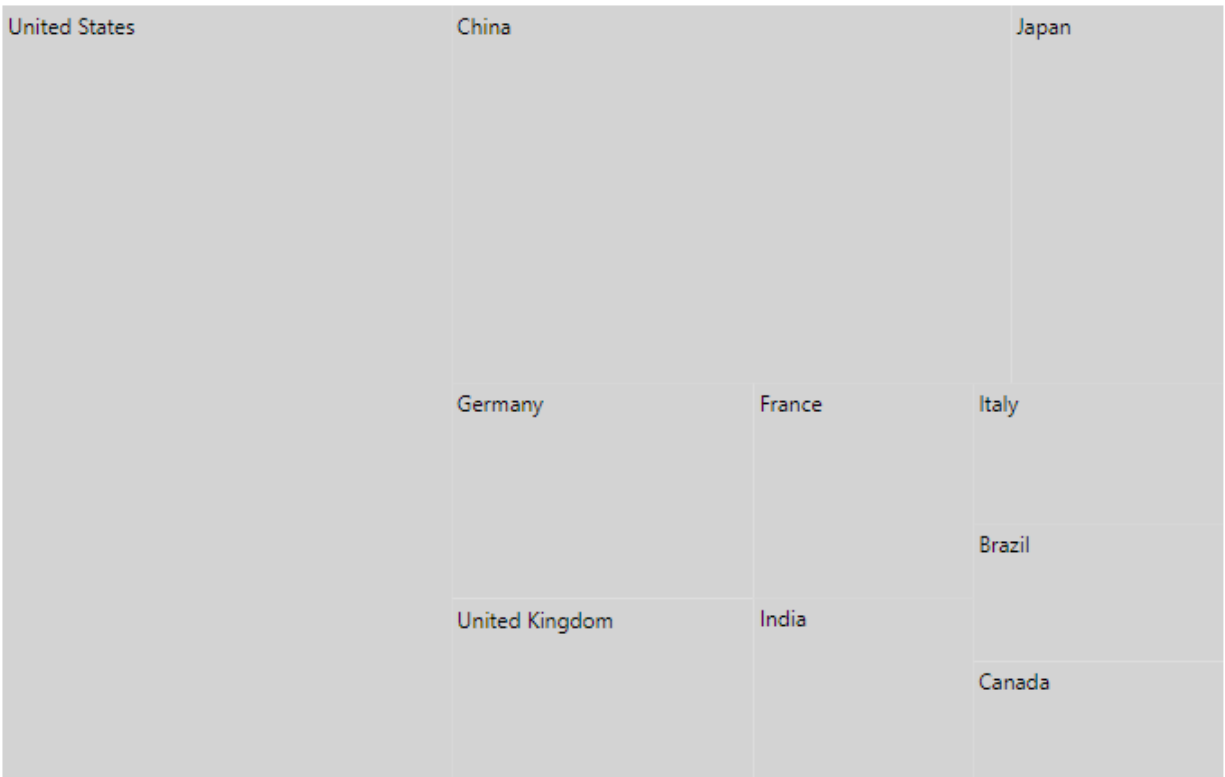
ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
<SfTreeMap WeightValuePath="GDP" TValue="GDPReport"
DataSource="GrowthReports">
  <TreeMapLeafItemSettings LabelPath="Country">
  </TreeMapLeafItemSettings>
</SfTreeMap>
@code {
public class GDPReport
{
```

```

public string Country { get; set; }
public double GDP { get; set; }
public double Percentage { get; set; }
public double Rank { get; set; }
};
public List<GDPReport> GrowthReports = new List<GDPReport> {
new GDPReport {Country = "United States", GDP=17946, Percentage=11.08,
Rank=1},
new GDPReport {Country="China", GDP=10866, Percentage= 28.42, Rank=2},
new GDPReport {Country="Japan", GDP=4123, Percentage=-30.78, Rank=3},
new GDPReport {Country="Germany", GDP=3355, Percentage=-5.19, Rank=4},
new GDPReport {Country="United Kingdom", GDP=2848, Percentage=8.28, Rank=5},
new GDPReport {Country="France", GDP=2421, Percentage=-9.69, Rank=6},
new GDPReport {Country="India", GDP=2073, Percentage=13.65, Rank=7},
new GDPReport {Country="Italy", GDP=1814, Percentage=-12.45, Rank=8},
new GDPReport {Country="Brazil", GDP=1774, Percentage=-27.88, Rank=9},
new GDPReport {Country="Canada", GDP=1550, Percentage=-15.02, Rank=10}
};
}

```



Hierarchical data

The following code example shows, how to bind a hierarchical collection as data source to the TreeMap component.

ASPX-CS

```

@using Syncfusion.Blazor.TreeMap
<SfTreeMap WeightValuePath="Population" DataSource="PopulationReport">
  <TreeMapLeafItemSettings LabelPath="Name" Fill="#0077b3">

```

```

<TreeMapLeafBorder Width="0.5" Color="black"></TreeMapLeafBorder>
</TreeMapLeafItemSettings>
<TreeMapLevels>
<TreeMapLevel GroupPath="Continent" Fill="#004466">
<TreeMapLevelBorder Width="0.5" Color="black"></TreeMapLevelBorder>
</TreeMapLevel>
<TreeMapLevel GroupPath="States" Fill="#0099e6">
<TreeMapLevelBorder Width="0.5" Color="black"></TreeMapLevelBorder>
</TreeMapLevel>
</TreeMapLevels>
</SfTreeMap>
@code{
public List<object> PopulationReport { get; set; } = new List<object>
{
    new {
        Continent = new List<object> { new {
            Name= "Africa",
            Population= 1216130000,
            States= new List<object> { new {
                Name= "Eastern Africa",
                Population= 410637987,
                Region= new List<object> { new {
                    Name= "Ethiopia",
                    Population= 107534882
                }}
            },
            new {
                Name= "Middle Africa",
                Population= 158562976,
                Region= new List<object>{ new {
                    Name= "Democratic, Republic of the Congo",
                    Population= 84004989
                }}
            }
        }}
    },
    new {
        Continent= new List<object> { new {
            Name= "Asia",
            Population= 4436224000,
            States= new List<object> { new {
                Name= "Central Asia",
                Population= 69787760,
                Region= new List<object> { new {
                    Name= "Uzbekistan",
                    Population= 32364996
                }}
            },
            new {
                Name= "Eastern Asia",
                Population= 1641908531,
                Region= new List<object> { new {
                    Name= "China",
                    Population= 1415045928
                }}
            }
        }}
    }
}
}

```



```

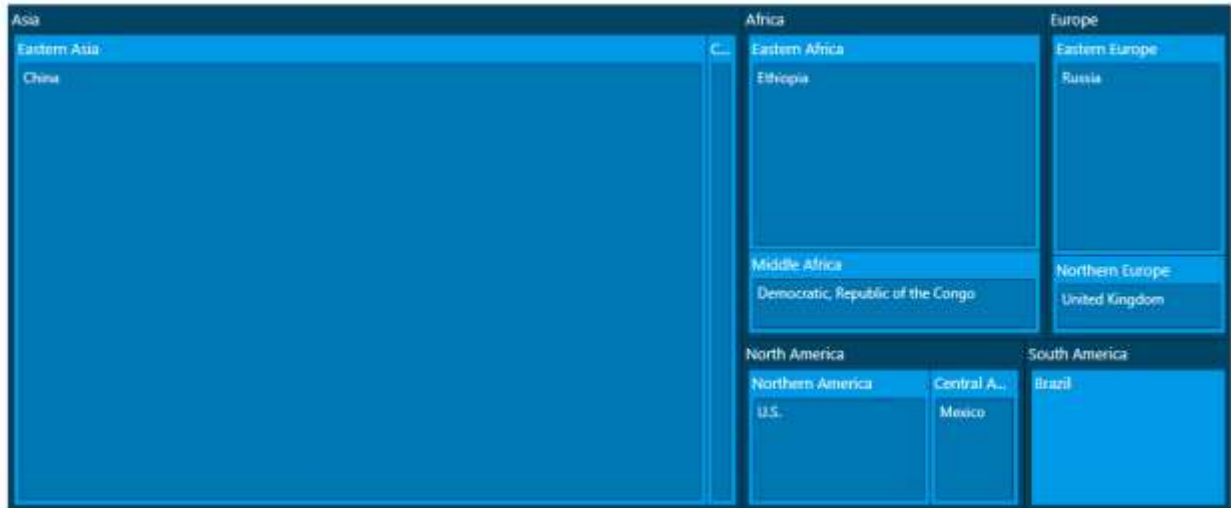
    }
  }
},
new {
  Continent= new List<object> { new {
    Name= "North America",
    Population= 579024000,
    States= new List<object> { new {
      Name= "Central America",
      Population= 174988756,
      Region= new List<object> { new {
        Name= "Mexico",
        Population= 130759074
      }}
    }},
  },
  new {
    Name= "Northern America",
    Population= 358593810,
    Region= new List<object> { new {
      Name= "U.S.",
      Population= 3267667480
    }}
  }
},
new {
  Continent= new List<object> { new {
    Name= "South America",
    Population= 422535000,
    States= new List<object> { new {
      Name= "Brazil",
      Population= 204519000
    }}
  }},
},
new {
  Continent= new List<object> { new {
    Name= "Europe",
    Population= 738849000,
    States= new List<object> { new {
      Name= "Eastern Europe",
      Population= 291953328,
      Region= new List<object> { new {
        Name= "Russia",
        Population= 143964709
      }}
    }},
  },
  new {
    Name= "Northern Europe",
    Population= 103642971,
    Region= new List<object> { new {
      Name= "United Kingdom",
      Population= 66573504
    }}
  }
}
}

```

```

    }
  }
};
}

```



Local data

Fetching data from the collection

The following code example shows, how to bind a `IEnumerable` object to the TreeMap component as a data source.

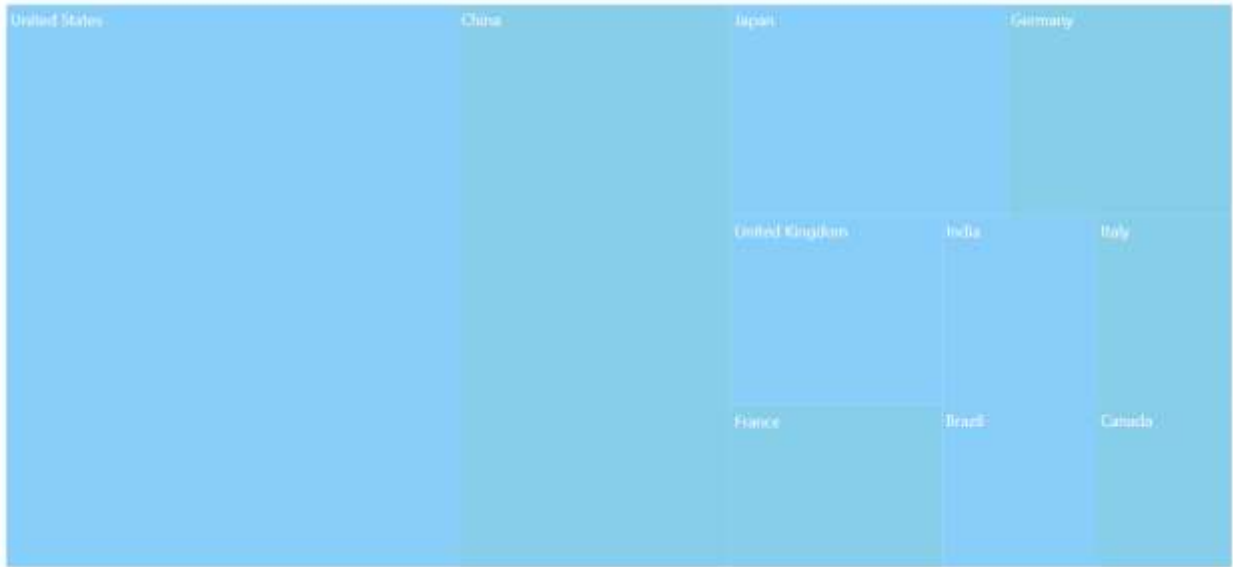
ASPX-CS

```

@using Syncfusion.Blazor.TreeMap
<SfTreeMap DataSource="GrowthReports" TValue="GDPReport" Palette="@Palette"
WeightValuePath="GDP">
  <TreeMapLeafItemSettings LabelPath="CountryName">
    <TreeMapLeafLabelStyle Color="White"></TreeMapLeafLabelStyle>
  </TreeMapLeafItemSettings>
</SfTreeMap>
@code{
public class GDPReport
{
    public string CountryName { get; set; }
    public double GDP { get; set; }
    public double Percentage { get; set; }
    public int Rank { get; set; }
};
public string[] Palette = new string[] { "#87CEFA", "#87CEEB" };
public List<GDPReport> GrowthReports = new List<GDPReport> {
    new GDPReport { CountryName="United States", GDP=17946, Percentage=11.08, Rank=1},
    new GDPReport { CountryName="China", GDP=10866, Percentage= 28.42, Rank=2},
    new GDPReport { CountryName="Japan", GDP=4123, Percentage=-30.78, Rank=3},
    new GDPReport { CountryName="Germany", GDP=3355, Percentage=-5.19, Rank=4},
    new GDPReport { CountryName="United Kingdom", GDP=2848, Percentage=8.28, Rank=5},
    new GDPReport { CountryName="France", GDP=2421, Percentage=-9.69, Rank=6},

```

```
new GDPReport {CountryName="India", GDP=2073, Percentage=13.65, Rank=7},
new GDPReport {CountryName="Italy", GDP=1814, Percentage=-12.45, Rank=8},
new GDPReport {CountryName="Brazil", GDP=1774, Percentage=-27.88, Rank=9},
new GDPReport {CountryName="Canada", GDP=1550, Percentage=-15.02, Rank=10}
};
}
```



Fetching data from the JSON file

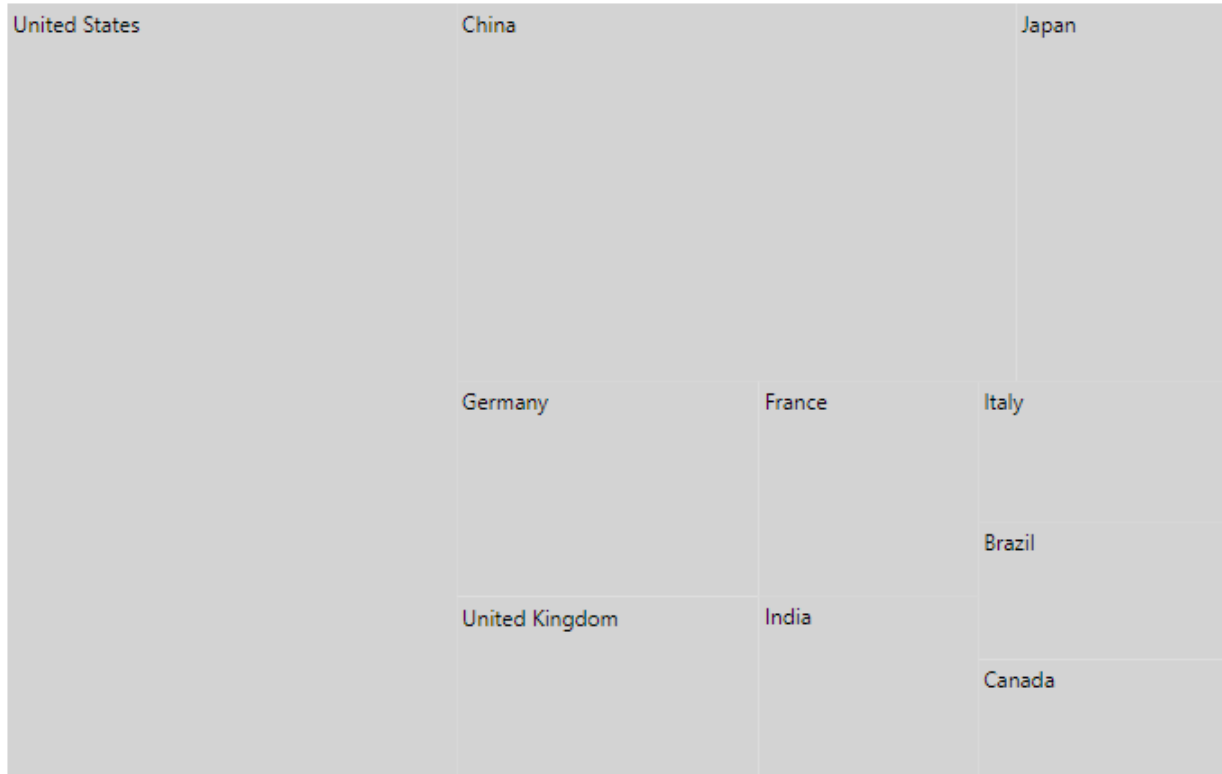
Read the JSON file data and it can be converted to the C# object, and assign it to the [DataSource](#) property of the TreeMap component.

The `Http.GetJsonAsync` method is used in the `OnInitializedAsync` life cycle method to load the JSON file data.

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
@inject HttpClient Http;
@if (GrowthReports == null)
{
    <p><em>Loading TreeMap component...</em></p>
}
else
{
    <SfTreeMap WeightValuePath="GDP" TValue="GDPReport"
    DataSource="GrowthReports">
        <TreeMapLeafItemSettings LabelPath="State">
        </TreeMapLeafItemSettings>
    </SfTreeMap>
}
@code{
public List<GDPReport> GrowthReports { get; set; }
protected override Task OnInitializedAsync()
{
    GrowthReports = await Http.GetJsonAsync<List<GDPReport>>("sample-
data/product-growth.json");
}
```

```
return base.OnInitializedAsync();
}
public class GDPReport
{
    public string State { get; set; }
    public int GDP { get; set; }
    public double Percentage { get; set; }
    public int Rank { get; set; }
};
}
```



Remote data

To interact with the remote data source, provide the endpoint [Url](#) within the [SfDataManager](#) class along with an appropriate [Adaptor](#). By default, the [SfDataManager](#) uses [ODataAdaptor](#) for remote data-binding.

If [SfDataManager](#) is used for data binding then the **TValue** must be provided explicitly to the TreeMap component.

Binding with OData services

[OData](#) is a standardized protocol for creating and consuming data. User can retrieve data from [OData](#) service using the [SfDataManager](#).

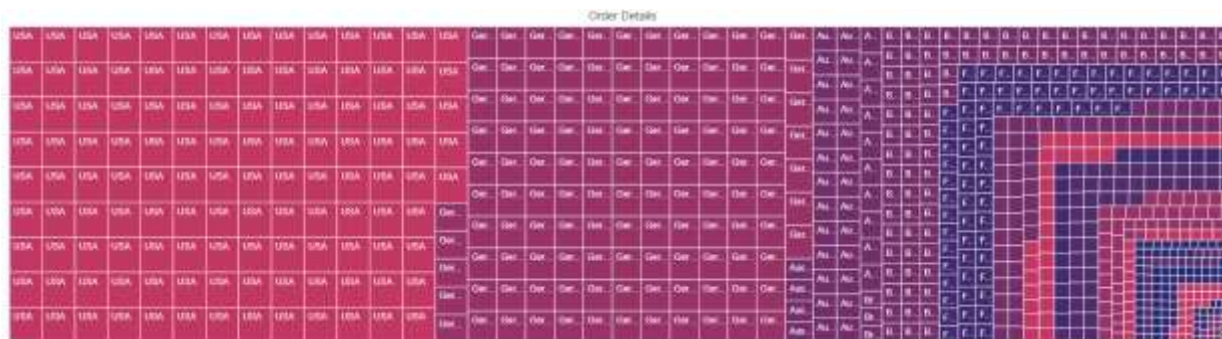
ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
@using Syncfusion.Blazor.Data
```

```

<SfTreeMap TValue="OrderDetails" WeightValuePath="Freight"
Palette="@Palette">
<SfDataManager
Url="https://js.syncfusion.com/ejServices/Wcf/Northwind.svc/Orders"
Adaptor="Syncfusion.Blazor.Adaptors.ODataAdaptor">
</SfDataManager>
<TreeMapTitleSettings Text="Order Details">
</TreeMapTitleSettings>
<TreeMapLeafItemSettings LabelPath="ShipCountry">
<TreeMapLeafBorder Color="white" Width="0.5">
</TreeMapLeafBorder>
</TreeMapLeafItemSettings>
<TreeMapTooltipSettings Visible="true">
</TreeMapTooltipSettings>
</SfTreeMap>
@code{
public string[] Palette = new string[] { "#C33764", "#AB3566", "#993367",
"#853169", "#742F6A", "#632D6C", "#532C6D", "#412A6F", "#312870", "#1D2671"
};
public class OrderDetails
{
public int OrderID { get; set; }
public string OrderDate { get; set; }
public string CustomerID { get; set; }
public string ShipCountry { get; set; }
public double Freight { get; set; }
}
}

```



Binding with OData V4 services

The [OData V4](#) is an improved version of [OData](#) protocols, and the [SfDataManager](#) can be used to retrieve and consume [OData V4](#) services.

For more details on OData V4 services, refer to the [OData documentation](#) to bind [OData V4](#) service using the [OData V4 Adaptor](#).

ASPX-CS

```

@using Syncfusion.Blazor.TreeMap
@using Syncfusion.Blazor.Data
<SfTreeMap TValue="OrderDetails" WeightValuePath="Freight"
Palette="@Palette">

```

```
<SfDataManager>
Url="https://services.odata.org/V4/Northwind/Northwind.svc/Orders/"
Adaptor="Syncfusion.Blazor.Adaptors.ODataV4Adaptor"></SfDataManager>
<TreeMapTitleSettings Text="Order Details">
</TreeMapTitleSettings>
<TreeMapLeafItemSettings LabelPath="ShipCountry">
<TreeMapLeafBorder Color="white" Width="0.5">
</TreeMapLeafBorder>
</TreeMapLeafItemSettings>
<TreeMapTooltipSettings Visible="true"></TreeMapTooltipSettings>
</SfTreeMap>

@code{
public string[] Palette = new string[] { "#C33764", "#AB3566", "#993367",
"#853169", "#742F6A", "#632D6C", "#532C6D", "#412A6F", "#312870", "#1D2671"
};

public class OrderDetails
{
public int OrderID { get; set; }
public string OrderDate { get; set; }
public string CustomerID { get; set; }
public string ShipCountry { get; set; }
public double Freight { get; set; }
}
}
```



Web API

Use [WebApiAdaptor](#) to bind TreeMap with Web API, created using OData endpoint.

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
@using Syncfusion.Blazor.Data
<SfTreeMap TValue="OrderDetails" WeightValuePath="Freight"
Palette="@Palette">
<SfDataManager Url="https://ej2services.syncfusion.com/production/web-
services/api/Orders" Adaptor="Syncfusion.Blazor.Adaptors.WebApiAdaptor">
</SfDataManager>
<TreeMapTitleSettings Text="Order Details">
</TreeMapTitleSettings>
<TreeMapLeafItemSettings LabelPath="ShipCity">
<TreeMapLeafBorder Color="white" Width="0.5">
</TreeMapLeafBorder>
</TreeMapLeafItemSettings>
<TreeMapTooltipSettings Visible="true">
```

```

</TreeMapTooltipSettings>
</SfTreeMap>
@code{
public string[] Palette = new string[] { "#C33764", "#AB3566", "#993367",
"#853169", "#742F6A", "#632D6C", "#532C6D", "#412A6F", "#312870", "#1D2671"
};
public class OrderDetails
{
public int OrderID { get; set; }
public string OrderDate { get; set; }
public string CustomerID { get; set; }
public string ShipCity { get; set; }
public double Freight { get; set; }
}
}

```



Entity Framework

Entity Framework acts as a modern object-database mapped for .NET. This section explains, how to consume data from the **Microsoft SQL Server** database and bind it to the TreeMap component.

Create DbContext class

The first step is to create a DbContext class called **OrderContext** for establishing connection to a Microsoft SQL Server database.

C# SHARP

```

using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using EFTreeMap.Data;
namespace EFTreeMap.Data
{
public class OrderContext : DbContext
{
public virtual DbSet<Order> Orders { get; set; }
protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
{
if (!optionsBuilder.IsConfigured)
{
// Configures the context to connect to a Microsoft SQL Serve database

```

```

optionsBuilder.UseSqlServer(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename='D:\blazor\EFTreeMap\App_Data
\NORTHWND.MDF';Integrated Security=True;Connect Timeout=30");
}
}
}
public class Order
{
    [Key]
    public int? OrderID { get; set; }
    [Required]
    public string CustomerID { get; set; }
    [Required]
    public int EmployeeID { get; set; }
}
}

```

Create data access layer to perform data operation

Now create a class called **OrderDataAccessLayer**, which acts as a data access layer to retrieve the records from the database table.

CSHARP

```

using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using EFTreeMap.Data;
namespace EFTreeMap.Data
{
    public class OrderDataAccessLayer
    {
        OrderContext db = new OrderContext();
        //To Get all Orders details
        public DbSet<Order> GetAllOrders()
        {
            try
            {
                return db.Orders;
            }
            catch
            {
                throw;
            }
        }
    }
}

```

Creating Web API Controller

A Web API Controller must be created, which allows the TreeMap to directly consume data from the Entity Framework.

CSHARP


```

using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Primitives;
using EFTreeMap.Data;
namespace EFTreeMap.Controller
{
    [Route("api/[controller]")]
    [ApiController]
    public class DefaultController : ControllerBase
    {
        OrderDataAccessLayer db = new OrderDataAccessLayer();
        [HttpGet]
        public object Get()
        {
            IQueryable<Order> data = db.GetAllOrders().AsQueryable();
            var count = data.Count();
            var queryString = Request.Query;
            if (queryString.Keys.Contains("$inlinecount"))
            {
                StringValues Skip;
                StringValues Take;
                int skip = (queryString.TryGetValue("$skip", out Skip)) ?
                    Convert.ToInt32(Skip[0]) : 0;
                int top = (queryString.TryGetValue("$top", out Take)) ?
                    Convert.ToInt32(Take[0]) : data.Count();
                return new { Items = data.Skip(skip).Take(top), Count = count };
            }
            else
            {
                return data;
            }
        }
    }
}

```

Add Web API Controller services in Startup.cs

Open the **Startup.cs** file, add services and endpoints required for the Web API Controller as follows.

CSHARP

```

using Newtonsoft.Json.Serialization;
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void ConfigureServices(IServiceCollection services)
        {
            ....
            ....
        }
    }
}

```

```

services.AddSingleton<OrderDataAccessLayer>();
// Adds services for controllers to the specified
Microsoft.Extensions.DependencyInjection.IServiceCollection.
services.AddControllers().AddNewtonsoftJson(options =>
{
options.SerializerSettings.ContractResolver = new DefaultContractResolver();
});
}
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
....
....
app.UseEndpoints(endpoints =>
{
// Adds endpoints for controller actions to the
Microsoft.AspNetCore.Routing.IEndpointRouteBuilder
endpoints.MapDefaultControllerRoute();
.....
.....
});
}
}
}

```

Configure treemap component

Configure the TreeMap to bind data using either [DataSource](#) property or [SfDataManager](#).

For instance, bind the data directly from the **OrderDataAccessLayer** class and assign to the [DataSource](#) property.

CSHARP

```

@inject OrderDataAccessLayer OrderData
@using EFTreeMap.Data
@using Syncfusion.Blazor.TreeMap
<SfTreeMap TValue="Order" WeightValuePath="OrderID" Palette="@Palette"
DataSource="@OrderData.GetAllOrders()">
<TreeMapTitleSettings Text="Order Details">
</TreeMapTitleSettings>
<TreeMapLeafItemSettings LabelPath="CustomerID">
<TreeMapLeafBorder Color="white" Width="0.5">
</TreeMapLeafBorder>
</TreeMapLeafItemSettings>
<TreeMapTooltipSettings Visible="true"></TreeMapTooltipSettings>
</SfTreeMap>
@code{
public string[] Palette = new string[] { "#C33764", "#AB3566", "#993367",
"#853169", "#742F6A", "#632D6C", "#532C6D", "#412A6F", "#312870", "#1D2671"
};
}

```

On the other hand, to configure the TreeMap using Web API, provide the appropriate endpoint URL in the [SfDataManager](#) along with [Adaptor](#). Here, [WebApiAdaptor](#) is used to interact with the Web API to consume data from the Entity Framework appropriately.

ASPX-CS

```

@using Syncfusion.Blazor.TreeMap
@using Syncfusion.Blazor.Data
<SfTreeMap TValue="Order" WeightValuePath="OrderID" Palette="@Palette">
<SfDataManager Url="api/Default"
Adaptor="Syncfusion.Blazor.Adaptors.WebApiAdaptor">
</SfDataManager>
<TreeMapTitleSettings Text="Order Details">
</TreeMapTitleSettings>
<TreeMapLeafItemSettings LabelPath="CustomerID">
<TreeMapLeafBorder Color="white" Width="0.5">
</TreeMapLeafBorder>
</TreeMapLeafItemSettings>
<TreeMapTooltipSettings Visible="true"></TreeMapTooltipSettings>
</SfTreeMap>
@code{
public string[] Palette = new string[] { "#C33764", "#AB3566", "#993367",
"#853169", "#742F6A", "#632D6C", "#532C6D", "#412A6F", "#312870", "#1D2671"
};
}

```

Layout in Blazor TreeMap Component

Determine the visual representation of nodes belonging to all the TreeMap levels using the [LayoutType](#) property.

Types of layout

The available layout types are,

- [Squarified](#)
- [SliceAndDiceVertical](#)
- [SliceAndDiceHorizontal](#)
- [SliceAndDiceAuto](#)

Squarified

The [Squarified](#) layout displays the nested rectangles based on aspect ratio in the TreeMap. The rectangles will be split based on the height and width of the parent. The default rendering type of the layout is [Squarified](#).

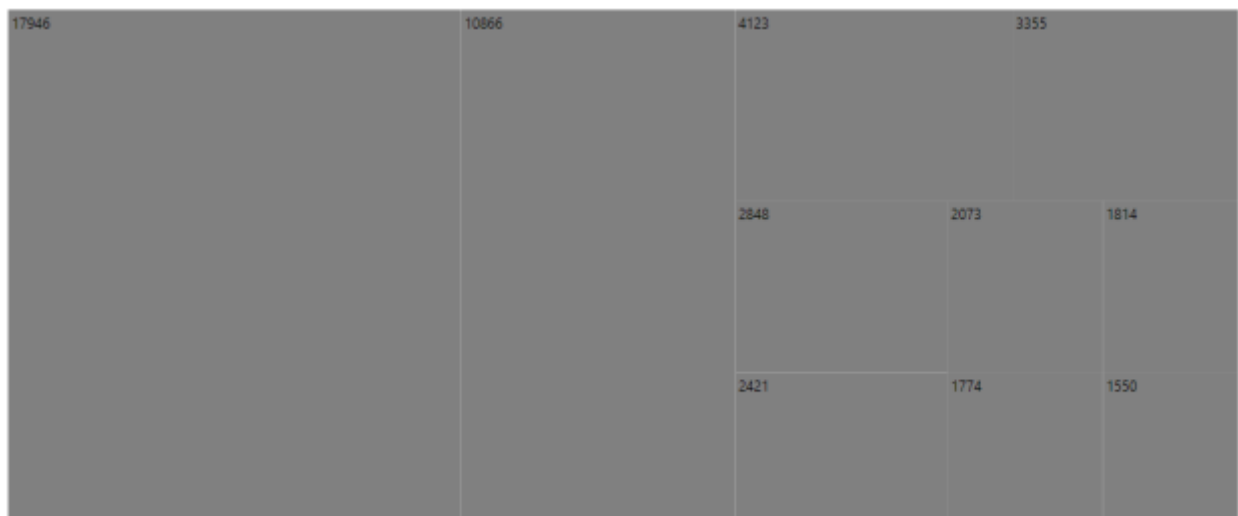
ASPX-CS

```

@using Syncfusion.Blazor.TreeMap
<SfTreeMap DataSource="GrowthReports" TValue="GDPReport"
WeightValuePath="GDP">
</SfTreeMap>
@code {
public class GDPReport
{
public string CountryName { get; set; }
public double GDP { get; set; }
public double Percentage { get; set; }
public int Rank { get; set; }
};
public List<GDPReport> GrowthReports = new List<GDPReport> {

```

```
new GDPReport {CountryName="United States", GDP=17946, Percentage=11.08, Rank=1},
new GDPReport {CountryName="China", GDP=10866, Percentage= 28.42, Rank=2},
new GDPReport {CountryName="Japan", GDP=4123, Percentage=-30.78, Rank=3},
new GDPReport {CountryName="Germany", GDP=3355, Percentage=-5.19, Rank=4},
new GDPReport {CountryName="United Kingdom", GDP=2848, Percentage=8.28, Rank=5},
new GDPReport {CountryName="France", GDP=2421, Percentage=-9.69, Rank=6},
new GDPReport {CountryName="India", GDP=2073, Percentage=13.65, Rank=7},
new GDPReport {CountryName="Italy", GDP=1814, Percentage=-12.45, Rank=8},
new GDPReport {CountryName="Brazil", GDP=1774, Percentage=-27.88, Rank=9},
new GDPReport {CountryName="Canada", GDP=1550, Percentage=-15.02, Rank=10}
};
}
```



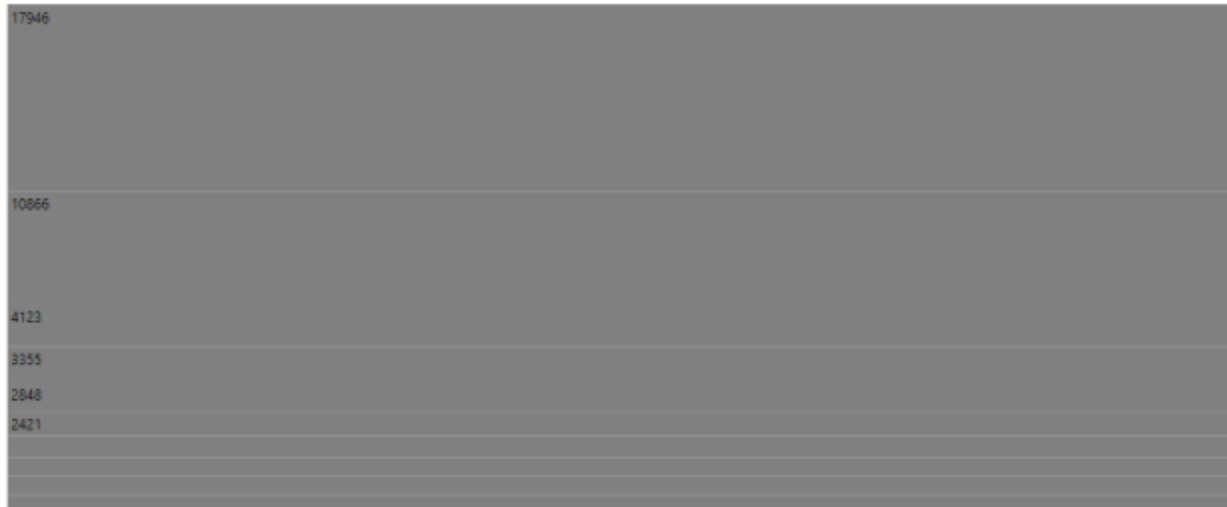
Slice and dice vertical

The [SliceAndDiceVertical](#) layout creates rectangles with high aspect ratio and displays items in a vertically sorted order.

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap;
<SfTreeMap DataSource="GrowthReports" TValue="GDPReport"
WeightValuePath="GDP" LayoutType="LayoutMode.SliceAndDiceVertical">
</SfTreeMap>
```

Refer to the [code block](#) to know about the property value of the **GrowthReports**.



Slice and dice horizontal

The [SliceAndDiceHorizontal](#) layout creates rectangles with high aspect ratio and displays items in a horizontally sorted order.

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap;
<SfTreeMap DataSource="GrowthReports" TValue="GDPReport"
WeightValuePath="GDP" LayoutType="LayoutMode.SliceAndDiceHorizontal">
</SfTreeMap>
```

Refer to the [code block](#) to know about the property value of the **GrowthReports**.



Slice and dice auto

The [SliceAndDiceAuto](#) layout creates rectangles with high aspect ratio and displays items sorted both horizontally and vertically.

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap;
```

```
<SfTreeMap DataSource="GrowthReports" TValue="GDPReport"
WeightValuePath="GDP" LayoutType="LayoutMode.SliceAndDiceAuto">
</SfTreeMap>
```

Refer to the [code block](#) to know about the property value of the **GrowthReports**.



Rendering direction

The direction of the TreeMap item is [TopLeftBottomRight](#) by default and customize the rendering direction of the TreeMap item by setting the [RenderDirection](#) property.

The TreeMap can be rendered in the following directions:

- TopLeftBottomRight,
- TopRightBottomLeft,
- BottomRightTopLeft
- BottomLeftTopRight

The following example shows, how to render the TreeMap in the RTL direction with `TopLeftBottomRight`.

CSHARP

```
@using Syncfusion.Blazor.TreeMap;
<SfTreeMap DataSource="Fruits" TValue="Fruit" WeightValuePath="Count"
Palette='new string[] { "#71B081", "#5A9A77", "#498770", "#39776C",
"#266665", "#124F5E" }' RenderDirection="RenderingMode.TopLeftBottomRight">
<TreeMapLeafItemSettings LabelPath="Name">
</TreeMapLeafItemSettings>
<TreeMapTooltipSettings Visible=true Format="${Count} : ${Name}">
</TreeMapTooltipSettings>
</SfTreeMap>
@code{
public class Fruit
{
public string Name { get; set; }
public int Count { get; set; }
};
```

```
public List<Fruit> Fruits = new List<Fruit> {
    new Fruit { Name="Apple", Count=5000 },
    new Fruit { Name="Mango", Count=3000 },
    new Fruit { Name="Orange", Count=2300 },
    new Fruit { Name="Banana", Count=500 },
    new Fruit { Name="Grape", Count=4300 },
    new Fruit { Name="Papaya", Count=1200 },
    new Fruit { Name="Melon", Count=4500 }
};
}
```



The following example shows, how to render the TreeMap in the RTL direction with [TopRightBottomLeft](#).

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap;
<SfTreeMap DataSource="Fruits" TValue="Fruit" WeightValuePath="Count"
Palette='new string[] {"#71B081", "#5A9A77", "#498770", "#39776C",
"#266665", "#124F5E"}' RenderDirection="RenderingMode.TopRightBottomLeft">
<TreeMapLeafItemSettings LabelPath="Name">
</TreeMapLeafItemSettings>
<TreeMapTooltipSettings Visible=true Format="{Count} : {Name}">
</TreeMapTooltipSettings>
</SfTreeMap>
```

Refer to the [code block](#) to know about the property value of **Fruits**.



The following example shows, how to render the TreeMap in the RTL direction with [BottomRightToLeft](#).

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap;
<SfTreeMap DataSource="Fruits" TValue="Fruit" WeightValuePath="Count"
Palette='new string[]{"#71B081", "#5A9A77", "#498770", "#39776C",
"#266665", "#124F5E"}' RenderDirection="RenderingMode.BottomRightToLeft">
<TreeMapLeafItemSettings LabelPath="Name">
</TreeMapLeafItemSettings>
<TreeMapTooltipSettings Visible=true Format="{Count} : {Name}">
</TreeMapTooltipSettings>
</SfTreeMap>
```

Refer to the [code block](#) to know about the property value of **Fruits**.

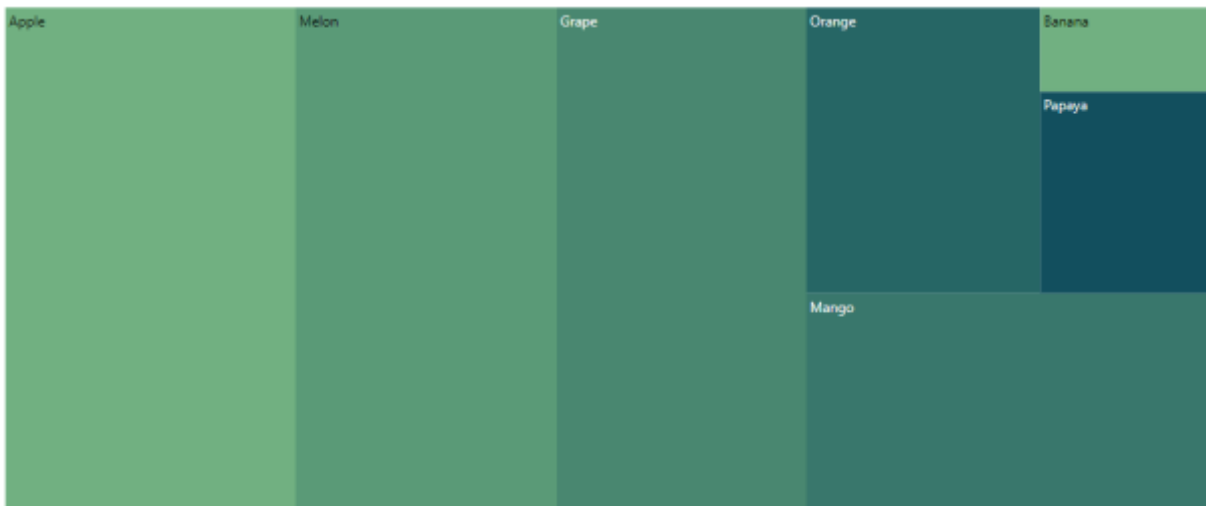


The following example shows, how to render the TreeMap in the RTL direction with [BottomLeftToTopRight](#).

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap;
<SfTreeMap DataSource="Fruits" TValue="Fruit" WeightValuePath="Count"
Palette='new string[] {"#71B081", "#5A9A77", "#498770", "#39776C",
"#266665", "#124F5E"}' RenderDirection="RenderingMode.BottomLeftTopRight">
<TreeMapLeafItemSettings LabelPath="Name">
</TreeMapLeafItemSettings>
<TreeMapTooltipSettings Visible=true Format="{Count} : {Name}">
</TreeMapTooltipSettings>
</SfTreeMap>
```

Refer to the [code block](#) to know about the property value of **Fruits**.



Leaf Item in Blazor TreeMap Component

A leaf item defines a visualized data element and does not contain child nodes but contains parent node if the levels are specified in the TreeMap.

Customization

The following properties are available to customize the leaf item in the [TreeMapLeafItemSettings](#).

- [LabelPath](#) - Represents the item name, which is available in the data source.
- [Fill](#) - Specifies the fill color for the leaf items.
- [Opacity](#) - Specifies the opacity of leaf item fill color.
- [ShowLabels](#) - Specifies the visibility of the leaf item label.
- [Padding](#) - Specifies the padding of leaf items
- [LabelTemplate](#) - Specifies the template of leaf item label and the position of the template to be customized using the [TemplatePosition](#) property.
- [TreeMapLeafLabelStyle](#) - To customize the label color, opacity, font size, font family, font weight and font style.
- [TreeMapLeafBorder](#) - Specifies leaf item border color and width.

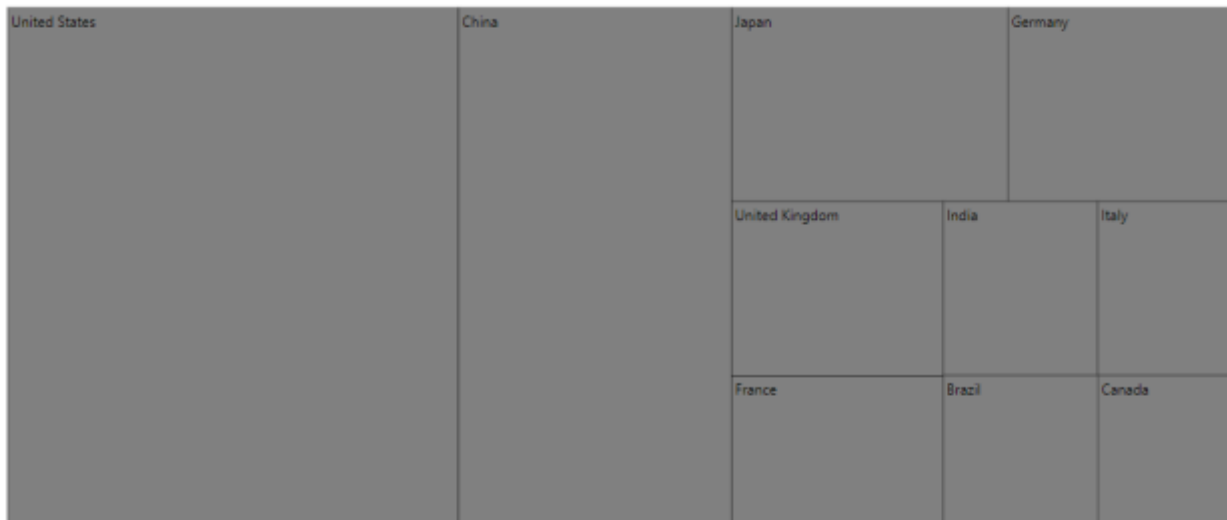
In the following example, the name of the property **CountryName** from data source is mapped to [LabelPath](#) property, so corresponding country name will be displayed in the label.

ASPX-CS

```

@using Syncfusion.Blazor.TreeMap
<SfTreeMap DataSource="GrowthReports" TValue="GDPReport"
WeightValuePath="GDP">
  <TreeMapLeafItemSettings LabelPath="CountryName">
    <TreeMapLeafLabelStyle Color="#000000"></TreeMapLeafLabelStyle>
    <TreeMapLeafBorder Color="#000000" Width="0.5"></TreeMapLeafBorder>
  </TreeMapLeafItemSettings>
</SfTreeMap>
@code{
public class GDPReport
{
public string CountryName { get; set; }
public double GDP { get; set; }
public double Percentage { get; set; }
public int Rank { get; set; }
};
public List<GDPReport> GrowthReports = new List<GDPReport> {
new GDPReport {CountryName="United States", GDP=17946, Percentage=11.08,
Rank=1},
new GDPReport {CountryName="China", GDP=10866, Percentage= 28.42, Rank=2},
new GDPReport {CountryName="Japan", GDP=4123, Percentage=-30.78, Rank=3},
new GDPReport {CountryName="Germany", GDP=3355, Percentage=-5.19, Rank=4},
new GDPReport {CountryName="United Kingdom", GDP=2848, Percentage=8.28,
Rank=5},
new GDPReport {CountryName="France", GDP=2421, Percentage=-9.69, Rank=6},
new GDPReport {CountryName="India", GDP=2073, Percentage=13.65, Rank=7},
new GDPReport {CountryName="Italy", GDP=1814, Percentage=-12.45, Rank=8},
new GDPReport {CountryName="Brazil", GDP=1774, Percentage=-27.88, Rank=9},
new GDPReport {CountryName="Canada", GDP=1550, Percentage=-15.02, Rank=10}
};
}

```

**Label position and format**

Positioning the leaf item label using the [LabelPosition](#) property and the text format can be customized by specifying data source properties name in the [LabelFormat](#) property.

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
<SfTreeMap DataSource="GrowthReports" TValue="GDPReport"
WeightValuePath="GDP">
  <TreeMapLeafItemSettings LabelPath="CountryName"
LabelPosition="LabelPosition.TopCenter"
LabelFormat="{CountryName}<br>${GDP} Trillion<br>({Percentage} %)">
  <TreeMapLeafLabelStyle Color="#000000"></TreeMapLeafLabelStyle>
  <TreeMapLeafBorder Color="#000000" Width="0.5"></TreeMapLeafBorder>
</TreeMapLeafItemSettings>
</SfTreeMap>
```

Refer to the [code block](#) to know about the property value of the **GrowthReports**.



Gap between the leaf items

The [Gap](#) property is used to separate an item from another item. Each item rectangle is split into equal space with specified gap.

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
<SfTreeMap DataSource="GrowthReports" TValue="GDPReport"
WeightValuePath="GDP">
  <TreeMapLeafItemSettings LabelPath="CountryName" Gap="20">
  </TreeMapLeafItemSettings>
</SfTreeMap>
```

Refer to the [code block](#) to know about the property value of the **GrowthReports**.



Levels in Blazor TreeMap Component

TreeMap supports **n** number of levels and each level is separated by using the [GroupPath](#) property.

Group path

The [GroupPath](#) property is used to separate each level of the TreeMap by specifying the property from the data source.

In the following example, three levels are added and each level is configured using the [GroupPath](#) property.

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
<SfTreeMap WeightValuePath="EmployeeCount" TValue="Employee"
DataSource="Employees" Palette='new string[] { "#f44336", "#29b6f6",
"#ab47bc", "#ffc107", "#5c6bc0", "#009688"}'>
<TreeMapLevels>
<TreeMapLevel GroupPath="Country">
<TreeMapLevelBorder Color="black" Width="0.5">
</TreeMapLevelBorder>
</TreeMapLevel>
<TreeMapLevel GroupPath="JobDescription">
<TreeMapLevelBorder Color="black" Width="0.5">
</TreeMapLevelBorder>
</TreeMapLevel>
<TreeMapLevel GroupPath="JobGroup">
<TreeMapLevelBorder Color="black" Width="0.5">
</TreeMapLevelBorder>
</TreeMapLevel>
</TreeMapLevels>
</SfTreeMap>
@code{
public class Employee
{
public string Country { get; set; }
public string JobDescription { get; set; }
public string JobGroup { get; set; }
public int EmployeeCount { get; set; }
};
```

```

public List<Employee> Employees = new List<Employee> {
    new Employee { Country= "USA", JobDescription= "Sales", JobGroup=
    "Executive", EmployeeCount= 20 },
    new Employee { Country= "USA", JobDescription= "Sales", JobGroup= "Analyst",
    EmployeeCount= 30 },
    new Employee { Country= "USA", JobDescription= "Marketing", EmployeeCount=
    40 },
    new Employee { Country= "USA", JobDescription= "Management", EmployeeCount=
    80 },
    new Employee { Country= "India", JobDescription= "Technical", JobGroup=
    "Testers", EmployeeCount= 100 },
    new Employee { Country= "India", JobDescription= "HR Executives",
    EmployeeCount= 30 },
    new Employee { Country= "India", JobDescription= "Accounts", EmployeeCount=
    40 },
    new Employee { Country= "UK", JobDescription= "Technical", JobGroup=
    "Testers", EmployeeCount= 30 },
    new Employee { Country= "UK", JobDescription= "HR Executives",
    EmployeeCount= 50 },
    new Employee { Country= "UK", JobDescription= "Accounts", EmployeeCount= 60
    }
};
}

```



Gap between groups

The [GroupGap](#) property is used to separate an item from each group or another item to differentiate the levels mentioned in the TreeMap.

ASPX-CS

```

@using Syncfusion.Blazor.TreeMap
<SfTreeMap WeightValuePath="EmployeeCount" TValue="Employee"
DataSource="Employees" Palette='new string[] { "#f44336", "#29b6f6",
"#ab47bc", "#ffc107", "#5c6bc0", "#009688"}'>
  <TreeMapLevels>
    <TreeMapLevel GroupPath="Country" GroupGap="10">
      <TreeMapLevelBorder Color="black" Width="0.5">
        </TreeMapLevelBorder>

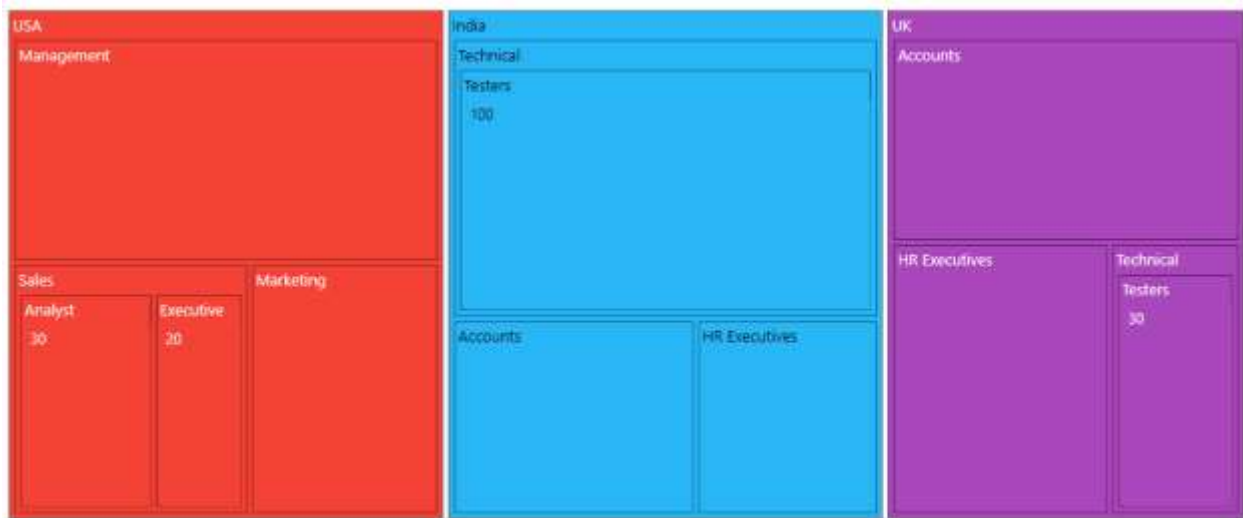
```

```

</TreeMapLevel>
<TreeMapLevel GroupPath="JobDescription" GroupGap="10">
<TreeMapLevelBorder Color="black" Width="0.5">
</TreeMapLevelBorder>
</TreeMapLevel>
<TreeMapLevel GroupPath="JobGroup" GroupGap="10">
<TreeMapLevelBorder Color="black" Width="0.5">
</TreeMapLevelBorder>
</TreeMapLevel>
</TreeMapLevels>
</SfTreeMap>

```

Refer to the [code block](#) to know about the property value of the **Employees**.



Header height and style

Customize the font color, family, weight, opacity and size using the [TreeMapHeaderStyle](#). Based on the font settings, the header height is given using the [HeaderHeight](#) property in the [TreeMapLevel](#).

ASPX-CS

```

@using Syncfusion.Blazor.TreeMap
<SfTreeMap WeightValuePath="EmployeeCount" TValue="Employee"
DataSource="Employees" Palette='new string[] { "#f44336", "#29b6f6",
"#ab47bc", "#ffc107", "#5c6bc0", "#009688"}'>
<TreeMapLevels>
<TreeMapLevel GroupPath="Country" HeaderHeight="35">
<TreeMapHeaderStyle Size="15px"></TreeMapHeaderStyle>
<TreeMapLevelBorder Color="black" Width="0.5">
</TreeMapLevelBorder>
</TreeMapLevel>
<TreeMapLevel GroupPath="JobDescription" HeaderHeight="45">
<TreeMapHeaderStyle Size="15px"></TreeMapHeaderStyle>
<TreeMapLevelBorder Color="black" Width="0.5">
</TreeMapLevelBorder>
</TreeMapLevel>
<TreeMapLevel GroupPath="JobGroup" HeaderHeight="40">
<TreeMapHeaderStyle Size="15px"></TreeMapHeaderStyle>
<TreeMapLevelBorder Color="black" Width="0.5">

```

```

</TreeMapLevelBorder>
</TreeMapLevel>
</TreeMapLevels>
</SfTreeMap>

```

Refer to the [code block](#) to know about the property value of **Employees**.



Customization

The following properties are available to customize the header content in the [TreeMapLevel](#).

- [HeaderFormat](#) - Represents the header name, which is available in the data source.
- [ShowHeader](#) - Specifies to visibility of the header.
- [HeaderTemplate](#) - Specifies the template of the header and the position of the template to be customized using the [TemplatePosition](#) property.
- [TreeMapLevelBorder](#) - Specifies TreeMap level border color and width.
- [HeaderAlignment](#) - Align the header to [Near](#), [Center](#) and [Far](#).

ASPX-CS

```

@using Syncfusion.Blazor.TreeMap
<SfTreeMap WeightValuePath="EmployeeCount" TValue="Employee"
DataSource="Employees" Palette='new string[] { "#f44336", "#29b6f6",
"#ab47bc", "#ffc107", "#5c6bc0", "#009688"}'>
<TreeMapLevels>
<TreeMapLevel GroupPath="Country" HeaderFormat="{Country}-{EmployeeCount}"
HeaderAlignment="Alignment.Center">
<TreeMapLevelBorder Color="black" Width="0.5">
</TreeMapLevelBorder>
</TreeMapLevel>
<TreeMapLevel GroupPath="JobDescription" GroupGap="10"
HeaderFormat="{JobDescription}-{EmployeeCount}"
HeaderAlignment="Alignment.Far">
<TreeMapLevelBorder Color="black" Width="0.5">
</TreeMapLevelBorder>
</TreeMapLevel>

```

```

<TreeMapLevel GroupPath="JobGroup" GroupGap="10" HeaderFormat="{JobGroup}-
${EmployeeCount}" HeaderAlignment="Alignment.Near">
<TreeMapLevelBorder Color="black" Width="0.5">
</TreeMapLevelBorder>
</TreeMapLevel>
</TreeMapLevels>
</SfTreeMap>

```

Refer to the [code block](#) to know about the property value of Employees.



Color Mapping in Blazor TreeMap Component

Color mapping is used to customize the color for each group or item based on the specified types. The following options are available to customize the group and leaf items in the TreeMap component.

Range color mapping

ASPX-CS

```

@using Syncfusion.Blazor.TreeMap
<SfTreeMap WeightValuePath="Count" TValue="Fruit" DataSource="Fruits"
RangeColorValuePath="Count">
<TreeMapLeafItemSettings LabelPath="FruitName">
<TreeMapLeafColorMappings>
<TreeMapLeafColorMapping StartRange="500" EndRange="3000" Color='new
string[] { "Orange" }'></TreeMapLeafColorMapping>
<TreeMapLeafColorMapping StartRange="3000" EndRange="5000" Color='new
string[] { "Green" }'></TreeMapLeafColorMapping>
</TreeMapLeafColorMappings>
</TreeMapLeafItemSettings>
</SfTreeMap>
@code {
public class Fruit
{
public string FruitName { get; set; }
public double Count { get; set; }
};
public List<Fruit> Fruits = new List<Fruit> {
new Fruit { FruitName="Apple", Count=5000 },
new Fruit { FruitName="Mango", Count=3000 },

```



```

new Fruit { FruitName="Orange", Count=2300 },
new Fruit { FruitName="Banana", Count=500 },
new Fruit { FruitName="Grape", Count=4300 },
new Fruit { FruitName="Papaya", Count=1200 },
new Fruit { FruitName="Melon", Count=4500 }
};
}

```



Equal color mapping

Equal color mapping is used to fill colors to each item by specifying equal value present in the data source, that can be specified in the [EqualColorValuePath](#) property.

ASPX-CS

```

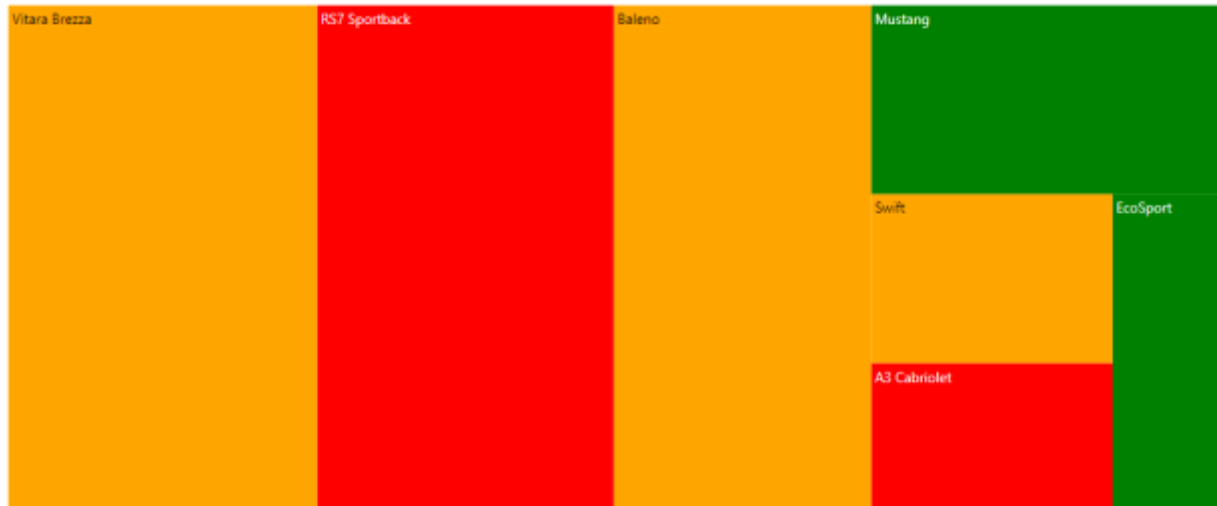
@using Syncfusion.Blazor.TreeMap
<SfTreeMap WeightValuePath="Count" TValue="Car" DataSource="Cars"
EqualColorValuePath="Brand">
  <TreeMapLeafItemSettings LabelPath="Name">
    <TreeMapLeafColorMappings>
      <TreeMapLeafColorMapping Value="Ford" Color='new string[] { "green"
}'></TreeMapLeafColorMapping>
      <TreeMapLeafColorMapping Value="Audi" Color='new string[] { "red"
}'></TreeMapLeafColorMapping>
      <TreeMapLeafColorMapping Value="Maruti" Color='new string[] {
"orange"}'></TreeMapLeafColorMapping>
    </TreeMapLeafColorMappings>
  </TreeMapLeafItemSettings>
</SfTreeMap>
@code {
public class Car
{
public string Name { get; set; }
public int Count { get; set; }
public string Brand { get; set; }
};
public List<Car> Cars = new List<Car> {
new Car { Name="Mustang", Brand="Ford", Count=232 },
new Car { Name="EcoSport", Brand="Ford", Count=121 },

```

```

new Car { Name="Swift", Brand="Maruti", Count=143 },
new Car { Name="Baleno", Brand="Maruti", Count=454 },
new Car { Name="Vitara Brezza", Brand="Maruti", Count=545 },
new Car { Name="A3 Cabriolet", Brand="Audi", Count=123 },
new Car { Name="RS7 Sportback", Brand="Audi", Count=523 }
};
}

```



Desaturation color mapping

Desaturation color mapping is used to apply colors to the items based on the [MinOpacity](#) and the [MaxOpacity](#) properties in the [TreeMapLeafColorMapping](#).

ASPX-CS

```

@using Syncfusion.Blazor.TreeMap
<SfTreeMap WeightValuePath="Count" TValue="Fruit" DataSource="Fruits"
RangeColorValuePath="Count">
<TreeMapLeafItemSettings LabelPath="FruitName">
<TreeMapLeafColorMappings>
<TreeMapLeafColorMapping StartRange="500" EndRange="3000" MinOpacity="0.2"
MaxOpacity="0.5" Color='new string[] { "Orange"}'></TreeMapLeafColorMapping>
<TreeMapLeafColorMapping StartRange="3000" EndRange="5000" MinOpacity="0.5"
MaxOpacity="0.8" Color='new string[] { "Green"}'></TreeMapLeafColorMapping>
</TreeMapLeafColorMappings>
</TreeMapLeafItemSettings>
</SfTreeMap>

```

Refer to the [code block](#) to know about the property value of the **Fruits**.



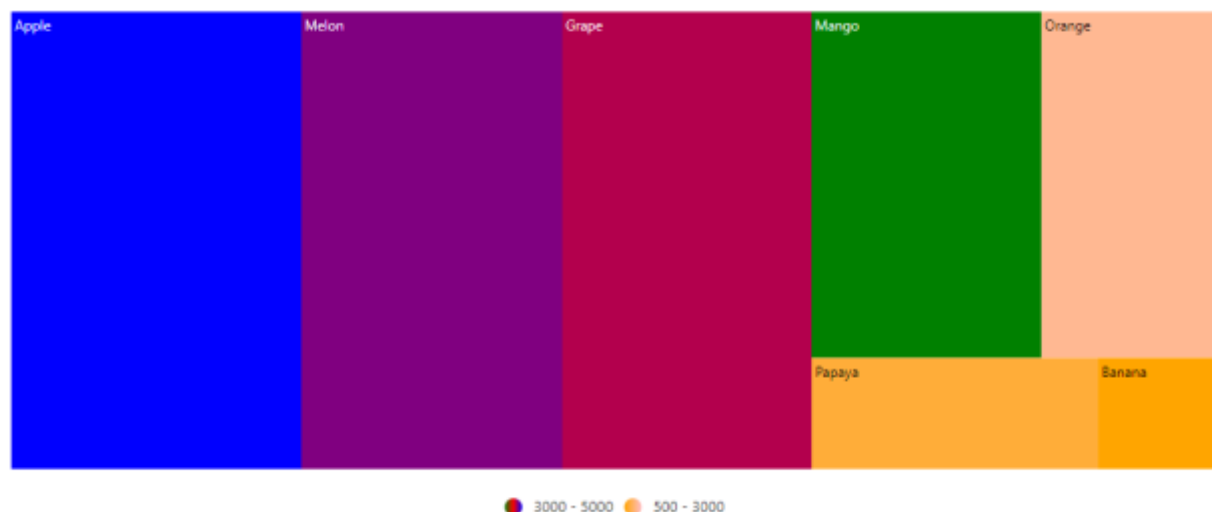
Desaturation with multiple colors

Multiple colors are used to provide gradient effect to the TreeMap items based on the [TreeMapLeafColorMapping](#) ranges and specify the **n** number of colors in the [Color](#) property.

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
<SfTreeMap WeightValuePath="Count" TValue="Fruit" DataSource="Fruits"
RangeColorValuePath="Count">
  <TreeMapLeafItemSettings LabelPath="FruitName">
    <TreeMapLeafColorMappings>
      <TreeMapLeafColorMapping StartRange="500" EndRange="3000" Color='new
string[] { "orange", "pink" }'></TreeMapLeafColorMapping>
      <TreeMapLeafColorMapping StartRange="3000" EndRange="5000" Color='new
string[] { "green", "red", "blue" }'></TreeMapLeafColorMapping>
    </TreeMapLeafColorMappings>
  </TreeMapLeafItemSettings>
</SfTreeMap>
```

Refer to the [code block](#) to know about the property value of **Fruits**.



Palette color mapping

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
<SfTreeMap WeightValuePath="Count" TValue="Car" DataSource="Cars"
  Palette='new string[] { "red", "green" }'>
  <TreeMapLeafItemSettings LabelPath="Name">
  </TreeMapLeafItemSettings>
</SfTreeMap>
```

Refer to the [code block](#) to know about the property value of **Cars**.



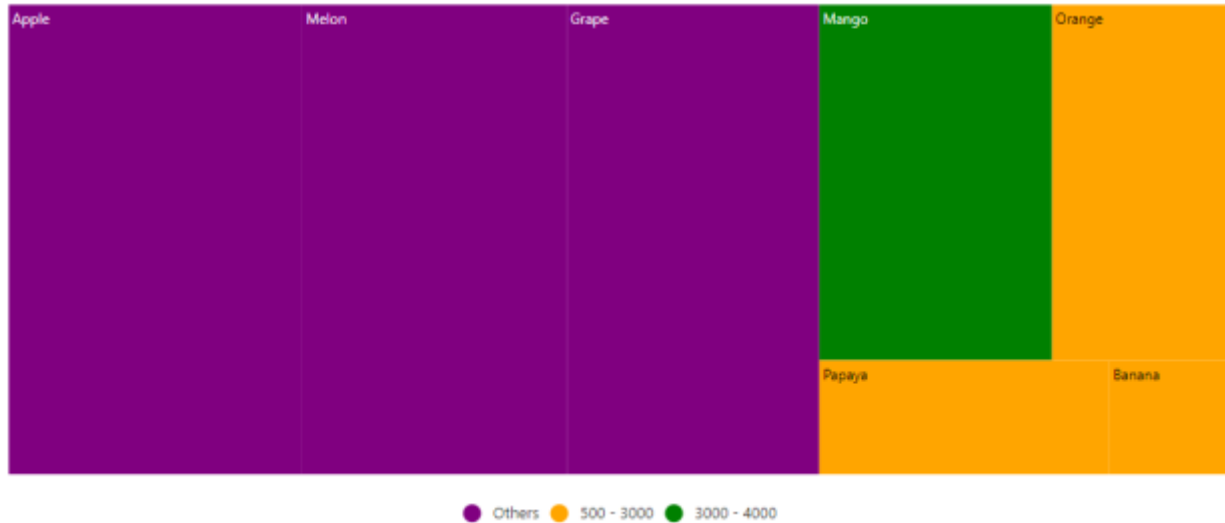
Color for items excluded from color mapping

Get the excluded ranges from data source using the color mapping and apply the specific color to those items, without specifying the [StartRange](#) and the [EndRange](#) properties.

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
<SfTreeMap WeightValuePath="Count" TValue="Fruit" DataSource="Fruits"
  RangeColorValuePath="Count">
  <TreeMapLeafItemSettings LabelPath="FruitName">
  <TreeMapLeafColorMappings>
  <TreeMapLeafColorMapping StartRange="500" EndRange="3000" Color='new
  string[] { "Orange" }'></TreeMapLeafColorMapping>
  <TreeMapLeafColorMapping StartRange="3000" EndRange="4000" Color='new
  string[] { "Green" }'></TreeMapLeafColorMapping>
  <TreeMapLeafColorMapping Color='new string[] {
  "purple" }'></TreeMapLeafColorMapping>
  </TreeMapLeafColorMappings>
  </TreeMapLeafItemSettings>
</SfTreeMap>
```

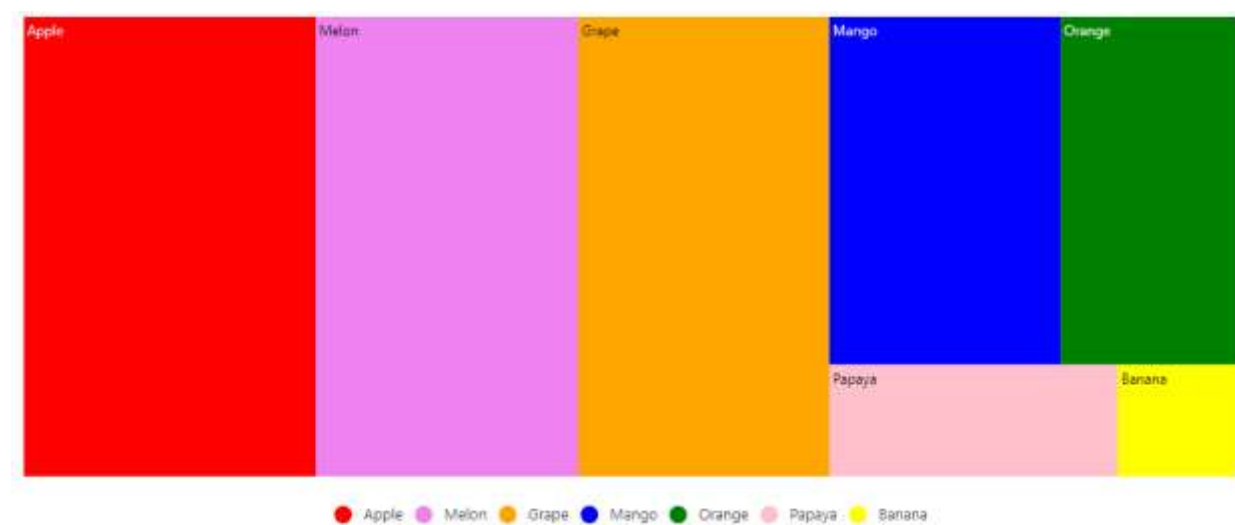
Refer to the [code block](#) to know about the property value of the **Fruits**.



Bind the colors to items from the data source

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
<SfTreeMap WeightValuePath="Count" TValue="Fruit" DataSource="Fruits"
RangeColorValuePath="Count" ColorValuePath="Color">
<TreeMapLeafItemSettings LabelPath="Name"></TreeMapLeafItemSettings>
<TreeMapLegendSettings Visible="true"></TreeMapLegendSettings>
</SfTreeMap>
@code {
public class Fruit
{
public string Name { get; set; }
public double Count { get; set; }
public string Color { get; set; }
};
public List<Fruit> Fruits = new List<Fruit> {
new Fruit { Name="Apple", Count=5000, Color = "red" },
new Fruit { Name="Mango", Count=3000, Color="blue" },
new Fruit { Name="Orange", Count=2300, Color="green" },
new Fruit { Name="Banana", Count=500 , Color="yellow"},
new Fruit { Name="Grape", Count=4300 , Color="orange"},
new Fruit { Name="Papaya",Count=1200 , Color="pink"},
new Fruit { Name="Melon", Count=4500, Color="violet" }
};
}
```



Labels in Blazor TreeMap Component

Data Labels are used to identify the name of items or groups in the TreeMap component. Data Labels will be shown by specifying the data source properties in the [LabelPath](#) of the [TreeMapLeafItemSettings](#).

Formatting labels

Customize the labels for each item using the [LabelFormat](#) property in the [TreeMapLeafItemSettings](#).

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
<SfTreeMap WeightValuePath="Count" TValue="Car" DataSource="Cars"
RangeColorValuePath="Count">
<TreeMapLeafItemSettings LabelPath="Name" LabelFormat="{Name} -
{Brand}"></TreeMapLeafItemSettings>
</SfTreeMap>
@code {
public class Car
{
public string Name { get; set; }
public string Brand { get; set; }
public int Count { get; set; }
};
public List<Car> Cars = new List<Car> {
new Car { Name="Mustang", Brand="Ford", Count=232 },
new Car { Name="EcoSport", Brand="Ford", Count=121 },
new Car { Name="Swift", Brand="Maruti", Count=143 },
new Car { Name="Baleno", Brand="Maruti", Count=454 },
new Car { Name="Vitara Brezza", Brand="Maruti", Count=545 },
new Car { Name="A3 Cabriolet", Brand="Audi", Count=123 },
new Car { Name="RS7 Sportback", Brand="Audi", Count=523 }
};
}
```



Label position

Customize the label position using the [LabelPosition](#) property by specifying any of the following locations.

- [BottomCenter](#)
- [BottomLeft](#)
- [BottomRight](#)
- [Center](#)
- [CenterLeft](#)
- [CenterRight](#)
- [TopCenter](#)
- [TopLeft](#)
- [TopRight](#)

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
<SfTreeMap WeightValuePath="Count" TValue="Car" DataSource="Cars"
RangeColorValuePath="Count">
<TreeMapLeafItemSettings LabelPath="Name" LabelFormat="{Name}-{Brand}"
LabelPosition="LabelPosition.Center" Gap="2"></TreeMapLeafItemSettings>
</SfTreeMap>
```

Refer to the [code block](#) to know about the property value of **Cars**.

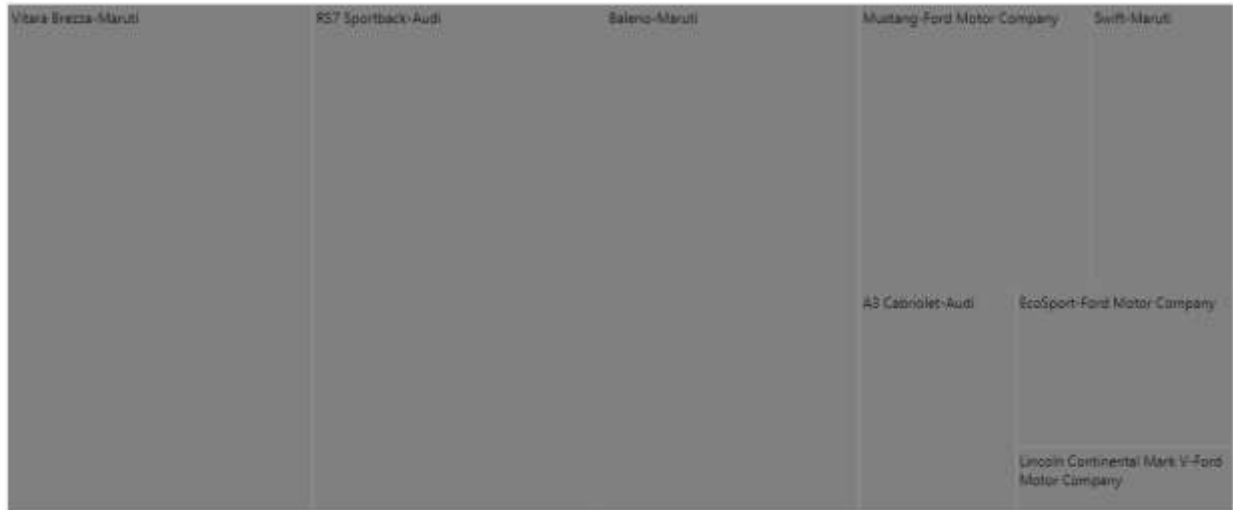


Intersect action

When the label size in each item exceeds the actual size, use the [InterSectAction](#) property in the [TreeMapLeafItemSettings](#) to customize the labels.

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
<SfTreeMap WeightValuePath="Count" TValue="Car" DataSource="Cars">
  <TreeMapLeafItemSettings LabelPath="Name" LabelFormat="{Name}-{Brand}"
    InterSectAction="LabelAlignment.WrapByWord">
  </TreeMapLeafItemSettings>
</SfTreeMap>
@code{
public class Car
{
    public string Name { get; set; }
    public string Brand { get; set; }
    public int Count { get; set; }
};
public List<Car> Cars = new List<Car> {
    new Car { Name="Mustang", Brand="Ford Motor Company", Count=232 },
    new Car { Name="Lincoln Continental Mark V", Brand="Ford Motor Company",
    Count=50},
    new Car { Name="EcoSport", Brand="Ford Motor Company", Count=121 },
    new Car { Name="Swift", Brand="Maruti", Count=143 },
    new Car { Name="Baleno", Brand="Maruti", Count=454 },
    new Car { Name="Vitar Brezza", Brand="Maruti", Count=545 },
    new Car { Name="A3 Cabriolet", Brand="Audi",Count=123 },
    new Car { Name="RS7 Sportback", Brand="Audi", Count=523 }
};
}
```

Legend in Blazor TreeMap Component

Legend is used to provide valuable information for interpreting what the TreeMap displays. The legends can be represented in various colors, shapes or other identifiers based on the data.

Types of legend

TreeMap component supports two different types of legend mode as following.

- [Default](#)
- [Interactive](#)

Default legend

In the default mode, the legends have symbols with legend labels that are used to identify the items in the TreeMap component.

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
<SfTreeMap DataSource="Cars" TValue="Car" WeightValuePath="Count"
EqualColorValuePath="Brand">
  <TreeMapLeafItemSettings LabelPath="Name">
    <TreeMapLeafColorMappings>
      <TreeMapLeafColorMapping Value="Ford" Color='new string[] { "green" }'></TreeMapLeafColorMapping>
      <TreeMapLeafColorMapping Value="Audi" Color='new string[] { "red" }'></TreeMapLeafColorMapping>
      <TreeMapLeafColorMapping Value="Maruti" Color='new string[] { "orange" }'></TreeMapLeafColorMapping>
    </TreeMapLeafColorMappings>
  </TreeMapLeafItemSettings>
  <TreeMapLegendSettings Visible="true" Position="LegendPosition.Top">
  </TreeMapLegendSettings>
</SfTreeMap>
@code {
public class Car
{
public string Name { get; set; }
public string Brand { get; set; }
}
```

```

public int Count { get; set; }
};
public List<Car> Cars = new List<Car> {
    new Car { Name="Mustang", Brand="Ford", Count=232},
    new Car { Name="EcoSport", Brand="Ford", Count=121},
    new Car { Name="Swift", Brand="Maruti", Count=143},
    new Car { Name="Baleno", Brand="Maruti", Count=454},
    new Car { Name="Vitara Brezza", Brand="Maruti", Count=545},
    new Car { Name="A3 Cabriolet", Brand="Audi", Count=123}
};
}

```



Interactive legend

The legends can be made interactive with an arrow mark that indicates exact range color in the legend when the mouse hovers on the TreeMap item. Enable this option by setting the [Mode](#) property in the [TreeMapLegendSettings](#) to [Interactive](#).

ASPX-CS

```

@using Syncfusion.Blazor.TreeMap
<SfTreeMap DataSource="Cars" TValue="Car" WeightValuePath="Count"
EqualColorValuePath="Brand">
    <TreeMapLeafItemSettings LabelPath="Name">
        <TreeMapLeafColorMappings>
            <TreeMapLeafColorMapping Value="Ford" Color='new string[] { "green" }'></TreeMapLeafColorMapping>
            <TreeMapLeafColorMapping Value="Audi" Color='new string[] { "red" }'></TreeMapLeafColorMapping>
            <TreeMapLeafColorMapping Value="Maruti" Color='new string[] { "orange" }'></TreeMapLeafColorMapping>
        </TreeMapLeafColorMappings>
    </TreeMapLeafItemSettings>
    <TreeMapLegendSettings Visible="true" Position="LegendPosition.Top"
Mode="LegendMode.Interactive">
    </TreeMapLegendSettings>
</SfTreeMap>

```

Refer to the [code block](#) to know about the property value of Cars.



Position and Alignment

The legend position is used to place the legend in various positions. Based on the legend position, the legend item will be aligned. For example, if the position is [Top](#) or [Bottom](#), the legend items are placed by rows. If the position is [Left](#) or [Right](#), the legend items are placed by columns.

The following options are available to customize the legend position:

- [Top](#)
- [Bottom](#)
- [Left](#)
- [Right](#)
- [Float](#)
- [Auto](#)

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
<SfTreeMap DataSource="Fruits" TValue="Fruit" WeightValuePath="Count"
RangeColorValuePath="Count">
<TreeMapLeafItemSettings LabelPath="FruitName">
<TreeMapLeafColorMappings>
<TreeMapLeafColorMapping From="500" To="3000" Color='new string[] { "Orange"
}'></TreeMapLeafColorMapping>
<TreeMapLeafColorMapping From="3000" To="5000" Color='new string[] { "Green"
}'></TreeMapLeafColorMapping>
</TreeMapLeafColorMappings>
</TreeMapLeafItemSettings>
<TreeMapLegendSettings Visible="true"
Position="LegendPosition.Top"></TreeMapLegendSettings>
</SfTreeMap>
@code {
public class Fruit
{
public string FruitName { get; set; }
public double Count { get; set; }
}
```

```
};
public List<Fruit> Fruits = new List<Fruit> {
    new Fruit { FruitName="Apple", Count=5000 },
    new Fruit { FruitName="Mango", Count=3000 },
    new Fruit { FruitName="Orange", Count=2300 },
    new Fruit { FruitName="Banana", Count=500 },
    new Fruit { FruitName="Grape", Count=4300 },
    new Fruit { FruitName="Papaya", Count=1200 },
    new Fruit { FruitName="Melon", Count=4500 }
};
}
```



The legend alignment is used to align the legend items in a specific location. The following options are available to customize the legend alignment:

- [Near](#)
- [Center](#)
- [Far](#)

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
<SfTreeMap DataSource="Fruits" TValue="Fruit" WeightValuePath="Count"
RangeColorValuePath="Count">
    <TreeMapLeafItemSettings LabelPath="FruitName">
    <TreeMapLeafColorMappings>
    <TreeMapLeafColorMapping From="500" To="3000" Color='new
string[] {"Orange"}'></TreeMapLeafColorMapping>
    <TreeMapLeafColorMapping From="3000" To="5000" Color='new
string[] {"Green"}'></TreeMapLeafColorMapping>
    </TreeMapLeafColorMappings>
    </TreeMapLeafItemSettings>
    <TreeMapLegendSettings Visible="true" Alignment="Alignment.Far">
    </TreeMapLegendSettings>
</SfTreeMap>
```

Refer to the [code block](#) to know about the property value of **Fruits**.



Legend size

Customize the legend size by modifying the [Height](#) and the [Width](#) properties in the [TreeMapLegendSettings](#). It accepts values in both percentage and pixel.

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
<SfTreeMap DataSource="Cars" WeightValuePath="Count"
EqualColorValuePath="Brand">
<TreeMapLeafItemSettings LabelPath="Name">
<TreeMapLeafColorMappings>
<TreeMapLeafColorMapping Value="Ford" Color='new string[] {
"green"}'></TreeMapLeafColorMapping>
<TreeMapLeafColorMapping Value="Audi" Color='new string[] { "red"
}'></TreeMapLeafColorMapping>
<TreeMapLeafColorMapping Value="Maruti" Color='new string[] {
"orange"}'></TreeMapLeafColorMapping>
</TreeMapLeafColorMappings>
</TreeMapLeafItemSettings>
<TreeMapLegendSettings Visible="true" Height="50px" Width="200px"
Position="LegendPosition.Top">
</TreeMapLegendSettings>
</SfTreeMap>
@code {
public class Car
{
public string Name { get; set; }
public string Brand { get; set; }
public int Count { get; set; }
};
public List<Car> Cars = new List<Car> {
new Car { Name="Mustang", Brand="Ford", Count=232},
new Car { Name="EcoSport", Brand="Ford", Count=121},
new Car { Name="Swift", Brand="Maruti", Count=143},
new Car { Name="Baleno", Brand="Maruti", Count=454},
new Car { Name="Vitara Brezza", Brand="Maruti", Count=545},
new Car { Name="A3 Cabriolet", Brand="Audi", Count=123},
```

```
new Car { Name="RS7 Sportback", Brand="Audi", Count=523 }
};
}
```



Legend with paging support

TreeMap supports legend paging, if the legend items cannot be placed within the provided [Height](#) and [Width](#) of the legend.

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
<SfTreeMap DataSource="Cars" TValue="Car" WeightValuePath="Count"
EqualColorValuePath="Brand">
<TreeMapLeafItemSettings LabelPath="Name">
<TreeMapLeafColorMappings>
<TreeMapLeafColorMapping Value="Ford" Color='new string[] { "green"
}'></TreeMapLeafColorMapping>
<TreeMapLeafColorMapping Value="Audi" Color='new string[] { "red"
}'></TreeMapLeafColorMapping>
<TreeMapLeafColorMapping Value="Maruti" Color='new string[] { "orange"
}'></TreeMapLeafColorMapping>
</TreeMapLeafColorMappings>
</TreeMapLeafItemSettings>
<TreeMapLegendSettings Visible="true" Height="50px" Width="100px"
Position="LegendPosition.Top">
<TreeMapLegendBorder Width="1"></TreeMapLegendBorder>
</TreeMapLegendSettings>
</SfTreeMap>
@code {
public class Car
{
public string Name { get; set; }
public string Brand { get; set; }
public int Count { get; set; }
};
public List<Car> Cars = new List<Car> {
new Car { Name="Mustang", Brand="Ford", Count=232},
```

```

new Car { Name="EcoSport", Brand="Ford", Count=121},
new Car { Name="Swift", Brand="Maruti", Count=143},
new Car { Name="Baleno", Brand="Maruti", Count=454},
new Car { Name="Vitara Brezza", Brand="Maruti", Count=545},
new Car { Name="A3 Cabriolet", Brand="Audi", Count=123},
new Car { Name="RS7 Sportback", Brand="Audi", Count=523 }
};
}

```



Excluded legend items from the color mapping

Based on the mapping ranges in the data source, get the excluded ranges from the color mapping, and show the legend with the excluded range values that are bound to the specific legend.

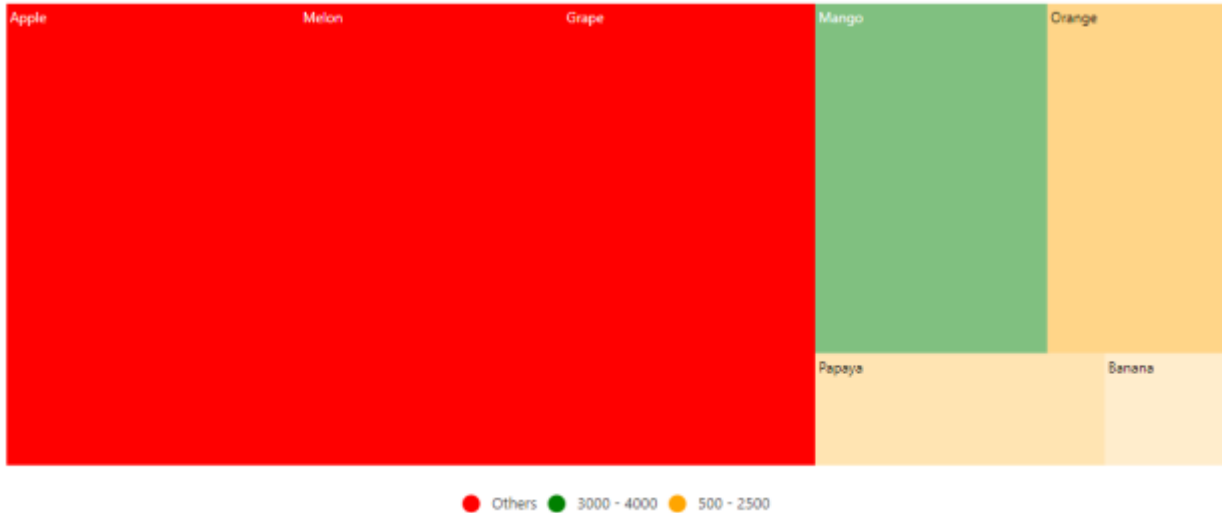
ASPX-CS

```

@using Syncfusion.Blazor.TreeMap
<SfTreeMap DataSource="Fruits" TValue="Fruit" WeightValuePath="Count"
RangeColorValuePath="Count">
<TreeMapLeafItemSettings LabelPath="FruitName">
<TreeMapLeafColorMappings>
<TreeMapLeafColorMapping From="500" To="3000" Color='new string[] { "Orange"
}'></TreeMapLeafColorMapping>
<TreeMapLeafColorMapping From="3000" To="4000" Color='new string[] { "Green"
}'></TreeMapLeafColorMapping>
<TreeMapLeafColorMapping Color='new string[] { "red"
}'></TreeMapLeafColorMapping>
</TreeMapLeafColorMappings>
</TreeMapLeafItemSettings>
<TreeMapLegendSettings Visible="true">
</TreeMapLegendSettings>
</SfTreeMap>
@code {
public class Fruit
{
public string FruitName { get; set; }
public double Count { get; set; }
};
public List<Fruit> Fruits = new List<Fruit> {
new Fruit { FruitName="Apple", Count=5000 },
new Fruit { FruitName="Mango", Count=3000 },
new Fruit { FruitName="Orange", Count=2300 },
new Fruit { FruitName="Banana", Count=500 },
new Fruit { FruitName="Grape", Count=4300 },

```

```
new Fruit { FruitName="Papaya", Count=1200 },
new Fruit { FruitName="Melon", Count=4500 }
};
}
```



Hide desired legend items

To enable or disable the desired legend item for each color mapping, set the [ShowLegend](#) property to **true** in the [TreeMapLeafColorMappings](#).

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
<SfTreeMap DataSource="Fruits" TValue="Fruit" WeightValuePath="Count"
RangeColorValuePath="Count">
<TreeMapLeafItemSettings LabelPath="FruitName">
<TreeMapLeafColorMappings>
<TreeMapLeafColorMapping From="500" To="3000" Color='new string[] { "Orange"
}'></TreeMapLeafColorMapping>
<TreeMapLeafColorMapping From="3000" To="4000" Color='new string[] { "Green"
}' ShowLegend="false"></TreeMapLeafColorMapping>
<TreeMapLeafColorMapping Color='new string[] { "red"
}'></TreeMapLeafColorMapping>
</TreeMapLeafColorMappings>
</TreeMapLeafItemSettings>
<TreeMapLegendSettings Visible="true">
</TreeMapLegendSettings>
</SfTreeMap>
@code {
public class Fruit
{
public string FruitName { get; set; }
public double Count { get; set; }
};
public List<Fruit> Fruits = new List<Fruit> {
new Fruit { FruitName="Apple", Count=5000 },
new Fruit { FruitName="Mango", Count=3000 },
new Fruit { FruitName="Orange", Count=2300 },
new Fruit { FruitName="Banana", Count=500 },
```



```
new Fruit { FruitName="Grape", Count=4300 },
new Fruit { FruitName="Papaya", Count=1200 },
new Fruit { FruitName="Melon", Count=4500 }
};
}
```

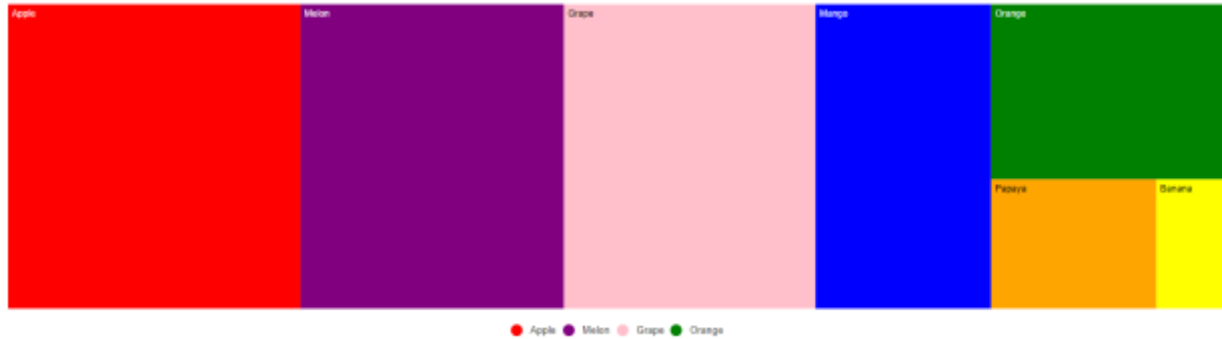


Hide legend items based on the data source value

To enable or disable the legend visibility for each item through the data source, bind the appropriate data source property name to [ShowLegendPath](#) property in the [TreeMapLegendSettings](#).

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
<SfTreeMap DataSource="Fruits" TValue="Fruit" WeightValuePath="Count"
ColorValuePath="Color">
<TreeMapLeafItemSettings LabelPath="Name">
</TreeMapLeafItemSettings>
<TreeMapLegendSettings Visible="true" ShowLegendPath="Visibility">
</TreeMapLegendSettings>
</SfTreeMap>
@code{
public class Fruit
{
public string Name { get; set; }
public int Count { get; set; }
public bool Visibility { get; set; }
public string Color { get; set; }
};
public List<Fruit> Fruits = new List<Fruit> {
new Fruit { Name="Apple", Count=5000, Visibility= true , Color="red" },
new Fruit { Name="Mango", Count=3000, Visibility= false , Color="blue"},
new Fruit { Name="Orange", Count=2300, Visibility= true , Color="green" },
new Fruit { Name="Banana", Count=500, Visibility= false , Color = "yellow"},
new Fruit { Name="Grape", Count=4300, Visibility= true , Color="pink"},
new Fruit { Name="Papaya", Count=1200, Visibility= false, Color="orange" },
new Fruit { Name="Melon", Count=4500, Visibility= true , Color="purple"}
};
}
```

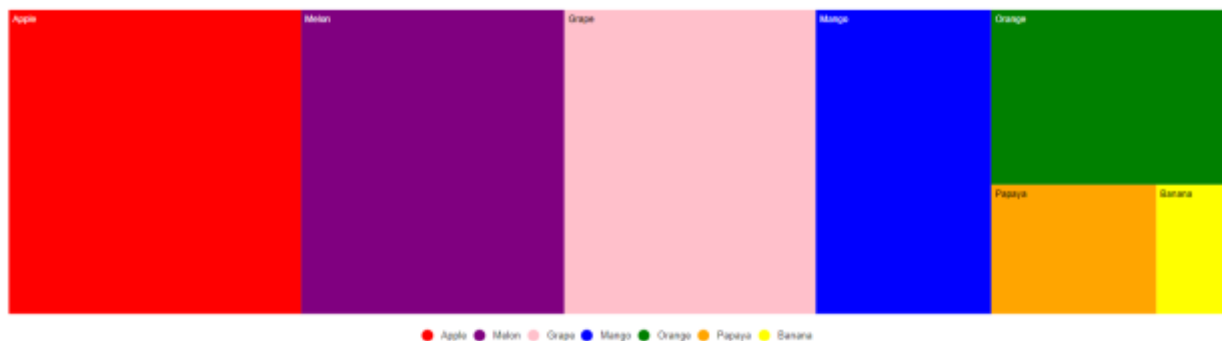


Bind legend item text from the data source

To show the legend item text from the data source, bind the property name from the data source to the [ValuePath](#) property in the [TreeMapLegendSettings](#).

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
<SfTreeMap DataSource="Fruits" TValue="Fruit" WeightValuePath="Count"
ColorValuePath="Color">
  <TreeMapLeafItemSettings LabelPath="Name">
  </TreeMapLeafItemSettings>
  <TreeMapLegendSettings Visible="true" ValuePath="Name">
  </TreeMapLegendSettings>
</SfTreeMap>
@code{
public class Fruit
{
    public string Name { get; set; }
    public int Count { get; set; }
    public string Color { get; set; }
};
public List<Fruit> Fruits = new List<Fruit> {
    new Fruit { Name="Apple", Count=5000, Color="red" },
    new Fruit { Name="Mango", Count=3000, Color="blue"},
    new Fruit { Name="Orange", Count=2300, Color="green" },
    new Fruit { Name="Banana", Count=500, Color = "yellow"},
    new Fruit { Name="Grape", Count=4300, Color="pink"},
    new Fruit { Name="Papaya", Count=1200, Color="orange" },
    new Fruit { Name="Melon", Count=4500, Color="purple"}
};
}
```

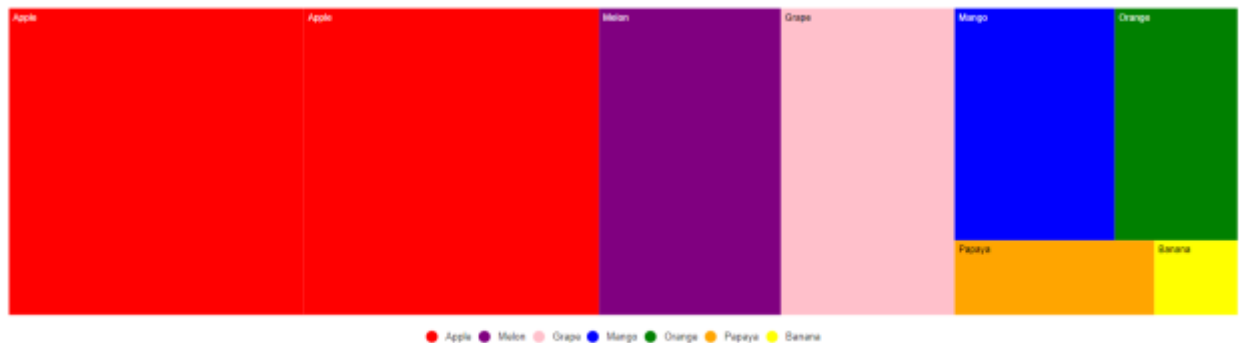


Hide duplicate legend items

To enable or disable the duplicate legend items, set the [RemoveDuplicateLegend](#) property to **true** in the [TreeMapLegendSettings](#).

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
<SfTreeMap DataSource="Fruits" TValue="Fruit" WeightValuePath="Count"
ColorValuePath="Color">
<TreeMapLeafItemSettings LabelPath="Name">
</TreeMapLeafItemSettings>
<TreeMapLegendSettings Visible="true" ValuePath="Name"
RemoveDuplicateLegend="true">
</TreeMapLegendSettings>
</SfTreeMap>
@code{
public class Fruit
{
public string Name { get; set; }
public int Count { get; set; }
public string Color { get; set; }
};
public List<Fruit> Fruits = new List<Fruit> {
new Fruit { Name="Apple", Count=5000, Color="red" },
new Fruit { Name="Apple", Count=2300, Color="yellow" },
new Fruit { Name="Mango", Count=3000, Color="blue" },
new Fruit { Name="Orange", Count=2300, Color="green" },
new Fruit { Name="Banana", Count=500, Color = "yellow"},
new Fruit { Name="Grape", Count=4300, Color="pink" },
new Fruit { Name="Papaya", Count=1200, Color="orange" },
new Fruit { Name="Melon", Count=4500, Color="purple" }
};
}
```



Positioning based on size

Use a responsive legend that switches positions between the right and the bottom based on the available height and width. To enable the responsive legend, set the [Position](#) property to [Auto](#) in the [TreeMapLegendSettings](#) and the legend position is changed based on the available height and width.

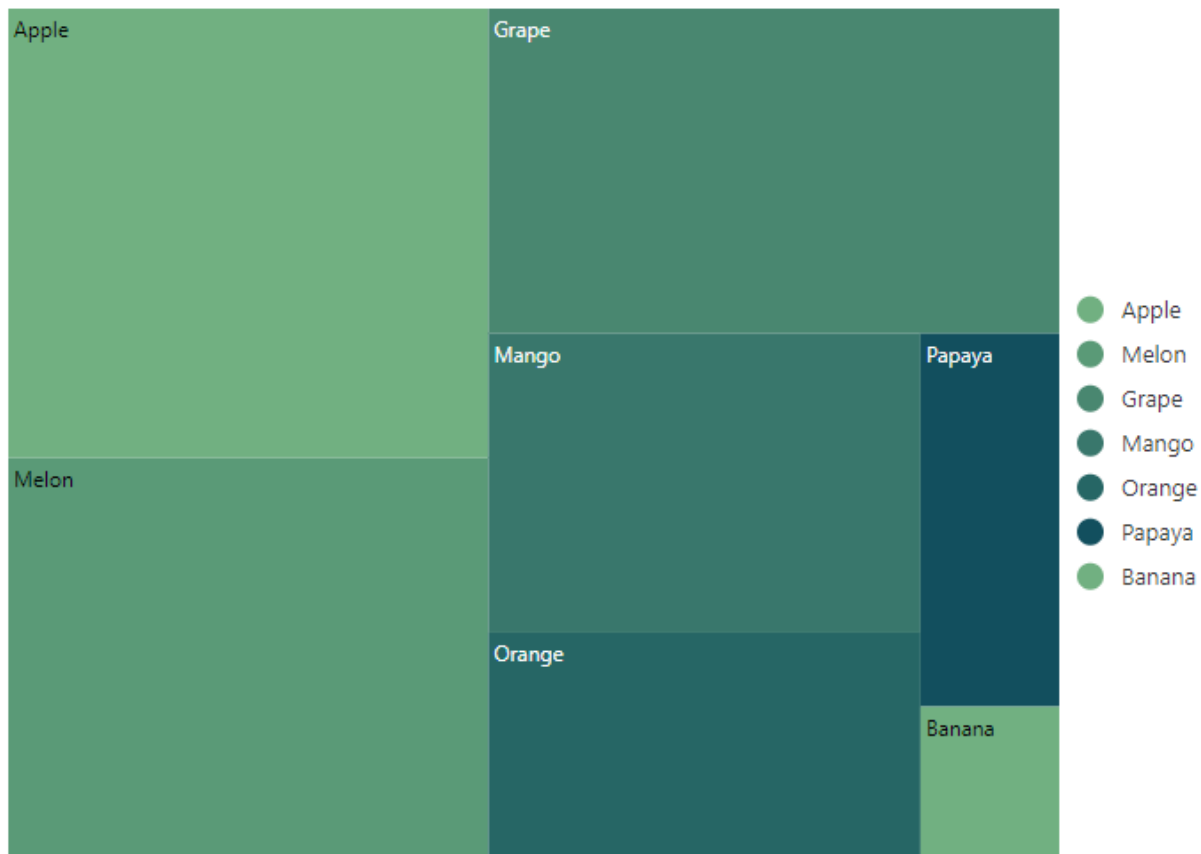
ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
```

```

<SfTreeMap DataSource="Fruits" TValue="Fruit" WeightValuePath="Count"
Width="700px" Height="500px" Palette='new string[] { "#71B081", "#5A9A77",
"#498770", "#39776C", "#266665", "#124F5E" }'>
<TreeMapLeafItemSettings LabelPath="Name">
</TreeMapLeafItemSettings>
<TreeMapLegendSettings Visible="true" Position="LegendPosition.Auto">
</TreeMapLegendSettings>
</SfTreeMap>
@code{
public class Fruit
{
public string Name { get; set; }
public int Count { get; set; }
};
public List<Fruit> Fruits = new List<Fruit> {
new Fruit { Name="Apple", Count=5000 },
new Fruit { Name="Mango", Count=3000 },
new Fruit { Name="Orange", Count=2300 },
new Fruit { Name="Banana", Count=500 },
new Fruit { Name="Grape", Count=4300 },
new Fruit { Name="Papaya", Count=1200 },
new Fruit { Name="Melon", Count=4500 }
};
}

```



Legend with RTL support

Set the [EnableRtl](#) property to **true**, the legend icon will be rendered on the right and the legend text will be rendered on the left of the legend icon.

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
<SfTreeMap DataSource="Cars" WeightValuePath="Count" ColorValuePath="Color"
EnableRtl="true">
  <TreeMapLeafItemSettings LabelPath="Name">
  </TreeMapLeafItemSettings>
  <TreeMapLegendSettings Visible="true" Position="LegendPosition.Top">
  </TreeMapLegendSettings>
</SfTreeMap>
@code {
public class Car
{
public string Name { get; set; }
public string Brand { get; set; }
public int Count { get; set; }
public string Color { get; set; }
};
public List<Car> Cars = new List<Car> {
new Car { Name="Mustang", Brand="Ford", Count=232, Color= "#71B081" },
new Car { Name="EcoSport", Brand="Ford", Count=121, Color= "#5A9A77" },
new Car { Name="Swift", Brand="Maruti", Count=143, Color= "#498770" },
new Car { Name="Baleno", Brand="Maruti", Count=454, Color= "#39776C" },
new Car { Name="Vitara Brezza", Brand="Maruti", Count=545, Color= "#266665" },
},
new Car { Name="A3 Cabriolet", Brand="Audi",Count=123, Color= "#124F5E" }
};
}
```



Drill-down in Blazor TreeMap Component

The TreeMap component supports drill-down to expose the hierarchy, achieved by clicking a node. If an item is clicked in the TreeMap, it will be moved to the next level or sub level hierarchy and returned back to the previous level by clicking the node.

Perform drill-down

The TreeMap items can be drilled by setting the [EnableDrillDown](#) property to **true**.

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
<SfTreeMap WeightValuePath="EmployeeCount" TValue="Employee"
DataSource="Employees" EnableDrillDown=true Palette='new string[]
{"#f44336", "#29b6f6", "#ab47bc", "#ffc107", "#5c6bc0", "#009688"}'>
  <TreeMapLevels>
    <TreeMapLevel GroupPath="Country">
      <TreeMapLevelBorder Color="black" Width="0.5">
      </TreeMapLevelBorder>
    </TreeMapLevel>
    <TreeMapLevel GroupPath="JobDescription">
      <TreeMapLevelBorder Color="black" Width="0.5">
      </TreeMapLevelBorder>
    </TreeMapLevel>
    <TreeMapLevel GroupPath="JobGroup">
      <TreeMapLevelBorder Color="black" Width="0.5">
      </TreeMapLevelBorder>
    </TreeMapLevel>
  </TreeMapLevels>
</SfTreeMap>
@code{
public class Employee
{
    public string Country { get; set; }
    public string JobDescription { get; set; }
    public string JobGroup { get; set; }
    public int EmployeeCount { get; set; }
};
public List<Employee> Employees = new List<Employee> {
    new Employee { Country= "USA", JobDescription= "Sales", JobGroup=
    "Executive", EmployeeCount= 20 },
    new Employee { Country= "USA", JobDescription= "Sales", JobGroup= "Analyst",
    EmployeeCount= 30 },
    new Employee { Country= "USA", JobDescription= "Marketing", EmployeeCount=
    40 },
    new Employee { Country= "USA", JobDescription= "Management", EmployeeCount=
    80 },
    new Employee { Country= "India", JobDescription= "Technical", JobGroup=
    "Testers", EmployeeCount= 100 },
    new Employee { Country= "India", JobDescription= "HR Executives",
    EmployeeCount= 30 },
    new Employee { Country= "India", JobDescription= "Accounts", EmployeeCount=
    40 },
    new Employee { Country= "UK", JobDescription= "Technical", JobGroup=
    "Testers", EmployeeCount= 30 },
    new Employee { Country= "UK", JobDescription= "HR Executives",
    EmployeeCount= 50 },
    new Employee { Country= "UK", JobDescription= "Accounts", EmployeeCount= 60
    }
};
}
```



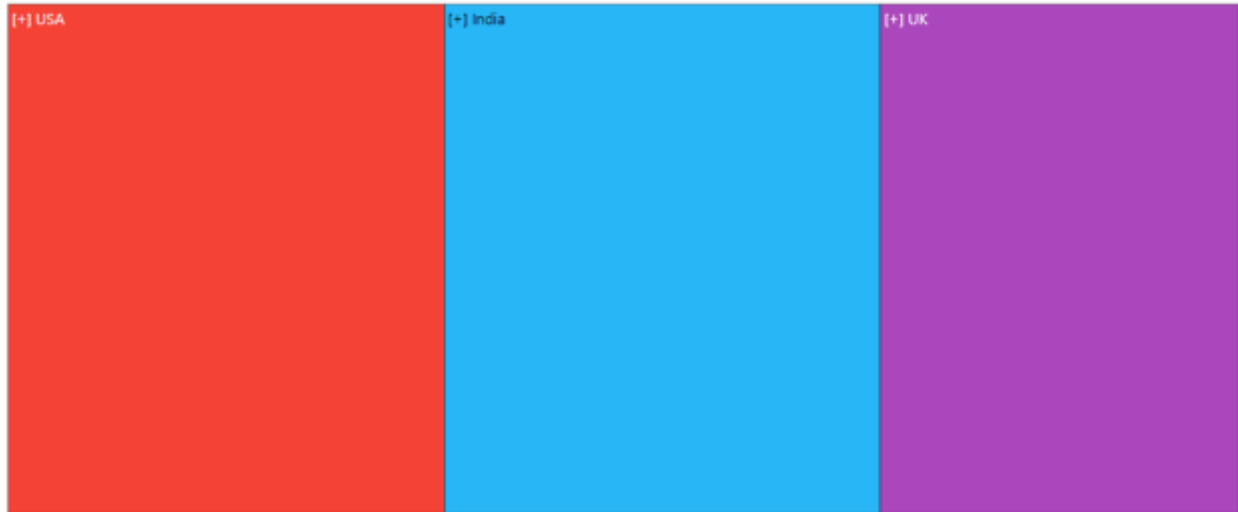
On-demand data loading

All the child items are rendered during the normal drill-down process, and visible at the initial rendering of the TreeMap. But on-demand data loading, it will not render child items at initial rendering, and child nodes will be rendered during the drill-down process by setting the [DrillDownView](#) property to **true**.

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
<SfTreeMap WeightValuePath="EmployeeCount" TValue="Employee"
DataSource="Employees" EnableDrillDown=true DrillDownView = "true"
Palette=new string[] { "#f44336", "#29b6f6", "#ab47bc", "#ffc107",
"#5c6bc0", "#009688"} '>
<TreeMapLevels>
<TreeMapLevel GroupPath="Country">
<TreeMapLevelBorder Color="black" Width="0.5">
</TreeMapLevelBorder>
</TreeMapLevel>
<TreeMapLevel GroupPath="JobDescription">
<TreeMapLevelBorder Color="black" Width="0.5">
</TreeMapLevelBorder>
</TreeMapLevel>
<TreeMapLevel GroupPath="JobGroup">
<TreeMapLevelBorder Color="black" Width="0.5">
</TreeMapLevelBorder>
</TreeMapLevel>
</TreeMapLevels>
</SfTreeMap>
```

Refer to the [code block](#) to know about the property value of **Employees**.



Breadcrumb support

TreeMap items are drilled, up to any level of parent using breadcrumb navigation and the level from root parent to current level is displayed at the top of item layout. It can be enabled by using the [EnableBreadcrumb](#) property to **true** and customize the breadcrumb connector using the [BreadcrumbConnector](#) property. By default, -(hyphen) is the connector.

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
<SfTreeMap WeightValuePath="EmployeeCount" TValue="Employee"
DataSource="Employees" EnableDrillDown=true EnableBreadcrumb="true"
BreadcrumbConnector=" -> " Palette='new string[] { "#f44336", "#29b6f6",
"#ab47bc", "#ffc107", "#5c6bc0", "#009688"}'>
<TreeMapLevels>
<TreeMapLevel GroupPath="Country">
<TreeMapLevelBorder Color="black" Width="0.5">
</TreeMapLevelBorder>
</TreeMapLevel>
<TreeMapLevel GroupPath="JobDescription">
<TreeMapLevelBorder Color="black" Width="0.5">
</TreeMapLevelBorder>
</TreeMapLevel>
<TreeMapLevel GroupPath="JobGroup">
<TreeMapLevelBorder Color="black" Width="0.5">
</TreeMapLevelBorder>
</TreeMapLevel>
</TreeMapLevels>
</SfTreeMap>
```

Refer to the [code block](#) to know about the property value of the **Employees**.



Initial drill-down

TreeMap items can be drilled on initial rendering and it can be enabled by specifying the item index in the [GroupIndex](#) property and level order name in the [GroupName](#) property of the [TreeMapInitialDrillSettings](#).

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
<SfTreeMap WeightValuePath="EmployeeCount" TValue="Employee"
DataSource="Employees" EnableDrillDown=true Palette='new string[]
{"#f44336", "#29b6f6", "#ab47bc", "#ffc107", "#5c6bc0", "#009688"}'>
<TreeMapInitialDrillSettings GroupIndex="0"
GroupName="India"></TreeMapInitialDrillSettings>
<TreeMapLevels>
<TreeMapLevel GroupPath="Country">
<TreeMapLevelBorder Color="black" Width="0.5">
</TreeMapLevelBorder>
</TreeMapLevel>
<TreeMapLevel GroupPath="JobDescription">
<TreeMapLevelBorder Color="black" Width="0.5">
</TreeMapLevelBorder>
</TreeMapLevel>
<TreeMapLevel GroupPath="JobGroup">
<TreeMapLevelBorder Color="black" Width="0.5">
</TreeMapLevelBorder>
</TreeMapLevel>
</TreeMapLevels>
</SfTreeMap>
```

Refer to the [code block](#) to know about the property value of **Employees**.



Tooltip in Blazor TreeMap Component

Tooltip is used to display details about the items in the TreeMap. When space constraints prevents from displaying the information using Data Labels, the tooltip comes in handy.

Default tooltip

The tooltip is not visible by default, to make it visible, set the [Visible](#) property in the [TreeMapTooltipSettings](#) to **true**.

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
<SfTreeMap WeightValuePath="Count" TValue="Fruit" DataSource="Fruits">
  <TreeMapLeafItemSettings LabelPath="Name"></TreeMapLeafItemSettings>
  <TreeMapTooltipSettings Visible=true></TreeMapTooltipSettings>
</SfTreeMap>

@code {
    public class Fruit
    {
        public string Name { get; set; }
        public int Count { get; set; }
    };

    public List<Fruit> Fruits = new List<Fruit> {
        new Fruit { Name="Apple", Count=5000 },
        new Fruit { Name="Mango", Count=3000 },
        new Fruit { Name="Orange", Count=2300 },
        new Fruit { Name="Banana", Count=500 },
        new Fruit { Name="Grape", Count=4300 },
        new Fruit { Name="Papaya", Count=1200 },
        new Fruit { Name="Melon", Count=4500 }
    };
}
```



Customization

Customize the TreeMap tooltip using the following properties.

- [Fill](#) - Specifies the color of the tooltip.
- [Opacity](#) - Specifies the opacity of the tooltip.
- [TreeMapTooltipBorder](#) - Specifies the tooltip border color and width.
- [TreeMapTooltipTextStyle](#) - Specifies the tooltip font family, style, weight, color and size.

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
<SfTreeMap WeightValuePath="Count" TValue="Fruit" DataSource="Fruits">
  <TreeMapLeafItemSettings LabelPath="Name" Fill="lightgray"
    Gap="2"></TreeMapLeafItemSettings>
  <TreeMapTooltipSettings Visible=true>
    <TreeMapTooltipTextStyle FontStyle="italic" FontWeight="bold" Size="15">
    </TreeMapTooltipTextStyle>
  </TreeMapTooltipSettings>
</SfTreeMap>
```

Refer to the [code block](#) to know about the property value of **Fruits**.



Formatting tooltip content

The tooltip content is displayed by default based on the [WeightValuePath](#). In addition, to show more information in the tooltip, use the [Format](#) property and define properties from the data source as following.

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
```

```
<SfTreeMap WeightValuePath="Count" TValue="Fruit" DataSource="Fruits">
  <TreeMapLeafItemSettings LabelPath="Name"></TreeMapLeafItemSettings>
  <TreeMapTooltipSettings Visible=true Format="Name: ${Name} - TotalCount:
    ${Count}"></TreeMapTooltipSettings>
</SfTreeMap>
```

Refer to the [code block](#) to know about the property value of **Fruits**.



Tooltip template

Tooltip can be rendered as a custom component using the [TooltipTemplate](#) property in the [TreeMapTooltipSettings](#) which accepts one or more UI elements as an input, that can be rendered as a part of the tooltip rendering.

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
<SfTreeMap WeightValuePath="Count" TValue="Fruit" DataSource="Fruits">
  <TreeMapLeafItemSettings LabelPath="Name" Fill="lightgray"
    Gap="2"></TreeMapLeafItemSettings>
  <TreeMapTooltipSettings Visible=true Opacity="0.8">
    <TooltipTemplate>
      @{
        var data = (context as Fruit);
        <table style="width:100%; background-color: #ffffff; border-spacing: 0px;
          border-collapse: separate; border: 1px solid grey; border-radius: 10px;
          padding-top: 5px; padding-bottom: 5px">
          <tr>
            <td style="font-weight: bold; color: black; padding-left: 5px;">Count:</td>
            <td style="font-weight: bold; color: black; padding-right: 5px;">@data.Count</td>
          </tr>
        </table>
      }
    </TooltipTemplate>
  </TreeMapTooltipSettings>
</SfTreeMap>
```

Refer to the [code block](#) to know about the property value of **Fruits**.



Selection and Highlight in Blazor TreeMap Component

Selection

Selection is used to select a particular group or item to differentiate from other items. Each item or each group can be selected and deselected while interacting with the items. The corresponding Treemap items are also selected while tapping a specific legend item, and vice versa.

The [Fill](#) property is used to change the selected item color. The [Color](#) and the [Width](#) properties are used to customize the selected item border, and the selection is enabled by using the [Enable](#) property to **true** in the [TreeMapSelectionSettings](#).

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
<SfTreeMap WeightValuePath="EmployeeCount" TValue="Employee"
DataSource="Employees">
  <TreeMapLeafItemSettings LabelPath="JobDescription" Fill="#8ebfe2">
  </TreeMapLeafItemSettings>
  <TreeMapLevels>
    <TreeMapLevel GroupPath="Country" Fill="#c5e2f7">
    </TreeMapLevel>
    <TreeMapLevel GroupPath="JobDescription" Fill="#a4d1f2">
    </TreeMapLevel>
    <TreeMapLevel GroupPath="JobGroup" Fill="#a4d1f2">
    </TreeMapLevel>
  </TreeMapLevels>
  <TreeMapSelectionSettings Enable="true" Fill="blue">
  <TreeMapSelectionBorder Color="black" Width="0.5"></TreeMapSelectionBorder>
  </TreeMapSelectionSettings>
</SfTreeMap>

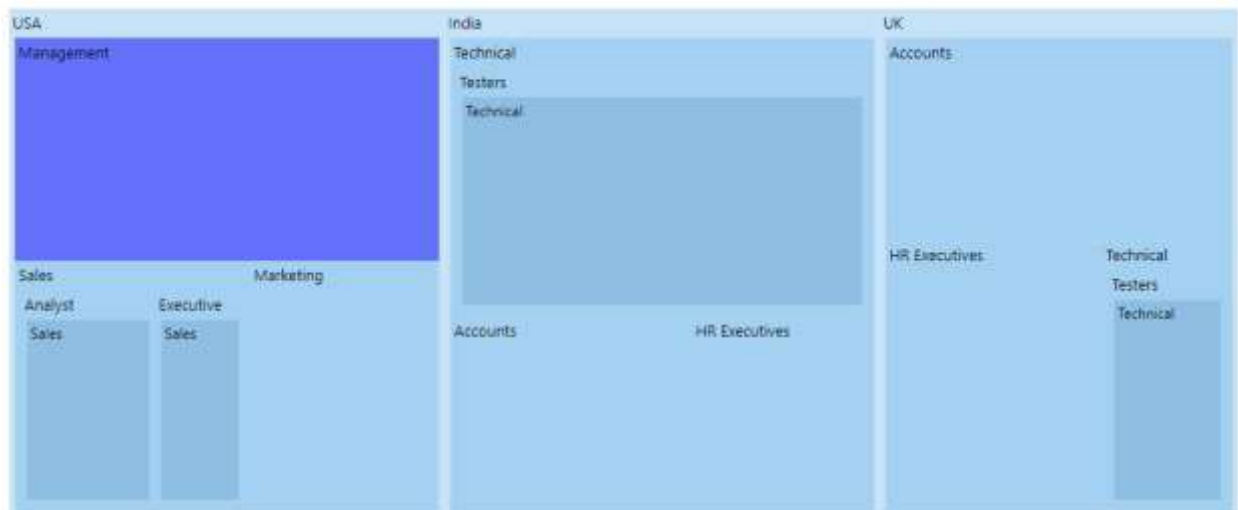
@code{
public class Employee
{
  public string Country { get; set; }
  public string JobDescription { get; set; }
  public string JobGroup { get; set; }
  public int EmployeeCount { get; set; }
};

public List<Employee> Employees = new List<Employee> {
  new Employee { Country= "USA", JobDescription= "Sales", JobGroup=
  "Executive", EmployeeCount= 20 },
  new Employee { Country= "USA", JobDescription= "Sales", JobGroup= "Analyst",
  EmployeeCount= 30 },
  new Employee { Country= "USA", JobDescription= "Marketing", EmployeeCount=
  40 },
```

```

new Employee { Country= "USA", JobDescription= "Management", EmployeeCount=
80 },
new Employee { Country= "India", JobDescription= "Technical", JobGroup=
"Testers", EmployeeCount= 100 },
new Employee { Country= "India", JobDescription= "HR Executives",
EmployeeCount= 30 },
new Employee { Country= "India", JobDescription= "Accounts", EmployeeCount=
40 },
new Employee { Country= "UK", JobDescription= "Technical", JobGroup=
"Testers", EmployeeCount= 30 },
new Employee { Country= "UK", JobDescription= "HR Executives",
EmployeeCount= 50 },
new Employee { Country= "UK", JobDescription= "Accounts", EmployeeCount= 60
}
};
}

```



Highlight

Highlight is used to highlight an item or group from other items or groups. Each item or each group can be highlighted by hovering the mouse over the items. The corresponding Treemap items are also highlighted while hovering over a specific legend item, and vice versa.

The [Fill](#) property is used to change the highlighted item color. The [Color](#) and the [Width](#) properties are used to customize the highlighted item border, and the highlight is enabled by setting the [Enable](#) property to **true** in the [TreeMapHighlightSettings](#).

ASPX-CS

```

@using Syncfusion.Blazor.TreeMap
<SfTreeMap WeightValuePath="EmployeeCount" TValue="Employee"
DataSource="Employees">
<TreeMapLeafItemSettings LabelPath="JobDescription" Fill="#8ebfe2">
</TreeMapLeafItemSettings>
<TreeMapLevels>
<TreeMapLevel GroupPath="Country" Fill="#c5e2f7">
</TreeMapLevel>
<TreeMapLevel GroupPath="JobDescription" Fill="#a4d1f2">
</TreeMapLevel>

```

```

<TreeMapLevel GroupPath="JobGroup" Fill="#a4d1f2">
</TreeMapLevel>
</TreeMapLevels>
<TreeMapHighlightSettings Enable=true Fill="blue">
<TreeMapHighlightBorder Color="black" Width="0.3">
</TreeMapHighlightBorder>
</TreeMapHighlightSettings>
</SfTreeMap>

```

Refer to the [code block](#) to know about the property value of **Employees**.



Print and Export in Blazor TreeMap Component

Print

The `PrintAsync` method can be used to print a rendered TreeMap directly from the browser and it can be enabled by setting the [AllowPrint](#) property to **true**.

ASPX-CS

```

@using Syncfusion.Blazor.TreeMap;
<button @onclick="PrintMap">Print Treemap</button>
<SfTreeMap @ref="Treemap" DataSource="@GrowthReport" WeightValuePath="GDP"
TValue="Country" AllowPrint="true">
<TreeMapLeafItemSettings LabelPath="Name" Fill="lightgray">
</TreeMapLeafItemSettings>
</SfTreeMap>
@code {
public SfTreeMap<Country> Treemap { get; set; }
public async Task PrintMap()
{
await Treemap.PrintAsync();
}
public class Country
{
public string Name { get; set; }
public double GDP { get; set; }
}
public List<Country> GrowthReport = new List<Country> {

```

```
new Country {Name="United States", GDP=17946 },
new Country {Name="China", GDP=10866 },
new Country {Name="Japan", GDP=4123 },
};
}
```



Export

Image Export

Export functionality can be enabled by setting the [AllowImageExport](#) property to **true**. The rendered TreeMap can be exported as an image with help of [ExportAsync](#) method and the method requires two parameters: image type and file name. The orientation setting is optional.

The TreeMap can be exported as an image in the following formats.

- [JPEG](#)
- [PNG](#)
- [SVG](#)

The following code example shows how to export the TreeMap in [PNG](#) format.

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap;
<button @onclick="ExportMap">Export Treemap</button>
<SfTreeMap @ref="Treemap" DataSource="@GrowthReport" WeightValuePath="GDP"
TValue="Country" AllowImageExport="true" AllowPdfExport="true">
<TreeMapLeafItemSettings LabelPath="Name" Fill="lightgray">
</TreeMapLeafItemSettings>
</SfTreeMap>
@code {
public SfTreeMap<Country> Treemap { get; set; }
public async Task ExportMap()
{
await Treemap.ExportAsync(ExportType.PNG, "Export",
Syncfusion.PdfExport.PdfPageOrientation.Portrait, true);
}
```



```
}
}
```

Refer to the [code block](#) to know about the property value of the **GrowthReport**.

PDF Export

PDF export functionality can be enabled by setting the [AllowPdfExport](#) property to **true**. The rendered TreeMap can be exported as **PDF** with the help of **ExportAsync** method and the export method requires two parameters: file type and file name. The orientation setting is optional.

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap;
<button @onclick="ExportMap">Export Treemap</button>
<SfTreeMap @ref="Treemap" DataSource="@GrowthReport" WeightValuePath="GDP"
TValue="Country" AllowImageExport="true" AllowPdfExport="true">
<TreeMapLeafItemSettings LabelPath="Name" Fill="lightgray">
</TreeMapLeafItemSettings>
</SfTreeMap>
@code {
public SfTreeMap<Country> Treemap { get; set; }
public async Task ExportMap()
{
await Treemap.ExportAsync(ExportType.PDF, "Export",
Syncfusion.PdfExport.PdfPageOrientation.Portrait);
}
}
```

Refer to the [code block](#) to know about the property value of the **GrowthReport**.

Accessibility in Blazor TreeMap Component

The TreeMap component provides built-in compliance with the [WAI-ARIA](#) specifications. The WAI-ARIA accessibility supports are achieved using attributes such as **aria-label**. It helps to provide information about the elements in a document for assistive technology.

This attribute provides text label with some default description for the following elements in the TreeMap.

<!-- markdownlint-disable MD033 -->

Element	Default description
TreeMap container	Reads the TreeMap description
TreeMap Title	Reads the TreeMap title
TreeMap Subtitle	Reads the TreeMap subtitle
Legend Title	Reads the legend title

Change this default description using the [Description](#) property available in the [TreeMapLegendSettings](#), [TreeMapTitleSettings](#), [TreeMapSubTitleSettings](#), and [SfTreeMap](#). It helps screen readers to read for assistive purpose.

Globalization in Blazor TreeMap Component

The TreeMap component supports globalization for the following elements:

- Data Label
- Tooltip

Globalization is the process of designing and developing a component that works in different cultures or locales. The [Format](#) property is used to globalize number, date, and time values in the TreeMap component.

In the following code example, tooltip and Data Label is globalized to currency format in Deutsch culture.

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap
<SfTreeMap WeightValuePath="GDP" DataSource="GrowthReports" Format="C">
  <TreeMapTooltipSettings Visible=true>
  </TreeMapTooltipSettings>
</SfTreeMap>
@code {
public class GDPReport
{
public string State { get; set; }
public double GDP { get; set; }
public double Percentage { get; set; }
public int Rank { get; set; }
};
public List<GDPReport> GrowthReports = new List<GDPReport> {
new GDPReport {State="United States", GDP=17946, Percentage=11.08, Rank=1},
new GDPReport {State="China", GDP=10866, Percentage= 28.42, Rank=2},
new GDPReport {State="Japan", GDP=4123, Percentage=-30.78, Rank=3},
new GDPReport {State="Germany", GDP=3355, Percentage=-5.19, Rank=4},
new GDPReport {State="United Kingdom", GDP=2848, Percentage=8.28, Rank=5},
new GDPReport {State="France", GDP=2421, Percentage=-9.69, Rank=6},
new GDPReport {State="India", GDP=2073, Percentage=13.65, Rank=7},
new GDPReport {State="Italy", GDP=1814, Percentage=-12.45, Rank=8},
new GDPReport {State="Brazil", GDP=1774, Percentage=-27.88, Rank=9},
new GDPReport {State="Canada", GDP=1550, Percentage=-15.02, Rank=10}
};
}
```

Refer [here](#) to configure localization for the Blazor server application, and refer [here](#) for the Blazor web assembly application.



Events in Blazor TreeMap Component

ItemHighlighted

Triggers, after highlighting the TreeMap items.

Argument name	Description
Cancel	Specifies the event cancel status.

ItemRendering

Triggers, before rendering the item of the TreeMap.

Argument name	Description
CurrentItem	Specifies the current rendering item.
Text	Specifies the text of the current item.
Cancel	Specifies the event cancel status.

ItemSelected

Triggers, after selecting the TreeMap item.

Argument name	Description
Text	Specifies the text of the selected item.
Cancel	Specifies the event cancel status.

LegendRendering

Triggers, before rendering the TreeMap legend.

Argument name	Description
---------------	-------------

Cancel	Specifies the event cancel status.
--------	------------------------------------

LegendItemRendering

Triggers, before rendering each of the legend item.

Argument name	Description
Fill	Specifies the legend shape color.
ImageUrl	Specifies the image URL.
Shape	Specifies the legend shape.
ShapeBorder	Specifies the legend border color and width.
Cancel	Specifies the event cancel status.

Loaded

Triggers, after the TreeMap component has been loaded.

Argument name	Description
IsResized	Specifies whether the component is resized or not.

Load

Triggers, before rendering the TreeMap. TreeMap will trigger this event first.

OnPrint

Triggers, before the print operation gets started.

Argument name	Description
Cancel	Specifies the event cancel status.

OnClick

Triggers, when clicking on the treemap.

Argument name	Description
MouseEvent	Specifies the pointer mouse event.
TreeMap	Specifies the current treemap instances.
Name	Specifies the name of the event.
Cancel	Specifies the event cancel status.

OnDoubleClick

Triggers, when double clicking on the treemap.

Argument name	Description
Cancel	Specifies the event cancel status.

DrillCompleted

Triggers, when drilling down functionality gets completed on the TreeMap item.

Argument name	Description
Cancel	Specifies the event cancel status.

OnDrillStart

Triggers, when drilling down functionality gets started on the TreeMap item.

Argument name	Description
GroupIndex	Specifies the index of the TreeMap item.
GroupName	Specifies the parent name of the TreeMap item.
Item	Specifies the current drill item.
RightClick	Return the boolean value whether it is right or not.
Cancel	Specifies the event cancel status.

OnItemClick

Triggers, when clicking on the TreeMap item.

Argument name	Description
GroupIndex	Specifies the index of the TreeMap item
GroupName	Specifies the parent name of the TreeMap item.
Item	Specifies the current item on click.
Text	Specifies the text of the current TreeMap item.
Cancel	Specifies the event cancel status.

OnItemMove

Triggers, when mouse moves on the TreeMap item.

Argument name	Description
Cancel	Specifies the event cancel status.

OnRightClick

Triggers, when right-clicked on the TreeMap.

Argument name	Description
Cancel	Specifies the event cancel status.

Resizing

Triggers, when resizing the TreeMap component.

Argument name	Description
CurrentSize	Specifies the size of the TreeMap.
PreviousSize	Specifies the previous size of the TreeMap.
Cancel	Specifies the event cancel status.

TooltipRendering

Triggers, before rendering the TreeMap tooltip.

Argument name	Description
Location	Specifies the location of the tooltip.
Text	Specifies the text of the tooltip.
TextStyle	Specifies the text style of the tooltip.
Data	Specifies the TreeMap item data, where the tooltip is to be rendered.
Cancel	Specifies the event cancel status.

Methods in Blazor TreeMap Component

Create an object for the TreeMap component using `@ref` property and call the desired TreeMap method.

Print

To print the rendered TreeMap component by setting the `AllowPrint` property to `true` and using the `PrintAsync` method.

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap;
<button @onclick="PrintMap">Print Treemap</button>
<SfTreeMap @ref="Treemap" DataSource="@growthReport" WeightValuePath="GDP"
TValue="Country" AllowPrint="true">
<TreeMapLeafItemSettings LabelPath="Name" Fill="lightgray">
</TreeMapLeafItemSettings>
</SfTreeMap>
@code {
public SfTreeMap<Country> Treemap { get; set; }
public async Task PrintMap()
{
await Treemap.PrintAsync();
}
public class Country
{
public string Name { get; set; }
public double GDP { get; set; }
}
public List<Country> growthReport = new List<Country> {
new Country {Name="United States", GDP=17946 },
new Country {Name="China", GDP=10866 },
new Country {Name="Japan", GDP=4123 },
};
```

```
}

```

Export

Using `ExportAsync` method the current TreeMap component will be exported to different file formats such as [PNG](#), [PDF](#), [JPEG](#) and [SVG](#).

The [AllowImageExport](#) and [AllowPdfExport](#) property represents to allow the file to be downloaded in an image and pdf type export.

Arguments	Description
type	Defines the export type such as PNG , PDF , JPEG and SVG .
fileName	Defines the file name.
orientation	Defines the orientation such as horizontal and vertical .
allowDownload	Defines the export file to be downloaded or not.

Export method returns the **Base64** string, if **allowDownload** argument is set to **false**. To download the file, paste the returned **Base64** string in the browser URL bar and press the enter button.

ASPX-CS

```
@using Syncfusion.Blazor.TreeMap;
<button @onclick="ExportMap">Export Treemap</button>
<SfTreeMap @ref="Treemap" DataSource="@growthReport" WeightValuePath="GDP"
TValue="Country" AllowImageExport="true" AllowPdfExport="true">
<TreeMapLeafItemSettings LabelPath="Name" Fill="lightgray">
</TreeMapLeafItemSettings>
</SfTreeMap>
@code {
public SfTreeMap<Country> Treemap { get; set; }
public async Task ExportMap()
{
await Treemap.ExportAsync(ExportType.SVG, "Export",
Syncfusion.PdfExport.PdfPageOrientation.Portrait, true);
}
public class Country
{
public string Name { get; set; }
public double GDP { get; set; }
}
private List<Country> growthReport = new List<Country> {
new Country {Name="United States", GDP=17946 },
new Country {Name="China", GDP=10866 },
new Country {Name="Japan", GDP=4123 },
};
}
```

Refresh

The `RefreshAsync` method helps to refresh the TreeMap component.

ASPX-CS

```

@using Syncfusion.Blazor.TreeMap;
<button @onclick="RefreshCall">Refresh</button>
<SfTreeMap @ref="Treemap" DataSource="@growthReport" WeightValuePath="GDP"
TValue="Country" AllowImageExport="true" AllowPdfExport="true">
</SfTreeMap>
@code {
public SfTreeMap<Country> Treemap { get; set; }
public async Task RefreshCall()
{
await Treemap.RefreshAsync();
}
public class Country
{
public string Name { get; set; }
public double GDP { get; set; }
}
private List<Country> growthReport = new List<Country> {
new Country {Name="United States", GDP=17946 },
new Country {Name="China", GDP=10866 },
new Country {Name="Japan", GDP=4123 },
};
}

```

SelectItem

The **SelectItemAsync** method can be used to select or unselect the TreeMap item programmatically.

Arguments	Description
levelOrder	Defines the level order name for the treemap item.
isSelected	Defines whether it has to select or unselect.

ASPX-CS

```

@using Syncfusion.Blazor.TreeMap;
<button @onclick="SelectCall">Select</button>
<SfTreeMap @ref="Treemap" DataSource="@growthReport" WeightValuePath="GDP"
TValue="Country">
<TreeMapLeafItemSettings LabelPath="Name" Fill="lightgray">
</TreeMapLeafItemSettings>
<TreeMapSelectionSettings Enable="true" Fill="#a4d1f2">
</TreeMapSelectionSettings>
</SfTreeMap>
@code {
public SfTreeMap<Country> Treemap { get; set; }
public async Task SelectCall()
{
await Treemap.SelectItemAsync(new string[] { "United States" }, true);
}
public class Country
{
public string Name { get; set; }
public double GDP { get; set; }
}
private List<Country> growthReport = new List<Country> {

```



```
new Country {Name="United States", GDP=17946 },  
new Country {Name="China", GDP=10866 },  
new Country {Name="Japan", GDP=4123 },  
};  
}
```

TreeView

<!-- markdownlint-disable MD024 -->

Getting Started with Blazor TreeView Component

This section briefly explains about how to include a **TreeView** in the Blazor server-side application. Refer [Getting Started with Syncfusion Blazor for Server-Side in Visual Studio](#) page for the introduction and configuring the common specifications.

To get started quickly with Blazor TreeView, check on this video:

{% youtube

"youtube:https://www.youtube.com/watch?v=mScCY8cgyM0"%}

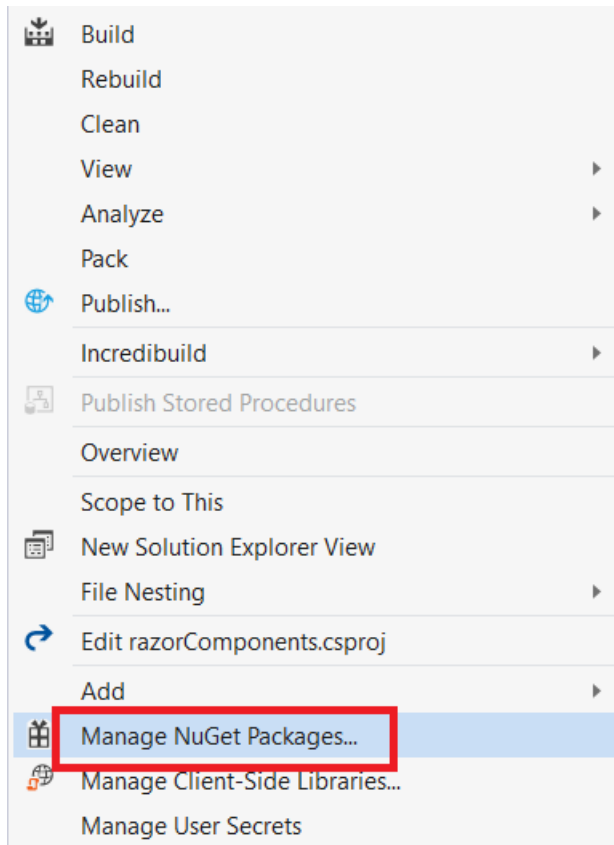
Importing Syncfusion Blazor component in the application

Use any one of the below standards to install the Syncfusion Blazor library in the application.

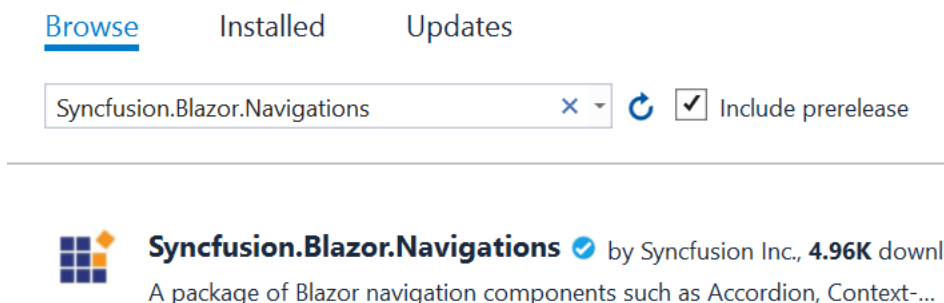
Using Syncfusion Blazor Individual NuGet Packages [New standard]

Starting with Volume 4, 2020 (v18.4.0.30) release, Syncfusion provides [individual NuGet packages](#) for the Syncfusion Blazor components. This new standard is highly recommended for the Blazor production applications. Refer to [this section](#) to know the benefits of the individual NuGet packages.

1. Install **Syncfusion.Blazor.Navigations** NuGet package to the application by using the **NuGet Package Manager**. Refer to the [Individual NuGet Packages](#) section for the available NuGet packages.



2. Search **Syncfusion.Blazor.Navigations** keyword in the Browse tab and install **Syncfusion.Blazor.Navigations** NuGet package in the application.



3. Once the installation process is completed, the Syncfusion Blazor Navigation package will be installed in the project.
4. Add the client-side style resources through [CDN](#) or from [NuGet](#) package in the `~/Pages/_Host.cshtml` page.

HTML

```
<head>
```

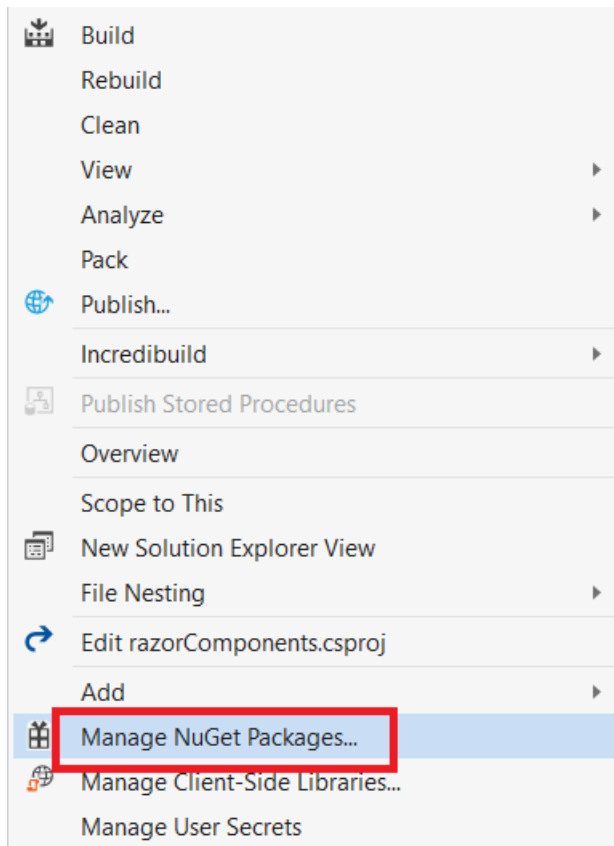
```
....  
....  
<link href="_content/Syncfusion.Blazor.Themes/bootstrap4.css"  
rel="stylesheet" />  
</head>
```

Warning: Syncfusion.Blazor package should not be installed along with [individual NuGet packages](#). Hence, the above Syncfusion.Blazor.Themes static web assets have to be added (styles) in the application.

Using Syncfusion.Blazor NuGet Package [Old standard]

Warning: If the above new standard (individual NuGet packages) is preferred, then skip this section. Using both old and new standards in the same application will throw ambiguous compilation errors.

1. Install **Syncfusion.Blazor** NuGet package to the application by using the **NuGet Package Manager** Right-click the project and then select **Manage NuGet Packages**.



2. Search **Syncfusion.Blazor** keyword in the Browse tab and install **Syncfusion.Blazor** NuGet package in the application.

[Browse](#)

Installed

Updates

Syncfusion.Blazor



Include prerelease

**Syncfusion.Blazor** by Syncfusion Inc., **192K** downloads

Syncfusion Blazor Components are built from the ground up to be lightw...

- Once the installation process is completed, the Syncfusion Blazor package will be installed in the project.
- Add the client-side style resources using NuGet package to the `element` of the `~/wwwroot/index.html` page in Blazor WebAssembly app or `~/Pages/_Host.cshtml` page in Blazor Server app.

HTML

```
<head>
<link href="_content/Syncfusion.Blazor/styles/bootstrap4.css"
rel="stylesheet" />
</head>
```

HTML

```
<head>
<link href="https://cdn.syncfusion.com/blazor/{{ site.blazorversion
}}/styles/bootstrap4.css" rel="stylesheet" />
</head>
```

For Internet Explorer 11 kindly refer the polyfills. Refer the [documentation](#) for more information.

HTML

```
<head>
<link href="_content/Syncfusion.Blazor/styles/bootstrap4.css"
rel="stylesheet" />
<script
src="https://github.com/Daddoon/Blazor.Polyfill/releases/download/3.0.1/blaz
or.polyfill.min.js"></script>
</head>
```

Add Syncfusion Blazor service in Startup.cs (Server-side application)

Open the **Startup.cs** file and add services required by Syncfusion components using `services.AddSyncfusionBlazor()` method. Add this method in the **ConfigureServices** function as follows.

C#

```
using Syncfusion.Blazor;
```

```
namespace BlazorApplication
{
    public class Startup
    {
        ....
        ....
        public void ConfigureServices(IServiceCollection services)
        {
            ....
            ....
            services.AddSyncfusionBlazor();
        }
    }
}
```

Add Syncfusion Blazor service in Program.cs (Client-side application)

Open the **Program.cs** file and add services required by Syncfusion components using `builder.Services.AddSyncfusionBlazor()` method. Add this method in the **Main** function as follows.

C#

```
namespace BlazorApplication
{
    public class Program
    {
        ....
        ....
        public static async Task Main(string[] args)
        {
            ....
            ....
            builder.Services.AddSyncfusionBlazor();
        }
    }
}
```

Adding component package to the application

Open `~/_Imports.razor` file and import the `Syncfusion.Blazor.Navigations` packages.

ASPX-CS

```
@using Syncfusion.Blazor
@using Syncfusion.Blazor.Navigations
```

Adding TreeView component to the application

Now, add the Syncfusion Blazor TreeView component in any web page razor in the **Pages** folder. For example, the Blazor TreeView component is added in the `~/Pages/Index.razor` page.

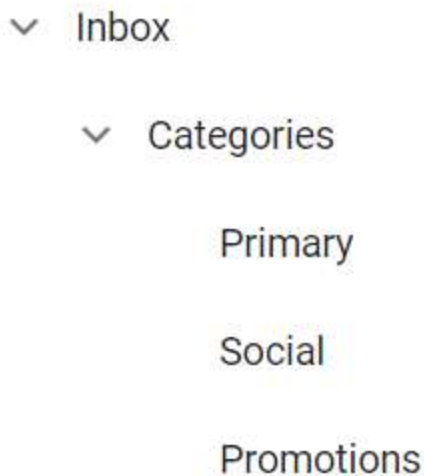
ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<SfTreeView TValue="MailItem">
```

```
<TreeViewFieldsSettings TValue="MailItem" Id="Id" DataSource="@MyFolder"
Text="FolderName" ParentID="ParentId" HasChildren="HasSubFolders"
Expanded="Expanded"></TreeViewFieldsSettings>
</SfTreeView>
@code{
public class MailItem
{
public string Id { get; set; }
public string ParentId { get; set; }
public string FolderName { get; set; }
public bool Expanded { get; set; }
public bool HasSubFolders { get; set; }
}
List<MailItem> MyFolder = new List<MailItem>();
protected override void OnInitialized()
{
base.OnInitialized();
MyFolder.Add(new MailItem
{
Id = "1",
FolderName = "Inbox",
HasSubFolders = true,
Expanded = true
});
MyFolder.Add(new MailItem
{
Id = "2",
ParentId = "1",
FolderName = "Categories",
Expanded = true,
HasSubFolders = true
});
MyFolder.Add(new MailItem
{
Id = "3",
ParentId = "2",
FolderName = "Primary"
});
MyFolder.Add(new MailItem
{
Id = "4",
ParentId = "2",
FolderName = "Social"
});
MyFolder.Add(new MailItem
{
Id = "5",
ParentId = "2",
FolderName = "Promotions"
});
}
}
```

Run the application

After successful compilation of the application, simply press **F5** to run the application.



Refer to the [Blazor TreeView](#) feature tour page for its groundbreaking feature representations. The [Blazor TreeView example](#) can also be explored to understand how to present and manipulate data.

See Also

- [Getting Started with Syncfusion Blazor for Client-Side in .NET Core CLI](#)
- [Getting Started with Syncfusion Blazor for Client-Side in Visual Studio 2019](#)
- [Getting Started with Syncfusion Blazor for Server-Side in .NET Core CLI](#)

Data Binding in Blazor TreeView Component

The Blazor TreeView component provides the option to load data either from the 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 objects 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.

Blazor TreeView has **load on demand** (Lazy load), by default. It reduces the bandwidth size when consuming huge data. It loads first level nodes initially, and when parent node is expanded, loads the child nodes based on the **ParentID/Child** member.

By default, the **LoadOnDemand** is set to true. By disabling this property, all the tree nodes are rendered at the beginning itself. The **DataBound** event can be used to perform actions. This event will be triggered once the data source is populated in the TreeView.

Local data

To bind local data to the Blazor TreeView, assign a list of objects to the **DataSource** property. The Blazor TreeView component requires three fields (Id, Text, and ParentID) 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

Blazor TreeView can be populated with hierarchical data source that contains nested list of objects. A hierarchical data can be directly assigned to the `DataSource` property, and map all the field members with corresponding keys from the hierarchical data to `Fields` property.

In the following example, **Id**, **FolderName**, and **SubFolders** columns from hierarchical data have been mapped to **Id**, **Text**, and **Child** fields, respectively.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<SfTreeView TValue="MailItem">
  <TreeViewFieldsSettings TValue="MailItem" Id="Id" Text="FolderName"
  Child="SubFolders" DataSource="@MyFolder"
  Expanded="Expanded"></TreeViewFieldsSettings>
</SfTreeView>
@code{
public class MailItem
{
public string Id { get; set; }
public string FolderName { get; set; }
public bool Expanded { get; set; }
public List<MailItem> SubFolders { get; set; }
}
List<MailItem> MyFolder = new List<MailItem>();
protected override void OnInitialized()
{
base.OnInitialized();
List<MailItem> Folder1 = new List<MailItem>();
MyFolder.Add(new MailItem
{
Id = "01",
FolderName = "Inbox",
SubFolders = Folder1
});
List<MailItem> Folder2 = new List<MailItem>();
Folder1.Add(new MailItem
{
Id = "01-01",
FolderName = "Categories",
SubFolders = Folder2
});
Folder2.Add(new MailItem
{
Id = "01-02",
FolderName = "Primary"
});
Folder2.Add(new MailItem
{
Id = "01-03",
FolderName = "Social"
});
Folder2.Add(new MailItem
{
```



```
Id = "01-04",
FolderName = "Promotions"
});
List<MailItem> Folder3 = new List<MailItem>();
MyFolder.Add(new MailItem
{
    Id = "02",
    FolderName = "Others",
    Expanded = true,
    SubFolders = Folder3
});
Folder3.Add(new MailItem
{
    Id = "02-01",
    FolderName = "Sent Items"
});
Folder3.Add(new MailItem
{
    Id = "02-02",
    FolderName = "Delete Items"
});
Folder3.Add(new MailItem
{
    Id = "02-03",
    FolderName = "Drafts"
});
Folder3.Add(new MailItem
{
    Id = "02-04",
    FolderName = "Archive"
});
}
```

> Inbox

▼ Others

Sent Items

Delete Items

Drafts

Archive

Self-referential data

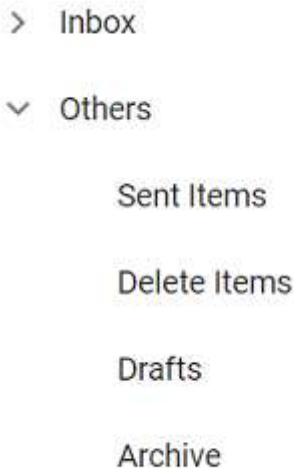
Blazor TreeView can be populated from self-referential data structure that contains list of objects with **ParentID** mapping. The self-referential data can be directly assigned to the **DataSource** property, and map all the field members with corresponding keys from self-referential data to **Fields** property.

To render the root level nodes, specify the **ParentID** as null or no need to specify the **ParentID** in **DataSource**. In the following example, **Id**, **Pid**, **HasSubFolders**, and **FolderName** columns from self-referential data have been mapped to **Id**, **ParentId**, **HasChildren**, and **Text** fields, respectively.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<SfTreeView TValue="MailItem">
  <TreeViewFieldsSettings TValue="MailItem" Id="Id" DataSource="@MyFolder"
  Text="FolderName" ParentID="ParentId" HasChildren="HasSubFolders"
  Expanded="Expanded"></TreeViewFieldsSettings>
</SfTreeView>
@code{
public class MailItem
{
public string Id { get; set; }
public string ParentId { get; set; }
public string FolderName { get; set; }
public bool Expanded { get; set; }
public bool HasSubFolders { get; set; }
}
List<MailItem> MyFolder = new List<MailItem>();
protected override void OnInitialized()
{
base.OnInitialized();
MyFolder.Add(new MailItem
{
Id = "1",
FolderName = "Inbox",
HasSubFolders = true,
});
MyFolder.Add(new MailItem
{
Id = "2",
ParentId = "1",
HasSubFolders = true,
FolderName = "Categories"
});
MyFolder.Add(new MailItem
{
Id = "3",
ParentId = "2",
FolderName = "Primary"
});
MyFolder.Add(new MailItem
{
Id = "4",
ParentId = "2",
FolderName = "Social"
});
MyFolder.Add(new MailItem
```

```
{
    Id = "5",
    ParentId = "2",
    FolderName = "Promotions"
});
MyFolder.Add(new MailItem
{
    Id = "6",
    FolderName = "Others",
    HasSubFolders = true,
    Expanded = true
});
MyFolder.Add(new MailItem
{
    Id = "7",
    ParentId = "6",
    FolderName = "Sent Items"
});
MyFolder.Add(new MailItem
{
    Id = "8",
    ParentId = "6",
    FolderName = "Delete Items"
});
MyFolder.Add(new MailItem
{
    Id = "9",
    ParentId = "6",
    FolderName = "Drafts"
});
MyFolder.Add(new MailItem
{
    Id = "10",
    ParentId = "6",
    FolderName = "Archive"
});
}
```



Remote data

Blazor TreeView can also be populated from a remote data service with the help of **DataManager** component 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. A service data can be assigned as an instance of **DataManager** to the **DataSource** property. To interact with remote data source, provide the endpoint url.

The **DataManager** that acts as an interface between the service endpoint and the TreeView 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.Blazor.Data** provides some predefined 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 remote services. The **EmployeeID**, **FirstName**, and **EmployeeID** columns from Employees table have been mapped to **Id**, **Text**, and **HasChildren** fields respectively for first level nodes.

The **OrderID**, **EmployeeID**, and **ShipName** columns from orders table have been mapped to **Id**, **ParentID**, and **Text** fields respectively for second level nodes.

ASPX-CS

```

@using Syncfusion.Blazor.Navigations
@using Syncfusion.Blazor.Data
<SfTreeView TValue="TreeData">
  <TreeViewFieldsSettings TValue="TreeData" Query="@Query" Id="EmployeeID"
  Text="FirstName" HasChildren="EmployeeID">
    <SfDataManager Url="http://services.odata.org/V4/Northwind/Northwind.svc"
    Adaptor="@Syncfusion.Blazor.Adaptors.ODataV4Adaptor"
    CrossDomain="true"></SfDataManager>
    <TreeViewFieldChild TValue="TreeData" Query="@SubQuery" Id="OrderID"
    Text="ShipName" ParentID="EmployeeID">
      <SfDataManager Url="http://services.odata.org/V4/Northwind/Northwind.svc"
      Adaptor="@Syncfusion.Blazor.Adaptors.ODataV4Adaptor"
      CrossDomain="true"></SfDataManager>
    </TreeViewFieldChild>
  </TreeViewFieldsSettings>
</SfTreeView>
@code{
public Query Query = new Query().From("Employees").Select(new List<string> {
  "EmployeeID", "FirstName" }).Take(5).RequiresCount();
public Query SubQuery = new Query().From("Orders").Select(new List<string> {
  "OrderID", "EmployeeID", "ShipName" }).Take(5).RequiresCount();
public class TreeData
{
  public int? EmployeeID { get; set; }
  public int OrderID { get; set; }
  public string ShipName { get; set; }
  public string FirstName { get; set; }
}
}

```

> Nancy

> Andrew

> Janet

> Margaret

> Steven

Entity Framework

The following steps must be followed to consume data from the **Entity Framework** in the TreeView component.

Create DbContext class

The first step is to create a DbContext class called **OrganizationContext** to connect to the Microsoft SQL Server database.

CSHARP

```

using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using SQLTreeView.Shared.Models;
using SQLTreeView.Data;
namespace SQLTreeView.Shared.DataAccess
{
    public class OrganizationContext : DbContext
    {
        public virtual DbSet<OrganizationDetails> Organization { get; set; }
        protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
        {
            if (!optionsBuilder.IsConfigured)
            {
                // To make the sample runnable, replace your local file path of the MDF file
                here
                optionsBuilder.UseSqlServer(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=
D:\Blazor\TreeView\SQLTreeView\Shared\App_Data\NORTHWND.MDF';Integrated
Security=True;Connect Timeout=30");
            }
        }
    }
}

```

Create data access layer to perform CRUD operation

Now, create a class named **OrganizationDataAccessLayer**, which act as the data access layer for retrieving the records from the database table. Also, add methods such as **AddEmployee**, **UpdateEmployee**, **DeleteEmployee** in the “**OrganizationDataAccessLayer.cs**” to handle the insert, update, and remove operations respectively.

C#SHARP

```

using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using SQLTreeView.Shared.Models;
using SQLTreeView.Data;
using SQLTreeView.Shared.DataAccess;
namespace SQLTreeView.Shared.DataAccess
{
    public class OrganizationDataAccessLayer
    {
        OrganizationContext db = new OrganizationContext();
        List<OrganizationDetails> EmployeeList = new List<OrganizationDetails>();
        // returns the organization data from the data base
        public DbSet<OrganizationDetails> GetAllEmployees()
        {
            try
            {
                return db.Organization;
            }
        }
    }
}

```

```
}  
catch  
{  
    throw;  
}  
}  
  
// Adds the new entry to the data base  
public void AddEmployee(OrganizationDetails Employee)  
{  
    try  
    {  
        db.Organization.Add(Employee);  
        OrganizationDetails ParentDetails = db.Organization.Find(Employee.ParentId);  
        if (ParentDetails != null)  
        {  
            ParentDetails.HasTeam = true;  
        }  
        db.SaveChanges();  
    }  
    catch  
    {  
        throw;  
    }  
}  
  
// Update the existing data in the data base  
public void UpdateEmployee(OrganizationDetails Employee)  
{  
    try  
    {  
        db.Entry(Employee).State = EntityState.Modified;  
        db.SaveChanges();  
    }  
    catch  
    {  
        throw;  
    }  
}  
  
// To delete an entry from the data base  
public void DeleteEmployee(int id)  
{  
    try  
    {  
        OrganizationDetails Employee = db.Organization.Find(id);  
        db.Organization.Remove(Employee);  
        DeleteChildEmployee(id);  
        db.Organization.RemoveRange(EmployeeList);  
        db.SaveChanges();  
    }  
    catch  
    {  
        throw;  
    }  
}  
  
// To delete the nested child from the data base  
public void DeleteChildEmployee(int id)  
{  
    try
```

```

{
    var data = GetAllEmployees().ToList();
    for (int i = 0; i < data.Count(); i++)
    {
        if (data[i].ParentId == id && EmployeeList.Contains(data[i]) == false)
        {
            EmployeeList.Add(data[i]);
            if (data[i].HasTeam == true)
            {
                DeleteChildEmployee(data[i].Id);
            }
        }
    }
}
catch
{
    throw;
}
}
}

```

Creating Web API Controller

A Web API Controller should be created which allows the TreeView directly to consume data from the Entity Framework. Also, create a new **Post**, **Put**, **Delete** method in the web API controller which will perform the CRUD operations and returns the appropriate resultant data. The '**SfDataManager**' will make requests to this action based on the route name.

C#SHARP

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Primitives;
using SQLTreeView.Data;
using SQLTreeView.Shared.DataAccess;
using SQLTreeView.Shared.Models;
namespace WebApplication1.Server.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class DefaultController : ControllerBase
    {
        OrganizationDataAccessLayer db = new OrganizationDataAccessLayer();
        [HttpGet]
        public object Get()
        {
            // Get the DataSource from Database
            var data = db.GetAllEmployees().ToList();
            var queryString = Request.Query;
            if (queryString.Keys.Contains("$filter"))
            {

```



```

StringValues Skip;
StringValues Take;
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();
string filter = string.Join("", queryString["$filter"].ToString().Split('
').Skip(2)); // get filter from querystring
data = data.Where(d => d.ParentId.ToString() == filter).ToList();
return data.Skip(skip).Take(top);
}
else
{
data = data.Where(d => d.ParentId == null).ToList();
return data;
}
}
[HttpGet("{id}")]
public object GetIndex(string id)
{
// Get the DataSource from Database
var data = db.GetAllEmployees().ToList();
int index;
var count = data.Count();
if (count > 0)
{
index = (data[data.Count - 1].Id);
} else
{
index = 0;
}
return index;
}
[HttpPost]
public void Post([FromBody]OrganizationDetails employee)
{
db.AddEmployee(employee);
}
[HttpPut]
public object Put([FromBody]OrganizationDetails employee)
{
db.UpdateEmployee(employee);
return employee;
}
[HttpDelete("{id}")]
public void Delete(int id)
{
db.DeleteEmployee(id);
}
}

```

Configure Blazor TreeView component using Web API adaptor

Now, the Blazor TreeView can be configured using the '**SfDataManager**' to interact with the created Web API and consume the data appropriately. To interact with web API, use web API adaptor.

The CRUD operation has been performed in the TreeView component using the context menu.

CSHARP

```
@using Syncfusion.Blazor
@using Syncfusion.Blazor.Navigations
@using Syncfusion.Blazor.Data
@using Newtonsoft.Json;
@inject HttpClient Http
<div id="treeview">
<SfTreeView @ref="tree" TValue="Employee">
<TreeViewFieldsSettings TValue="Employee" Id="Id" Text="Name"
ParentID="ParentId" HasChildren="HasTeam" Expanded="IsExpanded">
<SfDataManager Url="api/Default" Adaptor="Adaptors.WebApiAdaptor"
CrossDomain="true"></SfDataManager>
</TreeViewFieldsSettings>
<TreeViewEvents TValue="Employee"
NodeClicked="nodeClicked"></TreeViewEvents>
<SfContextMenu TValue="MenuItem" @ref="menu" Target="#treeview"
Items="@MenuItems">
<ContextMenuEvents TValue="MenuItem"
ItemSelected="MenuSelect"></ContextMenuEvents>
</SfContextMenu>
</SfTreeView>
</div>
@code{
SfTreeView<Employee> tree;
SfContextMenu<MenuItem> menu;
string selectedId;
int index;
// Datasource for menu items
public List<MenuItem> MenuItems = new List<MenuItem>{
new MenuItem { Text = "Edit" },
new MenuItem { Text = "Remove" },
new MenuItem { Text = "Add" }
};
public class Employee
{
public int Id { get; set; }
public int? ParentId { get; set; }
public bool? HasTeam { get; set; }
public bool? IsExpanded { get; set; }
public string Name { get; set; }
}
protected override async Task OnInitializedAsync()
{
// To get the last item index from the db
var count = await Http.GetJsonAsync<int>("api/Default/index");
this.index = count + 1;
}
// Triggers when TreeView node is clicked
public async void nodeClicked(NodeClickEventArgs args)
{
this.selectedId = null;
string eventString = JsonConvert.SerializeObject(args.Event);
Dictionary<string, dynamic> eventParameters =
JsonConvert.DeserializeObject<Dictionary<string, dynamic>>(eventString);
```

```

if ((eventParameters["which"]).ToString() == "3")
{
    // To get the selected node id upon context menu click
    this.selectedId = (await args.Node.GetAttribute("data-uid")).ToString();
}
}
// To add a new node
void AddNodes()
{
    List<Employee> TreeData = new List<Employee>();
    TreeData.Add(new Employee
    {
        Id = this.index,
        Name = "New Entry",
        ParentId = Int32.Parse(this.selectedId)
    });
    this.tree.AddNodes(TreeData, this.selectedId);
    this.index = this.index + 1;
}
// To delete a tree node
void RemoveNodes()
{
    string[] removeNode = new string[] { this.selectedId };
    this.tree.RemoveNodes(removeNode);
}
// To edit a tree node
async void RenameNodes()
{
    tree.BeginEdit(this.selectedId);
}
// Triggers when context menu is selected
public void MenuSelect(MenuEventArgs<MenuItem> args)
{
    string selectedText = args.Item.Text;
    if (selectedText == "Edit")
    {
        this.RenameNodes();
    }
    else if (selectedText == "Remove")
    {
        this.RemoveNodes();
    }
    else if (selectedText == "Add")
    {
        this.AddNodes();
    }
    this.selectedId = null;
}
}

```

The fully working sample can be found [here](#).

CheckBox in Blazor TreeView Component

The Blazor TreeView component allows to check more than one node in TreeView without affecting the UI's appearance by enabling the `ShowCheckBox` property. When this property is enabled, checkbox appears before each TreeView node text.

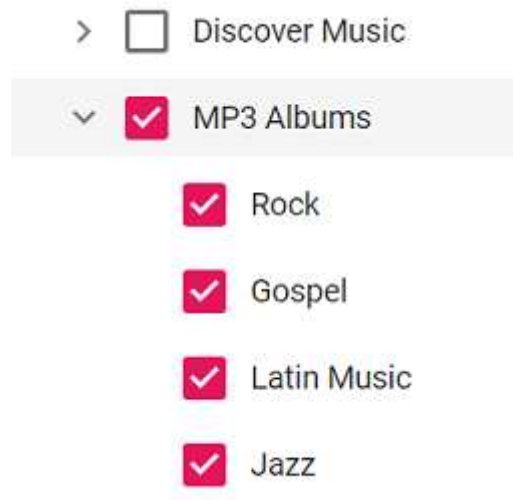
- If one of the child nodes is not in a checked state, then the parent node will be in an intermediate state.
- If all the child nodes are in checked state, then the parent node's state will also be checked.
- If a parent node is checked, then all the child nodes' state will also be checked.

By default, the checkbox state of parent and child nodes are dependent on each other. For independent checked state, achieve it using the `AutoCheck` property.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<SfTreeView TValue="MusicAlbum" ShowCheckBox="true" AutoCheck="true">
<TreeViewFieldsSettings TValue="MusicAlbum" Id="Id" DataSource="@Albums"
Text="Name" ParentID="ParentId" HasChildren="HasChild" Expanded="Expanded"
IsChecked="IsChecked"></TreeViewFieldsSettings>
</SfTreeView>
@code{
public class MusicAlbum
{
public int Id { get; set; }
public int? ParentId { get; set; }
public string Name { get; set; }
public bool Expanded { get; set; }
public bool? IsChecked { get; set; }
public bool HasChild { get; set; }
}
List<MusicAlbum> Albums = new List<MusicAlbum>();
protected override void OnInitialized()
{
base.OnInitialized();
Albums.Add(new MusicAlbum
{
Id = 1,
Name = "Discover Music",
HasChild = true,
});
Albums.Add(new MusicAlbum
{
Id = 2,
ParentId = 1,
Name = "Hot Singles"
});
Albums.Add(new MusicAlbum
{
Id = 3,
ParentId = 1,
Name = "Rising Artists"
});
Albums.Add(new MusicAlbum
{
```

```
Id = 4,
ParentId = 1,
Name = "Live Music"
});
Albums.Add(new MusicAlbum
{
Id = 14,
HasChild = true,
Name = "MP3 Albums",
Expanded = true,
IsChecked = true
});
Albums.Add(new MusicAlbum
{
Id = 15,
ParentId = 14,
Name = "Rock"
});
Albums.Add(new MusicAlbum
{
Id = 16,
Name = "Gospel",
ParentId = 14,
});
Albums.Add(new MusicAlbum
{
Id = 17,
ParentId = 14,
Name = "Latin Music"
});
Albums.Add(new MusicAlbum
{
Id = 18,
ParentId = 14,
Name = "Jazz"
});
}
}
```



Node Editing in Blazor TreeView Component

The Blazor TreeView allows to edit nodes by setting the `AllowEditing` property to **true**. To directly edit the nodes in place, **double click** the TreeView node or **select** the node and press **F2** key.

When editing is completed by focus out or by pressing the **Enter** key, the modified node's text saves automatically. If you do not want to save the modified node's text in TreeView node, press **Escape** key. It does not save the edited text to the TreeView node.

- Node editing can also be performed programmatically by using the `BeginEdit` method. On passing the node ID or element through this method, the edit textbox will be created for the particular node thus allowing us to edit it.
- In order to validate or prevent editing, the `NodeEditing` event can be used which is triggered before the TreeView node is renamed. On successfully renaming a node the `NodeEdited` event will be triggered.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<SfTreeView TValue="DriveData" AllowEditing="true">
  <TreeViewFieldsSettings TValue="DriveData" Id="NodeId" Text="NodeText"
  Child="Children" DataSource="@Drive"
  Expanded="Expanded"></TreeViewFieldsSettings>
</SfTreeView>
@code{
public class DriveData
{
    public string NodeId { get; set; }
    public string NodeText { get; set; }
    public bool Expanded { get; set; }
    public bool Selected { get; set; }
    public List<DriveData> Children;
}
object Child;
List<DriveData> Drive = new List<DriveData>();
protected override void OnInitialized()
```

```
{
base.OnInitialized();
List<DriveData> Folder1 = new List<DriveData>();
Drive.Add(new DriveData
{
    NodeId = "01",
    NodeText = "Local Disk (C:)",
    Children = Folder1,
});
List<DriveData> File1 = new List<DriveData>();
Folder1.Add(new DriveData
{
    NodeId = "01-01",
    NodeText = "Program Files",
    Children = File1
});
File1.Add(new DriveData
{
    NodeId = "01-01-01",
    NodeText = "Windows NT"
});
List<DriveData> File2 = new List<DriveData>();
Folder1.Add(new DriveData
{
    NodeId = "01-02",
    NodeText = "Users",
    Expanded = true,
    Children = File2
});
File2.Add(new DriveData
{
    NodeId = "01-02-01",
    NodeText = "Smith"
});
List<DriveData> File3 = new List<DriveData>();
Folder1.Add(new DriveData
{
    NodeId = "01-03",
    NodeText = "Windows",
    Children = File3
});
File3.Add(new DriveData
{
    NodeId = "01-03-01",
    NodeText = "Boot"
});
List<DriveData> Folder2 = new List<DriveData>();
Drive.Add(new DriveData
{
    NodeId = "02",
    NodeText = "Local Disk (D:)",
    Children = Folder2,
    Expanded = true,
});
List<DriveData> File4 = new List<DriveData>();
Folder2.Add(new DriveData
{

```

```

NodeId = "02-01",
NodeText = "Personals"
});
Folder2.Add(new DriveData
{
  NodeId = "02-02",
  NodeText = "Projects"
});
Folder2.Add(new DriveData
{
  NodeId = "02-03",
  NodeText = "Office"
});
}
}

```

> Local Disk (C:)

▼ Local Disk (D:)

Personals

Projects

Office

MultiSelection in Blazor TreeView Component

Selection provides an interactive support and highlights the node that is selected. Selection can be done through simple mouse down or keyboard interaction. The TreeView also supports selection of multiple nodes by setting `AllowMultiSelection` to **true**. To multi-select, press and hold **CTRL** key and click the desired nodes. To select range of nodes, press and hold the **SHIFT** key and click the nodes.

In the following example, the `AllowMultiSelection` property is enabled.

Multi selection is not applicable through touch interactions.

ASPX-CS

```

@using Syncfusion.Blazor.Navigations
<SfTreeView TValue="Country" AllowMultiSelection=true>
  <TreeViewFieldsSettings TValue="Country" Id="Id" DataSource="@Countries"
  Text="Name" ParentID="ParentId" HasChildren="HasChild" Expanded="Expanded"
  Selected="IsSelected"></TreeViewFieldsSettings>
</SfTreeView>
@code{
public class Country
{
  public int Id { get; set; }
  public int? ParentId { get; set; }
  public string Name { get; set; }
}

```

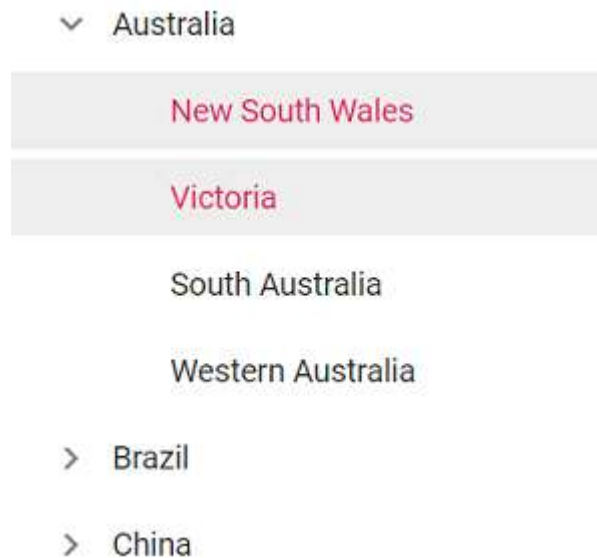


```
public bool HasChild { get; set; }
public bool Expanded { get; set; }
public bool IsSelected { get; set; }
}
List<Country> Countries = new List<Country>();
protected override void OnInitialized()
{
    base.OnInitialized();
    Countries.Add(new Country
    {
        Id = 1,
        Name = "Australia",
        HasChild = true,
        Expanded = true
    });
    Countries.Add(new Country
    {
        Id = 2,
        ParentId = 1,
        Name = "New South Wales",
        IsSelected = true
    });
    Countries.Add(new Country
    {
        Id = 3,
        ParentId = 1,
        Name = "Victoria",
        IsSelected = true
    });
    Countries.Add(new Country
    {
        Id = 4,
        ParentId = 1,
        Name = "South Australia"
    });
    Countries.Add(new Country
    {
        Id = 5,
        ParentId = 1,
        Name = "Western Australia"
    });
    Countries.Add(new Country
    {
        Id = 6,
        Name = "Brazil",
        HasChild = true
    });
    Countries.Add(new Country
    {
        Id = 7,
        ParentId = 6,
        Name = "Paraná"
    });
    Countries.Add(new Country
    {
        Id = 8,
        ParentId = 6,
```

```

Name = "Ceará"
});
Countries.Add(new Country
{
    Id = 9,
    Name = "China",
    HasChild = true
});
Countries.Add(new Country
{
    Id = 10,
    ParentId = 9,
    Name = "Guangzhou"
});
Countries.Add(new Country
{
    Id = 11,
    ParentId = 9,
    Name = "Shantou"
});
}
}

```



Drag and Drop in Blazor TreeView Component

The Blazor TreeView component allows to drag and drop any node by setting `AllowDragAndDrop` to **true**. Nodes can be dragged and dropped at all levels of the same TreeView.

The dragged nodes can be dropped at any level by indicator lines with **line**, **plus/minus**, and **restrict** icons. It represents the exact position where the node is to be dropped as sibling or child.

The following table explains the usage of indicator icons.

Icons	Description
----- -----	

- | Plus icon | Indicates that the dragged node is to be added as a child of target node. |
- | Minus or restrict icon | Indicates that the dragged node is not to be dropped at the hovered region. |
- | In between icon | Indicates that the dragged node is to be added as siblings of hovered region. |

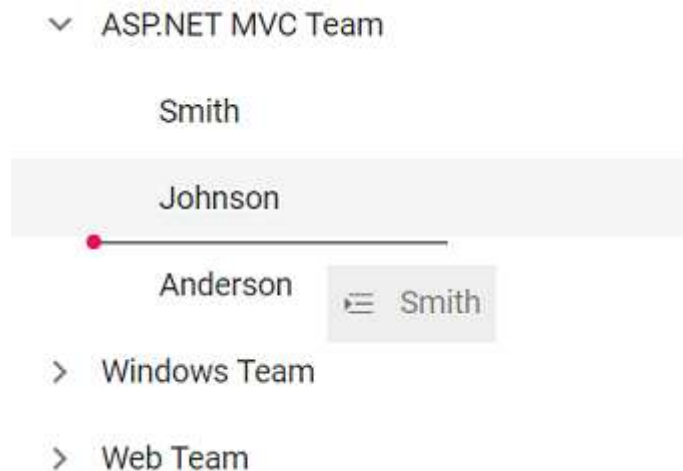
- In order to prevent dragging action for a particular node, the `OnNodeDragStart` event can be used which is triggered when the node drag is started. The `OnNodeDragged` event is triggered when the drag is stopped.
- The `NodeDropped` event is triggered when the TreeView node is dropped on the target element successfully.

In the `OnNodeDragged` event currently there is no option to cancel the event. However the other event arguments could be accessed.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<SfTreeView TValue="TeamDetails" AllowDragAndDrop="true">
  <TreeViewFieldsSettings TValue="TeamDetails" Id="Id" Text="Name"
  Child="Children" DataSource="@Team"
  Expanded="Expanded"></TreeViewFieldsSettings>
</SfTreeView>
@code{
public class TeamDetails
{
    public string Id { get; set; }
    public string Name { get; set; }
    public bool Expanded { get; set; }
    public bool Selected { get; set; }
    public List<TeamDetails> Children;
}
List<TeamDetails> Team = new List<TeamDetails>();
protected override void OnInitialized()
{
    base.OnInitialized();
    List<TeamDetails> EmployeeDetails = new List<TeamDetails>();
    Team.Add(new TeamDetails
    {
        Id = "01",
        Name = "ASP.NET MVC Team",
        Expanded = true,
        Children = EmployeeDetails,
    });
    EmployeeDetails.Add(new TeamDetails
    {
        Id = "01-01",
        Name = "Smith",
    });
    EmployeeDetails.Add(new TeamDetails
    {
        Id = "01-02",
        Name = "Johnson",
    });
    EmployeeDetails.Add(new TeamDetails
    {
```

```
Id = "01-03",
Name = "Anderson"
});
List<TeamDetails> EmployeeDetails1 = new List<TeamDetails>();
Team.Add(new TeamDetails
{
    Id = "02",
    Name = "Windows Team",
    Children = EmployeeDetails1,
});
EmployeeDetails1.Add(new TeamDetails
{
    Id = "02-01",
    Name = "Clark"
});
EmployeeDetails1.Add(new TeamDetails
{
    Id = "02-02",
    Name = "Wright"
});
List<TeamDetails> EmployeeDetails2 = new List<TeamDetails>();
Team.Add(new TeamDetails
{
    Id = "03",
    Name = "Web Team",
    Children = EmployeeDetails2,
});
EmployeeDetails2.Add(new TeamDetails
{
    Id = "03-01",
    Name = "Joshua"
});
EmployeeDetails2.Add(new TeamDetails
{
    Id = "03-02",
    Name = "Matthew"
});
}
}
```



Multiple-node drag and drop

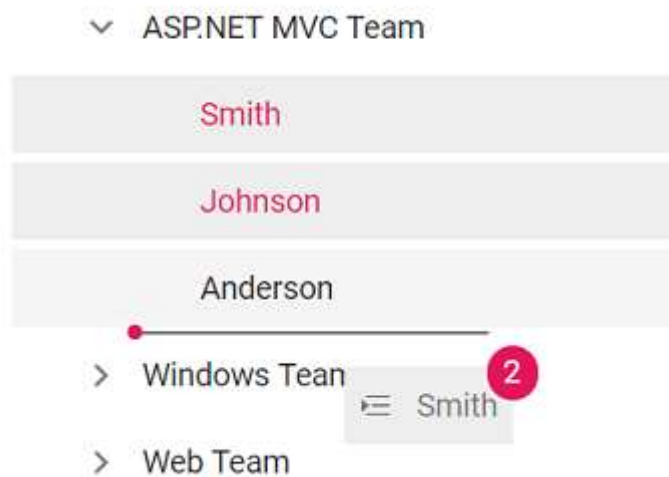
To drag and drop more than one node, enable the `AllowMultiSelection` property along with the `AllowDragAndDrop` property. To perform multi-selection, press and hold **CTRL** key and click the desired nodes. To select range of nodes, press and hold the **SHIFT** key and click the nodes.

In the following sample, the `AllowMultiSelection` property is enabled along with the `AllowDragAndDrop` property.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<SfTreeView TValue="TeamDetails" AllowDragAndDrop="true"
AllowMultiSelection="true">
<TreeViewFieldsSettings TValue="TeamDetails" Id="Id" Text="Name"
Selected="Selected" Child="Children" DataSource="@Team"
Expanded="Expanded"></TreeViewFieldsSettings>
</SfTreeView>
@code{
public class TeamDetails
{
public string Id { get; set; }
public string Name { get; set; }
public bool Expanded { get; set; }
public bool Selected { get; set; }
public List<TeamDetails> Children;
}
List<TeamDetails> Team = new List<TeamDetails>();
protected override void OnInitialized()
{
base.OnInitialized();
List<TeamDetails> EmployeeDetails = new List<TeamDetails>();
Team.Add(new TeamDetails
{
Id = "01",
Name = "ASP.NET MVC Team",
Expanded = true,
Children = EmployeeDetails,
});
};
```

```
EmployeeDetails.Add(new TeamDetails
{
    Id = "01-01",
    Name = "Smith",
    Selected = true
});
EmployeeDetails.Add(new TeamDetails
{
    Id = "01-02",
    Name = "Johnson",
    Selected = true
});
EmployeeDetails.Add(new TeamDetails
{
    Id = "01-03",
    Name = "Anderson"
});
List<TeamDetails> EmployeeDetails1 = new List<TeamDetails>();
Team.Add(new TeamDetails
{
    Id = "02",
    Name = "Windows Team",
    Children = EmployeeDetails1,
});
EmployeeDetails1.Add(new TeamDetails
{
    Id = "02-01",
    Name = "Clark"
});
EmployeeDetails1.Add(new TeamDetails
{
    Id = "02-02",
    Name = "Wright"
});
List<TeamDetails> EmployeeDetails2 = new List<TeamDetails>();
Team.Add(new TeamDetails
{
    Id = "03",
    Name = "Web Team",
    Children = EmployeeDetails2,
});
EmployeeDetails2.Add(new TeamDetails
{
    Id = "03-01",
    Name = "Joshua"
});
EmployeeDetails2.Add(new TeamDetails
{
    Id = "03-02",
    Name = "Matthew"
});
}
```



Accessibility in Blazor TreeView Component

The TreeView control has been designed keeping in mind the **WAI-ARIA** specifications, and applies WAI-ARIA roles, states, and properties along with **keyboard support**. This component is characterized by complete keyboard interaction support and ARIA accessibility support that makes it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation.

ARIA attributes

The TreeView control uses the **tree** role, and each tree parent node has a **group** role where each node has **treeitem** role. The following ARIA attributes are used in TreeView:

| Properties | Functionalities |

| --- | --- |

| aria-multiselectable | Indicates whether the TreeView enables multiple selection or not. |

| aria-expanded | Indicates whether the parent node has expanded or not. |

| aria-selected | Indicates the selected node. |

| aria-grabbed | Indicates the selected state on drag-and-drop of node. |

| aria-level | Indicates the level of node in TreeView. |

Keyboard interaction

The TreeView functionalities can be interactive when keyboard shortcuts are used.

TreeView supports the following keyboard shortcuts.

| Interaction Keys | Description |

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

| **Arrow Up** | Goes to the previous node. |

| **Arrow Down** | Goes to the next node. |

| **Arrow Right** | Expands the current node. |

- | Arrow Left | Collapses the current node. |
- | Home | Goes to the first node. |
- | End | Goes to the last node. |
- | F2 | Edits the focused node. |
- | Esc | Focuses out the edit state without saving the edited text. |
- | Enter | Selects the focused node/saves the edited text. |
- | Space | Checks the current node. |
- | Ctrl + A | Selects all nodes. |

Template in Blazor TreeView Component

The Blazor TreeView component allows to customize the look of TreeView nodes using the `NodeTemplate` property. The `NodeTemplate` tag is nested inside the `TreeViewTemplates` tag, where the custom structure for TreeView can be defined. Inside the `NodeTemplate` tag, a generic type context property is used to access the tree node details.

In the following sample, employee information such as employee photo, name, and designation have been included using the `NodeTemplate` property.

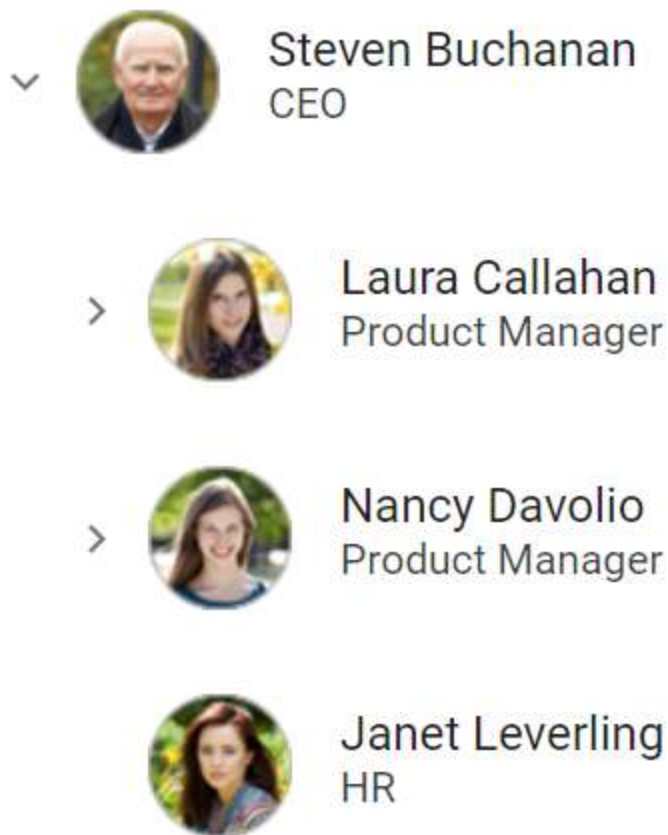
ASPX-CS

```
@using Syncfusion.Blazor.Navigations
@inject Microsoft.AspNetCore.Components.NavigationManager UriHelper
<SfTreeView TValue="EmployeeDetails" CssClass="custom">
  <TreeViewFieldsSettings TValue="EmployeeDetails" Id="EmployeeId"
  Text="EmployeeName" DataSource="@Employee" Expanded="Expanded"
  HasChildren="HasChild" Selected="Selected"
  ParentID="ParentId"></TreeViewFieldsSettings>
  <TreeViewTemplates TValue="EmployeeDetails">
    <NodeTemplate>
      @{
        var employee = ((context as EmployeeDetails));
        
        <div class="ename">@((@context as EmployeeDetails).EmployeeName)</div>
        <div class="ejob">@((@context as EmployeeDetails).Designation)</div>
      }
    </NodeTemplate>
  </TreeViewTemplates>
</SfTreeView>
@code
{
  public class EmployeeDetails
  {
    public string EmployeeName { get; set; }
    public int EmployeeId { get; set; }
    public int? ParentId { get; set; }
    public bool HasChild { get; set; }
    public bool Expanded { get; set; }
    public bool Selected { get; set; }
    public string Image { get; set; }
  }
}
```



```
public string Designation { get; set; }
}
List<EmployeeDetails> Employee = new List<EmployeeDetails>();
protected override void OnInitialized()
{
    base.OnInitialized();
    Employee.Add(new EmployeeDetails
    {
        EmployeeId = 1,
        EmployeeName = "Steven Buchanan",
        HasChild = true,
        Expanded = true,
        Image = "10",
        Designation = "CEO"
    });
    Employee.Add(new EmployeeDetails
    {
        EmployeeId = 2,
        ParentId = 1,
        EmployeeName = "Laura Callahan",
        Image = "2",
        Designation = "Product Manager",
        HasChild = true
    });
    Employee.Add(new EmployeeDetails
    {
        EmployeeId = 3,
        ParentId = 2,
        EmployeeName = "Andrew Fuller",
        Image = "7",
        Designation = "Team Lead",
        HasChild = true
    });
    Employee.Add(new EmployeeDetails
    {
        EmployeeId = 4,
        ParentId = 3,
        EmployeeName = "Anne Dodsworth",
        Image = "1",
        Designation = "Developer"
    });
    Employee.Add(new EmployeeDetails
    {
        EmployeeId = 5,
        ParentId = 1,
        EmployeeName = "Nancy Davolio",
        HasChild = true,
        Image = "4",
        Designation = "Product Manager"
    });
    Employee.Add(new EmployeeDetails
    {
        EmployeeId = 6,
        ParentId = 5,
        EmployeeName = "Michael Suyama",
        Image = "9",
        Designation = "Team Lead",
```

```
HasChild = true
});
Employee.Add(new EmployeeDetails
{
    EmployeeId = 7,
    ParentId = 6,
    EmployeeName = "Robert King",
    Image = "8",
    Designation = "Developer"
});
Employee.Add(new EmployeeDetails
{
    EmployeeId = 8,
    ParentId = 7,
    EmployeeName = "Margaret Peacock",
    Image = "6",
    Designation = "Developer"
});
Employee.Add(new EmployeeDetails
{
    EmployeeId = 9,
    ParentId = 1,
    EmployeeName = "Janet Leverling",
    Image = "3",
    Designation = "HR"
});
}
}
<style>
.custom.e-treeview .e-fullrow {
height: 72px;
}
.custom.e-treeview .e-list-text {
line-height: normal;
}
.eimage {
float: left;
padding: 11px 16px 11px 0;
height: 48px;
width: 48px;
box-sizing: content-box;
}
.ename {
font-size: 16px;
padding: 14px 0 0;
}
.ejob {
font-size: 14px;
opacity: .87;
}
</style>
```



How To

Customize the expand and collapse icons in Blazor TreeView Component

TreeView expand and collapse icons could be customized by using the `CssClass` property of TreeView. Refer to the sample to customize expand or collapse icons.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<SfTreeView TValue="DriveData" CssClass="custom">
  <TreeViewFieldsSettings TValue="DriveData" Id="NodeId" Text="NodeText"
  Child="Children" DataSource="@Drive"
  Expanded="Expanded"></TreeViewFieldsSettings>
</SfTreeView>
@code{
public class DriveData
{
    public string NodeId { get; set; }
    public string NodeText { get; set; }
    public bool Expanded { get; set; }
    public bool Selected { get; set; }
    public List<DriveData> Children;
}
List<DriveData> Drive = new List<DriveData>();
protected override void OnInitialized()
{

```

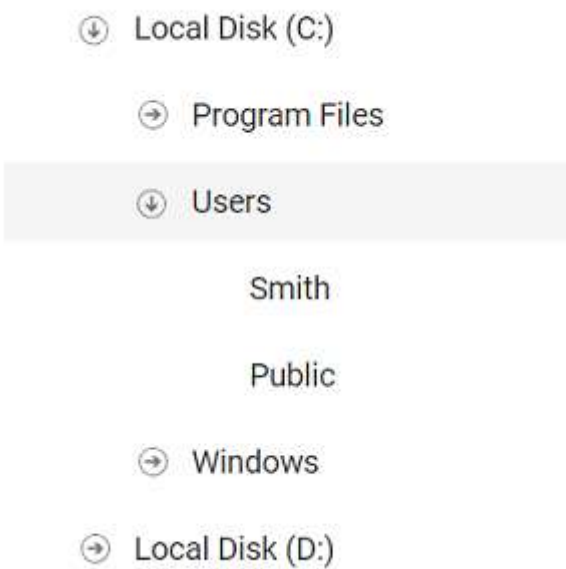
```
base.OnInitialized();
List<DriveData> Folder1 = new List<DriveData>();
Drive.Add(new DriveData
{
    NodeId = "01",
    NodeText = "Local Disk (C:)",
    Expanded = true,
    Children = Folder1,
});
List<DriveData> File1 = new List<DriveData>();
Folder1.Add(new DriveData
{
    NodeId = "01-01",
    NodeText = "Program Files",
    Children = File1
});
File1.Add(new DriveData
{
    NodeId = "01-01-01",
    NodeText = "Windows NT"
});
List<DriveData> File2 = new List<DriveData>();
Folder1.Add(new DriveData
{
    NodeId = "01-02",
    NodeText = "Users",
    Expanded = true,
    Children = File2
});
File2.Add(new DriveData
{
    NodeId = "01-02-01",
    NodeText = "Smith"
});
File2.Add(new DriveData
{
    NodeId = "01-02-02",
    NodeText = "Public"
});
List<DriveData> File3 = new List<DriveData>();
Folder1.Add(new DriveData
{
    NodeId = "01-03",
    NodeText = "Windows",
    Children = File3
});
File3.Add(new DriveData
{
    NodeId = "01-03-01",
    NodeText = "Boot"
});
List<DriveData> Folder2 = new List<DriveData>();
Drive.Add(new DriveData
{
    NodeId = "02",
    NodeText = "Local Disk (D:)",
    Children = Folder2,
```

```

});
Folder2.Add(new DriveData
{
    NodeId = "02-01",
    NodeText = "Personals"
});
Folder2.Add(new DriveData
{
    NodeId = "02-02",
    NodeText = "Projects"
});
}
}

<style>
.custom.e-list-item.e-icons {
font-family: "Customize-icon";
}
.custom.e-treeview.e-list-item.e-icon-expandable::before, .custom.e-
treeview.e-list-item.e-icon-collapsible:before {
content: '\e700';
font-size: 12px;
}
@@font-face {
font-family: 'Customize-icon';
src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgTlMvMj0gSRcAAAEoAAAAVmNtYXdEODaAAABjAAAADhnbHlmcYq
IngAAAcwAAAD8aGVhZBWT124AAADQAAAAANmhoZWEHmANtAAAArAAAACRobXR4C9AAAAAAYAAAA
MbG9jYQBAAH4AAAHEAAAACG1heHABEAAxAAABCAAAACBuYW1l/qscPAAAAsgAAAJ5cG9zdIPGFvo
AAAVEAAAAVgABAAADUv9qAFoEAAAA//8D6QABAAAAAAAAAAAAAAAAAAwABAAAAQAAlKcGUl8
PPPUACwPoAAAAANlGSVAAAAAA2UZJUAAAAAAD6QPpAAACAACAAAAAAAAAAAAAADACUAAwAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQpWAZAABQAAANoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAQNS/2oAWgPpAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAAAAAAAgAAAAAUAAMAAQAAABQABAaKAAAAABAAEAAEA0cB//8AA0cA//8AAAAABAAQAAAA
BAAIAAAAAEAafgADAAAAAPpA+kACAACWACQAAAEhFSEHMzcnIyUWEAcGICcmEDc+ATIWBQYQFXY
gNzYQJy4BIgYCMf6kAWqUqMK8rgF+goKK/qCEfn5Coquf/amRkZoBkpqRkUq3xLcCKmSTybt4if6
ghYKChQFgiUJBQRma/m6akZGaAZKaSElJAAMAAAAA+gD6QAGABQAIgAAASMXNyMRIyUWEAcGICc
mEDc+ATIWBQYQFXYgNzYQJy4BIgYBvrLp6JmGAW6BgYf+oYiBgUGhqH9qZOTmgGOMPOTSrbCtgG
y6kBBwU/qGHgYGIaV6IQEFBFpr+cZmTk5oBj5lKSUKAAAAAABIA3gABAAAAAAAAAAAAEAAAABAAA
AAAAABAA4AAQABAAAAAAACAAcADwABAAAAAADAA4AFgABAAAAAAAEAA4AJAABAAAAAAFAAASAMgA
BAAAAAAGAA4APQABAAAAAAAKACwASwABAAAAAALABIAdwADAAEECQAAAAIAiQADAAEECQABABw
AiwADAAEECQACAA4ApwADAAEECQADABwAtQADAAEECQAEABwA0QADAAEECQAFABYA7QADAAEECQA
GABwBAwADAAEECQAKAFgBHwADAAEECQALACQBdyBDdXN0b21pemUtaWNvb1JlZ3VsYXJDDdXN0b21
pemUtaWNvb1JlZ3VsYXJDDdXN0b21pemUtaWNvb1JlZ3VsYXJDDdXN0b21pemUtaWNvb1JlZ3VsYX
hdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3RlZGlvZ3d3LnN5bmNmdXNpb24uY29tACAAQwB
1AHMAdABvAG0AaQB6AGUALQBpAGMABwBuAFIAZQBnAHUAbABhAHIAQwB1AHMAdABvAG0AaQB6AGU
ALQBpAGMABwBuAEMAdQBzAHQAbwBtAGkAegBlAC0AaQBjAG8AbgBWAGUAcgBzAGkAbwBuACAAMQA
uADAAQwB1AHMAdABvAG0AaQB6AGUALQBpAGMABwBuAEYAbwBuAHQAIAIBnAGUAbgBlAHIAIYQB0AGU
AZAAgAHUAcwBpAG4AZwAgAFMAeQBwAGMAZgB1AHMAaQBvAG4AIABNAGUAdABYAG8AIABTAHQAQB
kAGkAbwB3AHcAdwAuAHMAeQBwAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAMBAGEDAQQAfYlhcncJvdyljaXJjbGUTcmlnaHQQtXzAxXzE1hcncJ
vdyljaXJjbGUTZG93bgAAAAA=) format('trueType');
font-weight: normal;
font-style: normal;
}
</style>

```



Customize expand/collapse icon's position and color

The expand or collapse icon's position and color can also be customized using the following CSS.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<div id="treeview">
<SfTreeView TValue="TreeViewData" CssClass="custom-tree">
<TreeViewFieldsSettings Id="NodeId" ParentID="ParentId"
DataSource="@ListData" Text="NodeText" HasChildren="HasChild"
Expanded="Expanded" IconCss="IconCss"></TreeViewFieldsSettings>
</SfTreeView>
</div>
@code
{
public class TreeViewData
{
public string NodeId { get; set; }
public string NodeText { get; set; }
public string ParentId { get; set; }
public bool HasChild { get; set; }
public string IconCss { get; set; }
public bool Expanded { get; set; }
}
List<TreeViewData> ListData = new List<TreeViewData>();
protected override void OnInitialized()
{
base.OnInitialized();
ListData = new List<TreeViewData>();
ListData.Add(new TreeViewData
{
NodeId = "1",
NodeText = "Installation",
IconCss = "icon-microchip icon"
});
};
```

```
ListData.Add(new TreeViewData
{
    NodeId = "2",
    NodeText = "Deployment",
    IconCss = "icon-thumbs-up-alt icon"
});
ListData.Add(new TreeViewData
{
    NodeId = "3",
    NodeText = "Quick Start",
    IconCss = "icon-docs icon"
});
ListData.Add(new TreeViewData
{
    NodeId = "4",
    NodeText = "Components",
    HasChild = true,
    IconCss = "icon-th icon",
    Expanded = true
});
ListData.Add(new TreeViewData
{
    NodeId = "5",
    ParentId = "4",
    NodeText = "Calendar",
    IconCss = "icon-circle-thin icon"
});
ListData.Add(new TreeViewData
{
    NodeId = "7",
    ParentId = "4",
    IconCss = "icon-circle-thin icon",
    NodeText = "DatePicker",
});
ListData.Add(new TreeViewData
{
    NodeId = "8",
    ParentId = "4",
    IconCss = "icon-circle-thin icon",
    NodeText = "DateTimePicker",
});
ListData.Add(new TreeViewData
{
    NodeId = "9",
    NodeText = "API Reference",
    HasChild = true,
    IconCss = "icon-code icon",
});
ListData.Add(new TreeViewData
{
    NodeId = "10",
    ParentId = "9",
    NodeText = "Calendar",
    IconCss = "icon-circle-thin icon"
});
ListData.Add(new TreeViewData
{

```

```

NodeId = "11",
ParentId = "9",
NodeText = "DatePicker",
IconCss = "icon-circle-thin icon"
});
ListData.Add(new TreeViewData
{
    NodeId = "12",
    IconCss = "icon-chrome icon",
    NodeText = "Browser Compatibility",
});
ListData.Add(new TreeViewData
{
    NodeId = "13",
    NodeText = "Upgrade Packages",
    IconCss = "icon-up-hand icon"
});
ListData.Add(new TreeViewData
{
    NodeId = "14",
    NodeText = "FAQ",
    IconCss = "icon-help-circled icon"
});
}
}

<style>
/*To apply border and background color for treeview*/
#treeview {
    max-width: 400px;
    margin: auto;
    border: 1px solid #dddddd;
    border-radius: 3px;
    background: #1c86c8;
}
/* customize icon styles */
.e-treeview.custom-tree .e-list-icon {
    font-family: 'fontello';
    font-size: 16px;
    margin-top: -4px;
    color: white;
}
/* icon styles */
@@font-face {
    font-family: 'fontello';
    src: url('data:application/octet-
stream;base64,AAEAAAAPAIAAAwBwR1NVQiCLJXoAAAD8AAAAVE9TLzI+JUkyAAABUAAAFZjbW
Fw0almQAAAAagAAAIgY3Z0IAbV/vwAABfUAAAAIGZwZ22KkZBZAAAX9AAAC3BnYXNwAAAAEAAAF8
wAAAAIZ2x5Zk30JrMAAAPIAAAPrGhlyWQTW6AfAAATdAAAADZoaGVhB2gDnAAAE6wAAAAkaG10eD
Hm//YAABPQAAAAOGxvY2EejhqYAAAUCAAAAB5tYXhwAfYMkAAAFcGgAAAAGbmFtZcydHiAAABRIA
ACzXBvc3RuKDzPAAAXGAAALRwcmVw5UErvAAAI2QAAACGAAEAAAAKADAAPgACREZMVAAObGF0bg
AaaaQAAAAAAAAAAAAQAAAAAAAAAAAAQAAAFsaWdhAAgAAAABAAAAAQEAAQAAAAABAagAAQAGAA
AAAQAAAAEDkAGQAAUAAAJ6ArwAAACMAAnoCvAAAAeAAMQECAAACAAUDAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAFBmRWQAQOgB6BMDUv9qAFoDUgCaAAAAAQAAAAAAAAAAAAUAAADAAAAALAAAAQAAAF0AA
EAAAAAG4AAwABAAAAAALADAAoAAAF0AAQAQgAAAAAYABAABAAALoCegT//8AAOgB6BD//wAAAAAAQ
AGABYAAAAABAAIAAwAEAAUABgAHAAGACQAKAAsADAANAAABBgAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

```


Copyright © 2001 - 2022 Syncfusion Inc.

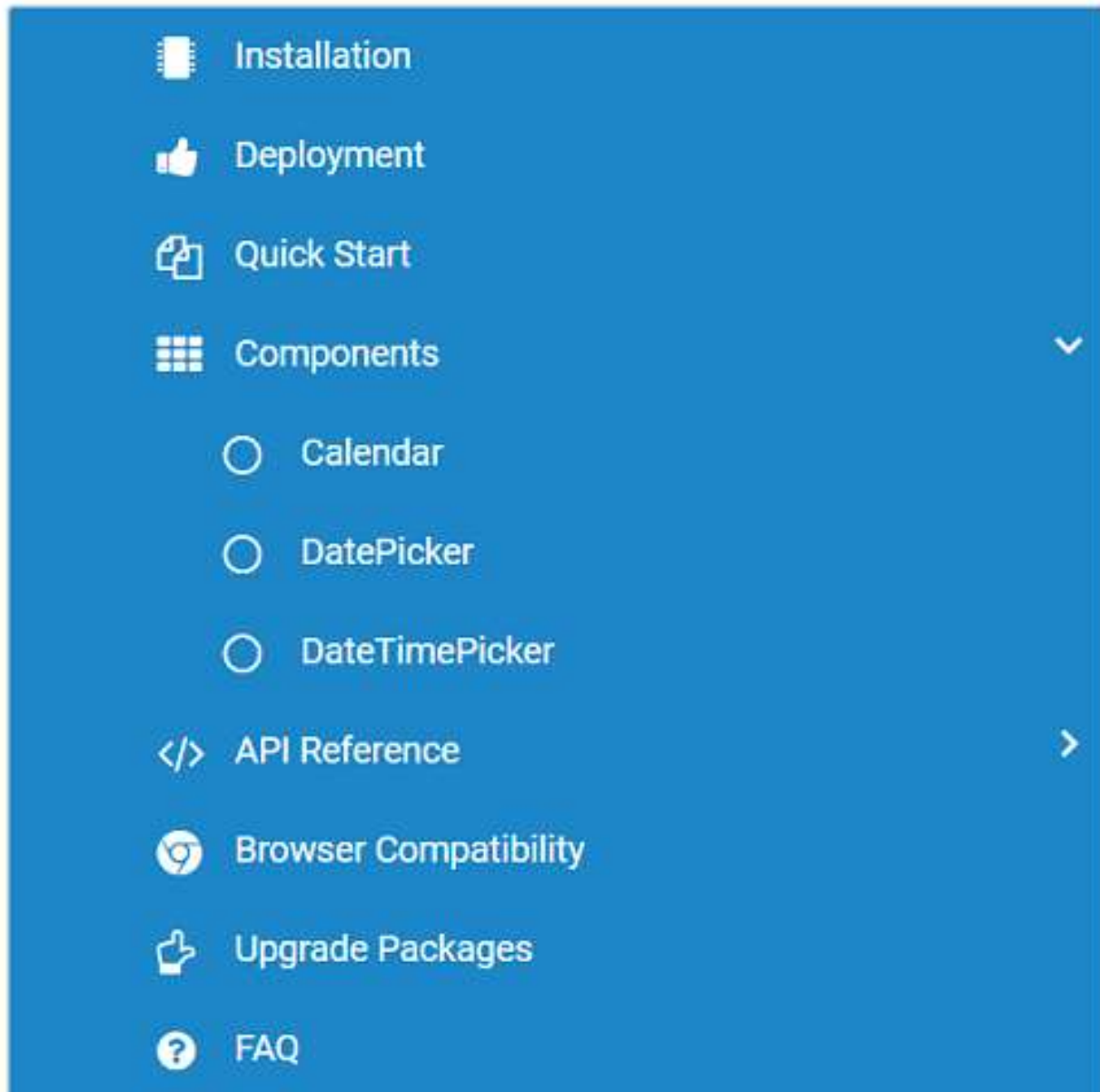
2062

```

IgYLABYbUQEAEADgBCQopgsRIGK7ByKxsiWS2wHyxxAB4rLbAgLLEBHistsCEssQIeKy2wIiyxAx
4rLbAjLLEEHistscQssQeKy2wJSyxBh4rLbAmLLEHHistsCcscQgeKy2wKCyxCR4rLbApLCA8sA
FgLbAqLCBgSBBgIEMjsAFgQ7ACJWGwAWCwKSohLbArLLAqK7AqKi2wLCwgIEcgILALQ2O4BABiIL
AAUFiWQGBZzrABY2AjYtGjIIPVWCBHICcCwC0NjuAQAYiCwAFBYsEBgWWawAWNgi2E4GyFZLbAtLA
CxAAJFVFiwARawLCqWARUwGyJZLbAuLACwDSuxAAJFVFiwARawLCqWARUwGyJZLbAvLCA1sAFgLb
AwLACwAUvjuAQAYiCwAFBYsEBgWWawAWOWASuWC0NjuAQAYiCwAFBYsEBgWWawAWOWASuWABa0AA
AAAABEPiM4sS8BFSotsDEsIDwgRyCwC0NjuAQAYiCwAFBYsEBgWWawAWNgsABDYTgttsDIshc8Lb
AzLCA8IEcgsAtDY7gEAGIGsABQWLBAYFlmsAFjYLAQ2GwAUNjOC2wNCyxAgAWJSauIEewACNCsA
ILSYqKRYNHI2EgWGIBiVmwASNCSjMBARUUKi2wNSywABawBCWwBCVHI0cjYbAJQytlii4jICA8ij
gtsDYssAAWsAQ1sAQ1IC5HI0cjYSCwBCNCsAlDKyCwYFBIYILBAUVizAiADIBuzAiYDG1lCQiMgsA
hDIIOjRyNHI2EjRmCwBEOWAmIgsABQWLBAYFlmsAFjYCCwASSgiophILACQ2Bki7ADQ2FkUFIwAk
NhG7ADQ2BZsAMlsAJiILAAUFiWQGBZzrABY2EjICCwBCYjRme4GyOwCENGsAilshDRyNHI2FgIL
AEQ7ACYiCwAFBYsEBgWWawAWNgiYcWASSjsARDYLAK7AFJWGwBSWwAmIgsABQWLBAYFlmsAFjsA
QmYSCwBCVgZCOWAyVgZFBYIRsjIVkjICCwBCYjRme4WS2wNyywABYgICCwBSYgLkcjRyNhiZw4Lb
A4LLAAFiCwCCNCICAgRiNHsAErI2E4LbA5LLAAFrADJbACJUcjRyNhsABUWC4gPCMhG7ACJbACJU
cjRyNhiLAFJbAEJUcjRyNhsAYlsAUlSbACJWG5CAAiAGNjIyBYyhshWWO4BABiILAAUFiWQGBZzr
ABY2AjLiMgIDyKOCmHsW2wOiywABYgsAhDIC5HI0cjYSBgsCBgZrACYiCwAFBYsEBgWWawAWMjIC
A8ijgtsDssIyAuRRACJUZSWCA8WS6xKwEUKy2wPCWjIC5GsAilRlBYIDxZLrErARQrLbA9LCMgLk
awAiVGULggPFkjIC5GsAilRlBYIDxZLrErARQrLbA+LLA1KyMgLkAwAiVGULggPFkusSsBFCstsD
8ssDYriiAgPLAEI0KKOCMgLkAwAiVGULggPFkusSsBFCwBEMusCsrLbBALLAAFrAEJbAEJiAuRy
NHI2GwCUMrIyA8IC4jOLErARQrLbBBLLEIBCVCSAAWsAQ1sAQ1IC5HI0cjYSCwBCNCsAlDKyCwYF
BYILBAUVizAiADIBuzAiYDG1lCQiMgR7AEQ7ACYiCwAFBYsEBgWWawAWNgiLABKyCKimEgsAJDYG
QjsANDYWRQWLACQ2EbsANDYFmwAyWwAmIgsABQWLBAYFlmsAFjYbACJUZhOCMgPCM4GyEgIEYjR7
ABKyNhOCFZsSsBFCstsEIssDUrLrErARQrLbBDLLA2KyEjICA8sAQjQiM4sSsBFCwBEMusCsrLb
BELLAAFSBHsAAjQrIAAQEVFBMusDEqLbBFLLAAFSBHsAAjQrIAAQEVFBMusDEqLbBGLLEAARQTsD
IqLbBHLLA0Ki2wSCyWABZFIyAuIEaKI2E4sSsBFCstsEkssAgjQrBIKy2wSiyyAABBKy2wSyyyAA
FBKy2wTCyyAQBBKy2wTSyyAQFBKy2wTiyyAABCKy2wTyyyAAFCKy2wUCyyAQBCKy2wUSyyAQFCKy
2wUiyyAAA+Ky2wUyyyAAE+Ky2wVCyyAQA+Ky2wVSyyAQE+Ky2wViyyAABAKy2wVyyyAAFAKy2wWC
yyAQBAKy2wWSyyAQFAKy2wWiyyAABDKy2wWyyyAAFDKy2wXCyyAQBDKy2wXSyyAQFDKy2wXiyyAA
A/Ky2wXyyyAAE/Ky2wYCyyAQA/Ky2wYSyyAQE/Ky2wYiywNysusSsBFCstsGMssDcrsDsrLbBkLL
A3K7A8Ky2wZSywABawNyuwPSstsGYssDgrLrErARQrLbBnLLA4K7A7Ky2waCywOCuwPCstsGkssD
grsD0rLbBqLLA5Ky6xKwEUKy2wayywOSuwOystsGwssDkrsDwrLbBtLLA5K7A9Ky2wbiywOisusS
sBFCstsG8ssDorsDsrLbBwLLA6K7A8Ky2wcSywOiuwPSstsHIsswKEAgNFWCEbIyFZQiuwCGWwAy
RQeLABFTAtAEu4AMhSWLEBAY5ZsAG5CAAiAGNwsQAFQrIAAQAsQAFQrMKAgeIKrEABUKzDgABCC
qxAAZCugLAAAEACsQxAAdCugBAAAEACsQxAwBESQBIFFYsECIWLEDZESxJgGIUVi6CIAAAQRAiG
NUWLEDAERZwV1ZswwCAQwquAH/hbAEjBECAEQAAA==') format('truetype');
}
.e-treeview.custom-tree .e-list-icon.icon-microchip::before {
content: '\e806';
}
.e-treeview.custom-tree .e-list-icon.icon-thumbs-up-alt::before {
content: '\e805';
}
.e-treeview.custom-tree .e-list-icon.icon-chrome::before {
content: '\e807';
}
.e-treeview.custom-tree .e-list-icon.icon-up-hand::before {
content: '\e810';
}
.e-treeview.custom-tree .e-list-icon.icon-docs::before {
content: '\e802';
}
.e-treeview.custom-tree .e-list-icon.icon-th::before {
content: '\e803';
}
.e-treeview.custom-tree .e-list-icon.icon-code::before {
content: '\e804';
}

```

```
}
.e-treeview.custom-tree .e-list-icon.icon-help-circled::before {
content: '\e813';
}
.e-treeview.custom-tree .e-list-icon.icon-circle-thin::before {
content: '\e808';
}
/*To float the expand/collapse icon right*/
.e-treeview.custom-tree .e-icon-collapsible, .e-treeview .e-icon-expandable
{
float: right;
}
/*To customize the expand collapse icon color*/
.e-treeview.custom-tree .e-icon-collapsible::before, .e-treeview.custom-tree
.e-icon-expandable::before {
color: white;
}
/*To change the text color for treeview*/
.e-treeview .e-text-content > .e-list-text {
color: white;
}
</style>
```



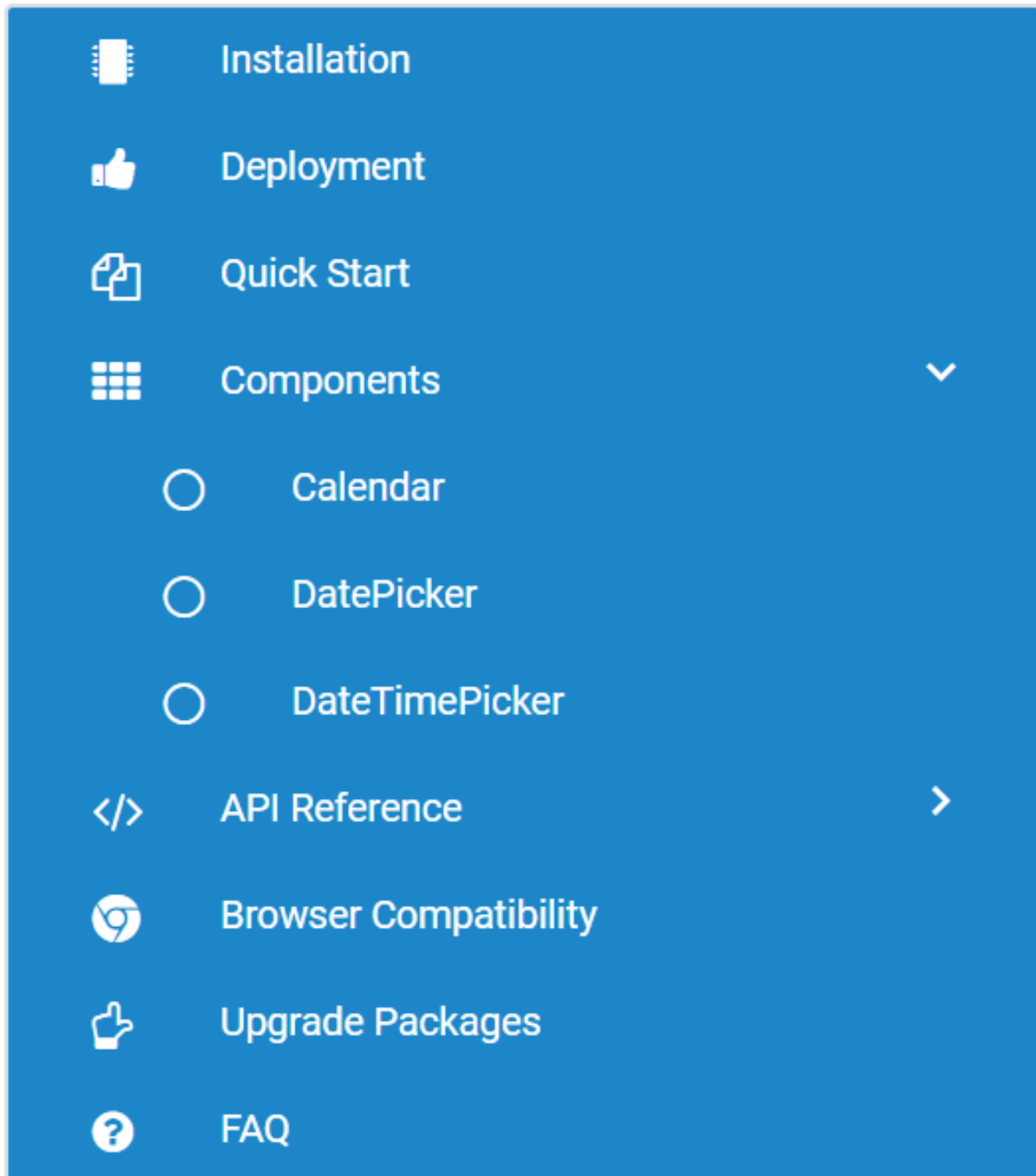
Increase the padding between the text, expand/collapse icon and custom icons

The padding between the text, expand or collapse icon and custom icons can be increased using the following CSS in the above sample.

CSHARP

```
/* customize icon styles */
.e-treeview.custom-tree .e-list-icon {
font-family: 'fontello';
font-size: 16px;
margin-top: -4px;
color: white;
/*To increase the padding between icon and text*/
margin-left: -20px;
}
```

```
/*To float the expand/collapse icon right*/
.e-treeview.custom-tree .e-icon-collapsible, .e-treeview .e-icon-expandable
{
  float: right;
/*To increase the space between the text and chevron*/
  margin-right: 20px;
}
/*To change the text color for treeview*/
.e-treeview .e-text-content > .e-list-text {
  color: white;
/*To increase the padding between the text*/
  padding-left: 25px;
}
```



Customize the tree nodes based on levels in Blazor TreeView Component

The tree nodes can be customized level wise by adding custom CssClass to the component and enabling styles.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations  
<SfTreeView TValue="DriveData" CssClass="mytree">
```

```

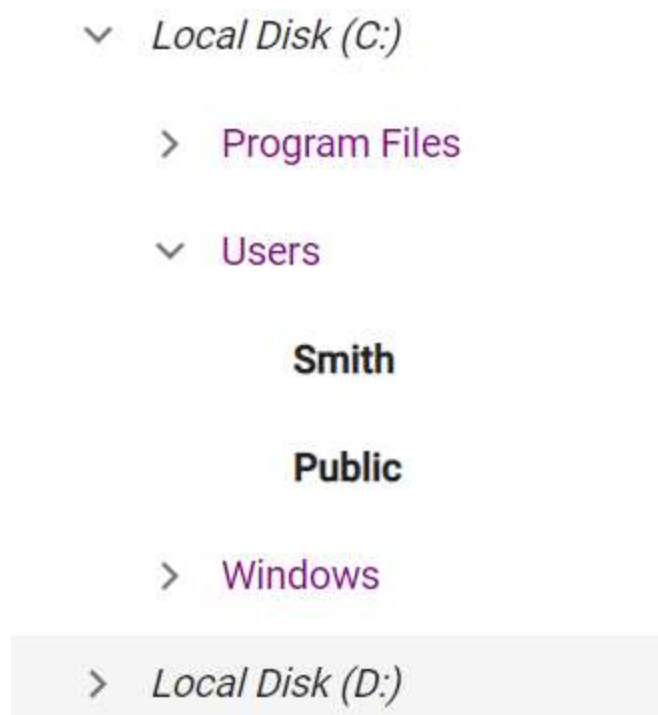
<TreeViewFieldsSettings TValue="DriveData" Id="NodeId" Text="NodeText"
Child="Children" DataSource="@Drive"
Expanded="Expanded"></TreeViewFieldsSettings>
</SfTreeView>
<div id="details">
<label>Note:</label>
<div><b>1. The font-weight "Bold" is applied for all the leaf
nodes</b></div>
<div><i>2. The font-weight "Italic" is applied for first level
nodes</i></div>
<div style="color: darkmagenta">3. The color "darkmagenta" is applied for
second level nodes</div>
</div>
@code{
public class DriveData
{
public string NodeId { get; set; }
public string NodeText { get; set; }
public bool Expanded { get; set; }
public bool Selected { get; set; }
public List<DriveData> Children;
}
object Child;
List<DriveData> Drive = new List<DriveData>();
protected override void OnInitialized()
{
base.OnInitialized();
List<DriveData> Folder1 = new List<DriveData>();
Drive.Add(new DriveData
{
NodeId = "01",
NodeText = "Local Disk (C:)",
Expanded = true,
Children = Folder1,
});
List<DriveData> File1 = new List<DriveData>();
Folder1.Add(new DriveData
{
NodeId = "01-01",
NodeText = "Program Files",
Children = File1
});
File1.Add(new DriveData
{
NodeId = "01-01-01",
NodeText = "Windows NT"
});
List<DriveData> File2 = new List<DriveData>();
Folder1.Add(new DriveData
{
NodeId = "01-02",
NodeText = "Users",
Expanded = true,
Children = File2
});
File2.Add(new DriveData
{

```



```
NodeId = "01-02-01",
NodeText = "Smith"
});
File2.Add(new DriveData
{
    NodeId = "01-02-02",
    NodeText = "Public"
});
List<DriveData> File3 = new List<DriveData>();
Folder1.Add(new DriveData
{
    NodeId = "01-03",
    NodeText = "Windows",
    Children = File3
});
File3.Add(new DriveData
{
    NodeId = "01-03-01",
    NodeText = "Boot"
});
List<DriveData> Folder2 = new List<DriveData>();
Drive.Add(new DriveData
{
    NodeId = "02",
    NodeText = "Local Disk (D:)",
    Children = Folder2,
});
Folder2.Add(new DriveData
{
    NodeId = "02-01",
    NodeText = "Personals"
});
Folder2.Add(new DriveData
{
    NodeId = "02-02",
    NodeText = "Projects"
});
this.Child = "Children";
}
}

<style>
.mytree .e-level-1 > .e-text-content .e-list-text {
font-style: italic;
}
.mytree .e-level-2 > .e-text-content .e-list-text {
color: darkmagenta;
}
.mytree .e-level-3 > .e-text-content .e-list-text {
font-weight: bold;
}
</style>
```



Process the Blazor TreeView node operations using context menu

The context menu can be integrated with **TreeView** component in order to perform the TreeView related operations like add, remove and renaming node.

Following example demonstrates the above cases which are used to manipulate TreeView operations in the **ItemSelected** event of context menu.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<div id="treeview">
  <SfTreeView TValue="EmployeeData" @ref="tree" AllowDragAndDrop="true" @bind-
  SelectedNodes="@selectedNodes" @bind-ExpandedNodes="expandedNodes">
    <TreeViewFieldsSettings Id="Id" ParentID="Pid" DataSource="@ListData"
    Text="Name" HasChildren="HasChild"></TreeViewFieldsSettings>
    <TreeViewEvents TValue="EmployeeData" NodeSelected="OnSelect"
    NodeClicked="nodeClicked"></TreeViewEvents>
    <SfContextMenu TValue="MenuItem" @ref="menu" Target="#treeview"
    Items="@MenuItems">
      <MenuEvents TValue="MenuItem" ItemSelected="MenuSelect"></MenuEvents>
    </SfContextMenu>
  </SfTreeView>
</div>
@code
{
  // Reference for treeview
  SfTreeView<EmployeeData> tree;
  // Reference for context menu
  SfContextMenu<MenuItem> menu;
  string selectedId;
  public string[] selectedNodes = Array.Empty<string>();
  public string[] expandedNodes = new string[] { "" };
```

```

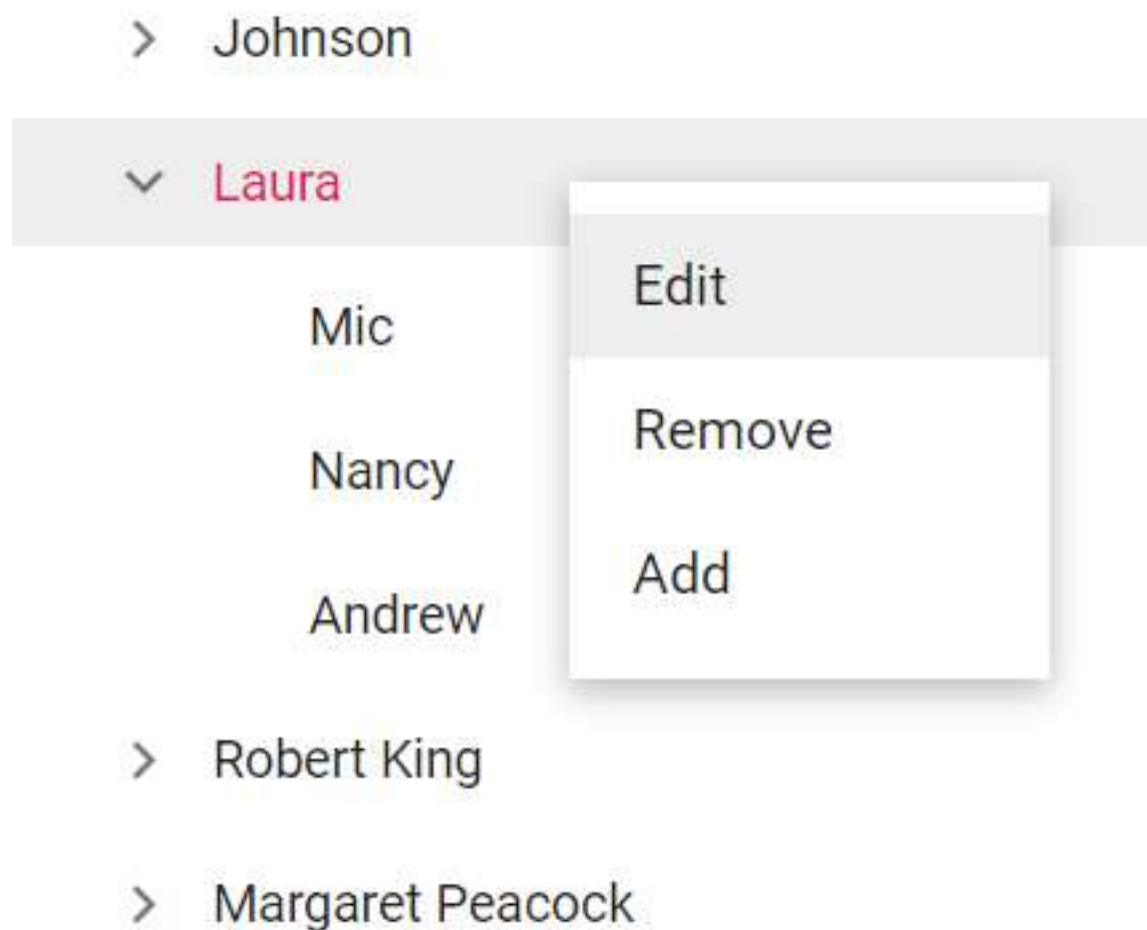
int index = 100;
// Datasource for menu items
public List<MenuItem> MenuItems = new List<MenuItem>{
    new MenuItem { Text = "Edit" },
    new MenuItem { Text = "Remove" },
    new MenuItem { Text = "Add" }
};
public class EmployeeData
{
    public string Id { get; set; }
    public string Name { get; set; }
    public string Pid { get; set; }
    public bool HasChild { get; set; }
}
// Triggers when TreeView Node is selected
public void OnSelect(NodeSelectEventArgs args)
{
    this.selectedId = args.NodeData.Id;
}
// Triggers when TreeView node is clicked
public void nodeClicked(NodeClickEventArgs args)
{
    selectedId = args.NodeData.Id;
    selectedNodes = new string[] { args.NodeData.Id };
}
// To add a new node
async Task AddNodes()
{
    // Expand the selected nodes
    expandedNodes = new string[] { this.selectedId };
    string NodeId = "tree_" + this.index.ToString();
    ListData.Add(new EmployeeData
    {
        Id = NodeId,
        Name = "NewItem",
        Pid = this.selectedId
    });
    await Task.Delay(100);
    // Edit the added node.
    await this.tree.BeginEditAsync(NodeId);
    this.index = this.index + 1;
}
// To delete a tree node
void RemoveNodes()
{
    List<EmployeeData> removeNode = tree.GetTreeData(selectedId);
    ListData.Remove(removeNode.ElementAt(0));
}
// To edit a tree node
async Task RenameNodes()
{
    await this.tree.BeginEdit(this.selectedId);
}
// Triggers when context menu is selected
public async Task MenuSelect(MenuEventArgs<MenuItem> args)
{
    string selectedText;

```

```
selectedText = args.Item.Text;
if (selectedText == "Edit")
{
    await this.RenameNodes();
}
else if (selectedText == "Remove")
{
    this.RemoveNodes();
}
else if (selectedText == "Add")
{
    await this.AddNodes();
}
}
// local data source
List<EmployeeData> ListData = new List<EmployeeData>();
protected override void OnInitialized()
{
    selectedNodes = new string[] { "1" };
    base.OnInitialized();
    ListData = new List<EmployeeData>();
    ListData.Add(new EmployeeData
    {
        Id = "1",
        Name = "Johnson",
        HasChild = true,
    });
    ListData.Add(new EmployeeData
    {
        Id = "2",
        Pid = "1",
        Name = "Sourav",
    });
    ListData.Add(new EmployeeData
    {
        Id = "3",
        Pid = "1",
        Name = "Sanjay",
    });
    ListData.Add(new EmployeeData
    {
        Id = "4",
        Pid = "1",
        Name = "Steve",
    });
    ListData.Add(new EmployeeData
    {
        Id = "6",
        Pid = "1",
        Name = "Martin",
    });
    ListData.Add(new EmployeeData
    {
        Id = "7",
        Name = "Laura",
        HasChild = true,
    });
}
```

```
ListData.Add(new EmployeeData
{
    Id = "8",
    Pid = "7",
    Name = "Mic",
});
ListData.Add(new EmployeeData
{
    Id = "9",
    Pid = "7",
    Name = "Nancy",
});
ListData.Add(new EmployeeData
{
    Id = "10",
    Pid = "7",
    Name = "Andrew",
});
ListData.Add(new EmployeeData
{
    Id = "11",
    Name = "Robert King",
    HasChild = true,
});
ListData.Add(new EmployeeData
{
    Id = "12",
    Pid = "11",
    Name = "Richard",
});
ListData.Add(new EmployeeData
{
    Id = "13",
    Pid = "11",
    Name = "James",
});
ListData.Add(new EmployeeData
{
    Id = "14",
    Pid = "11",
    Name = "Murrey",
});
ListData.Add(new EmployeeData
{
    Id = "15",
    Pid = "11",
    Name = "Chris",
});
ListData.Add(new EmployeeData
{
    Id = "16",
    Name = "Margaret Peacock",
    HasChild = true,
});
ListData.Add(new EmployeeData
{
    Id = "17",
```

```
Pid = "16",
Name = "Ryaz",
});
ListData.Add(new EmployeeData
{
    Id = "18",
    Pid = "16",
    Name = "Mary",
});
ListData.Add(new EmployeeData
{
    Id = "19",
    Pid = "16",
    Name = "Stephen",
});
ListData.Add(new EmployeeData
{
    Id = "20",
    Pid = "16",
    Name = "Raffel",
});
}
```



Check/uncheck on clicking the tree node text in Blazor TreeView

The checkboxes of the tree view can be checked and unchecked by clicking the tree node using the `NodeClicked` event of TreeView.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<SfTreeView TValue="MusicAlbum" @ref="tree" ShowCheckBox="true"
AutoCheck="true" CheckedNodes="@CheckedNodes.ToArray()">
<TreeViewEvents TValue="MusicAlbum" OnKeyPress="TreeNodeClick"
NodeClicked="NodeClick" NodeChecking="BeforeCheck"
NodeExpanding="ExpandCollapse"
NodeCollapsing="ExpandCollapse"></TreeViewEvents>
<TreeViewFieldsSettings TValue="MusicAlbum" Id="Id" DataSource="@Albums"
Text="Name" ParentID="ParentId" HasChildren="HasChild" Expanded="Expanded"
IsChecked="IsChecked"></TreeViewFieldsSettings>
</SfTreeView>
@code{
SfTreeView<MusicAlbum> tree;
public bool ExpandIconClick { get; set; } = false;
```

```
public bool CheckBoxClick { get; set; } = false;
List<MusicAlbum> NodeDetails;
public List<string> CheckedNodes = new List<string>();
public class MusicAlbum
{
    public int Id { get; set; }
    public int? ParentId { get; set; }
    public string Name { get; set; }
    public bool Expanded { get; set; }
    public bool? IsChecked { get; set; }
    public bool HasChild { get; set; }
}
List<MusicAlbum> Albums = new List<MusicAlbum>();
public MusicAlbum ModelType = new MusicAlbum();
async void TreeNodeClick(NodeKeyPressEventArgs args)
{
    string Key = args.Event.Key;
    if (Key == "Enter" && !ExpandIconClick)
    {
        NodeDetails = this.tree.GetTreeData(args.NodeData.Id);
        bool check = NodeDetails[0].IsChecked ?? false;
        if (check)
        {
            CheckedNodes.Remove(args.NodeData.Id);
        }
        else
        {
            CheckedNodes.Add(args.NodeData.Id);
        }
    }
    ExpandIconClick = false;
}
async void NodeClick(NodeClickEventArgs args)
{
    if (!ExpandIconClick && !CheckBoxClick)
    {
        List<MusicAlbum> NodeDetails = this.tree.GetTreeData(args.NodeData.Id);
        var Element = new[] { args.Node };
        if (NodeDetails[0].IsChecked == true)
        {
            CheckedNodes.Remove(args.NodeData.Id);
        }
        else
        {
            CheckedNodes.Add(args.NodeData.Id);
        }
    }
    ExpandIconClick = false;
    CheckBoxClick = false;
}
void ExpandCollapse(NodeExpandEventArgs args)
{
    ExpandIconClick = true;
}
void BeforeCheck(NodeCheckEventArgs args)
{
    if (args.IsInteracted)
```

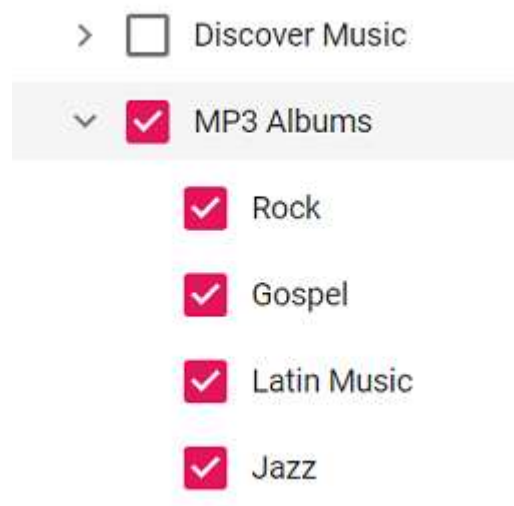


```
{
    CheckBoxClick = true;
}
protected override void OnInitialized()
{
    base.OnInitialized();
    Albums.Add(new MusicAlbum
    {
        Id = 1,
        Name = "Discover Music",
        HasChild = true,
    });
    Albums.Add(new MusicAlbum
    {
        Id = 2,
        ParentId = 1,
        Name = "Hot Singles"
    });
    Albums.Add(new MusicAlbum
    {
        Id = 3,
        ParentId = 1,
        Name = "Rising Artists"
    });
    Albums.Add(new MusicAlbum
    {
        Id = 4,
        ParentId = 1,
        Name = "Live Music"
    });
    Albums.Add(new MusicAlbum
    {
        Id = 14,
        HasChild = true,
        Name = "MP3 Albums",
        Expanded = true,
        IsChecked = true
    });
    Albums.Add(new MusicAlbum
    {
        Id = 15,
        ParentId = 14,
        Name = "Rock"
    });
    Albums.Add(new MusicAlbum
    {
        Id = 16,
        Name = "Gospel",
        ParentId = 14,
    });
    Albums.Add(new MusicAlbum
    {
        Id = 17,
        ParentId = 14,
        Name = "Latin Music"
    });
});
```

```

Albums.Add(new MusicAlbum
{
    Id = 18,
    ParentId = 14,
    Name = "Jazz"
});
}
}

```



Validate the text when renaming the tree node in Blazor TreeView

The tree node text could be validated while editing using `NodeEdited` event of the TreeView. Following is an example that shows how to validate and prevent empty values in tree node.

ASPX-CS

```

@using Syncfusion.Blazor.Navigations
<SfTreeView TValue="MusicAlbum" AllowEditing="true" @ref="tree">
<TreeViewEvents TValue="MusicAlbum" NodeEdited="AfterEdit"></TreeViewEvents>
<TreeViewFieldsSettings TValue="MusicAlbum" Id="Id" DataSource="@Albums"
Text="Name" ParentID="ParentId" HasChildren="HasChild" Expanded="Expanded"
IsChecked="IsChecked"></TreeViewFieldsSettings>
</SfTreeView>
@if (EditedStatus != null)
{
    @EditedStatus
}
@code{
public class MusicAlbum
{
    public int Id { get; set; }
    public int? ParentId { get; set; }
    public string Name { get; set; }
    public bool Expanded { get; set; }
    public bool? IsChecked { get; set; }
    public bool HasChild { get; set; }
}
SfTreeView<MusicAlbum> tree;

```

```
string EditedStatus = null;
List<MusicAlbum> Albums = new List<MusicAlbum>();
void AfterEdit(NodeEditEventArgs args)
{
    if (args.NewText.Trim() == "")
    {
        args.Cancel = true;
        EditedStatus = "TreeView item text should not be empty";
        StateHasChanged();
    }
    else if (args.NewText != args.OldText)
    {
        EditedStatus = "TreeView item text edited successfully";
        StateHasChanged();
    }
    else
    {
        EditedStatus = null;
    }
}
protected override void OnInitialized()
{
    base.OnInitialized();
    Albums.Add(new MusicAlbum
    {
        Id = 1,
        Name = "Discover Music",
        HasChild = true,
    });
    Albums.Add(new MusicAlbum
    {
        Id = 2,
        ParentId = 1,
        Name = "Hot Singles"
    });
    Albums.Add(new MusicAlbum
    {
        Id = 3,
        ParentId = 1,
        Name = "Rising Artists"
    });
    Albums.Add(new MusicAlbum
    {
        Id = 4,
        ParentId = 1,
        Name = "Live Music"
    });
    Albums.Add(new MusicAlbum
    {
        Id = 14,
        HasChild = true,
        Name = "MP3 Albums",
        Expanded = true,
        IsChecked = true
    });
    Albums.Add(new MusicAlbum
    {
```

```
Id = 15,  
ParentId = 14,  
Name = "Rock"  
});  
Albums.Add(new MusicAlbum  
{  
Id = 16,  
Name = "Gospel",  
ParentId = 14,  
});  
Albums.Add(new MusicAlbum  
{  
Id = 17,  
ParentId = 14,  
Name = "Latin Music"  
});  
Albums.Add(new MusicAlbum  
{  
Id = 18,  
ParentId = 14,  
Name = "Jazz"  
});  
}  
}
```

> Discover Music

▼ MP3 Albums

newNode

Gospel

Latin Music

Jazz

TreeView item text edited successfully

Customize treeview as accordion in Blazor TreeView Component

Accordion is an interface where a list of items can be collapsed or expanded, but only one list can be collapsed or expanded at a time. Customize the TreeView to make it behave as an accordion. Refer to the following code sample to create an accordion tree.

ASPX-CS

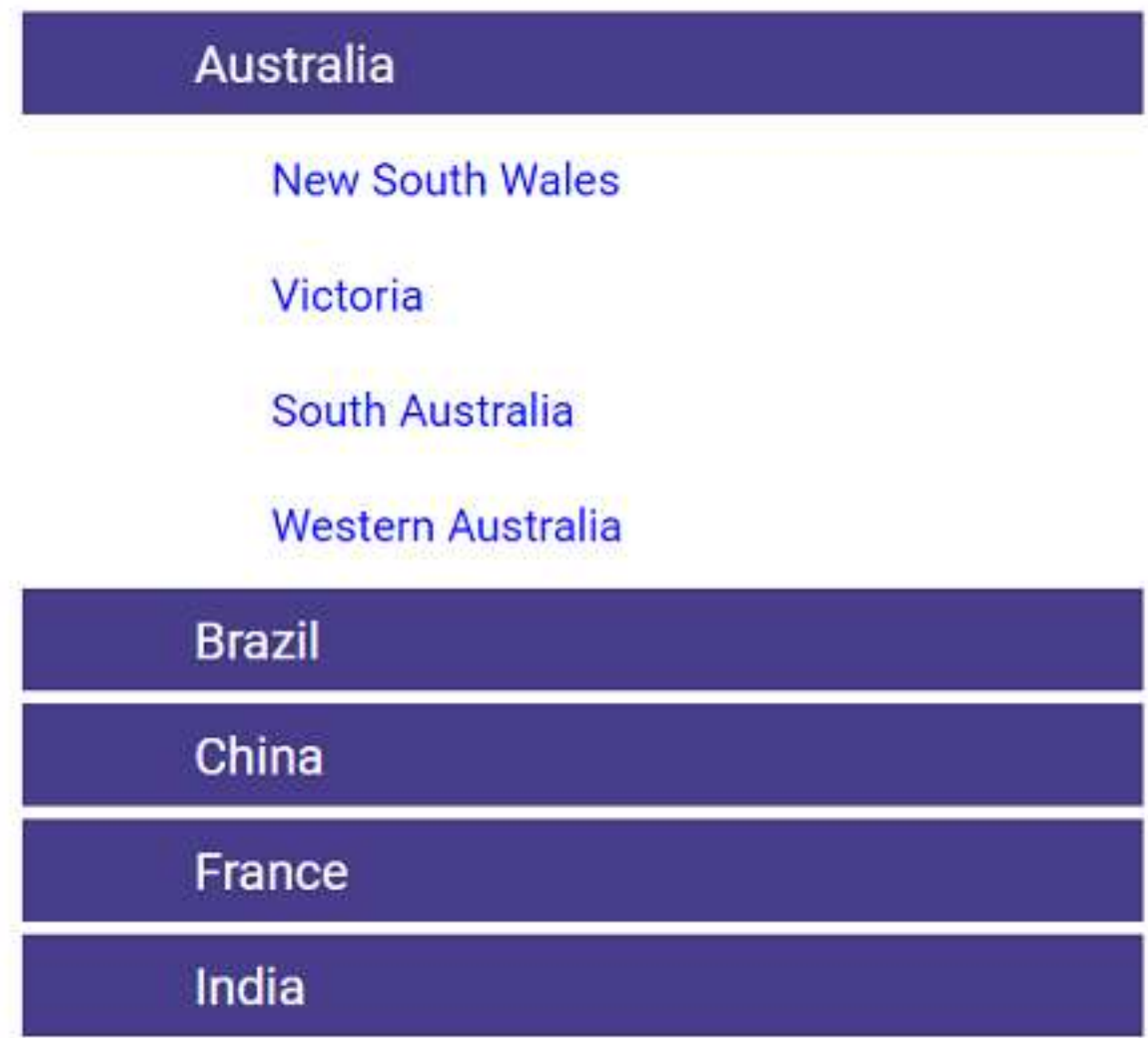
```
@using Syncfusion.Blazor.Navigations
<div class="tree-container">
<SfTreeView TValue="TreeItem" CssClass="accordiontree" @ref="tree"
ExpandOn="ExpandAction.Click">
<TreeViewEvents TValue="TreeItem"
NodeSelecting="BeforeSelect"></TreeViewEvents>
<TreeViewFieldsSettings DataSource="@TreeItems" Id="Id" Text="Name"
Child="Child"></TreeViewFieldsSettings>
</SfTreeView>
</div>
@code{
List<TreeItem> TreeItems = new List<TreeItem>();
SfTreeView<TreeItem> tree;
public List<string> ExpandedNodes = new List<string>();
async void BeforeSelect(NodeSelectEventArgs args)
{
this.tree.CollapseAll();
}
protected override void OnInitialized()
{
base.OnInitialized();
TreeItems.Add(new TreeItem
{
Id = 1,
Name = "Australia",
Child = new List<TreeItem>()
{
new TreeItem
{
Id = 2,
Name = "New South Wales",
},
new TreeItem
{
Id = 3,
Name = "Victoria"
},
new TreeItem
{
Id = 4,
Name = "South Australia"
},
new TreeItem
{
Id = 6,
Name = "Western Australia",
}
});
TreeItems.Add(new TreeItem
{
Id = 7,
Name = "Brazil",
Child = new List<TreeItem>()
{

```

```
new TreeItem
{
    Id = 8,
    Name = "Paraná"
},
new TreeItem
{
    Id = 9,
    Name = "Ceará"
},
new TreeItem
{
    Id = 10,
    Name = "Acre"
}
});
TreeItems.Add(new TreeItem
{
    Id = 11,
    Name = "China",
    Child = new List<TreeItem>()
    {
        new TreeItem
        {
            Id = 12,
            Name = "Guangzhou"
        },
        new TreeItem
        {
            Id = 13,
            Name = "Shanghai"
        },
        new TreeItem
        {
            Id = 14,
            Name = "Beijing"
        },
        new TreeItem
        {
            Id = 15,
            Name = "Shantou"
        }
    }
});
TreeItems.Add(new TreeItem
{
    Id = 16,
    Name = "France",
    Child = new List<TreeItem>()
    {
        new TreeItem
        {
            Id = 17,
            Name = "Pays de la Loire"
        },
        new TreeItem
```

```
{
    Id = 18,
    Name = "Aquitaine"
},
new TreeItem
{
    Id = 19,
    Name = "Brittany"
},
new TreeItem
{
    Id = 20,
    Name = "Lorraine"
}
});
TreeItems.Add(new TreeItem
{
    Id = 21,
    Name = "India",
    Child = new List<TreeItem>()
    {
        new TreeItem
        {
            Id = 22,
            Name = "Assam"
        },
        new TreeItem
        {
            Id = 23,
            Name = "Bihar"
        },
        new TreeItem
        {
            Id = 24,
            Name = "Tamil Nadu"
        }
    }
});
}
class TreeItem
{
    public int Id { get; set; }
    public string Name { get; set; }
    public bool HasChild { get; set; }
    public bool Expanded { get; set; }
    public List<TreeItem> Child;
}
}
<style>
/*To display border for the tree*/
.tree-container {
max-width: 350px;
max-height: 350px;
margin: auto;
overflow: auto;
}
```

```
/* To change the background color for the first level nodes*/
.accordiontree .e-list-item.e-level-1 > .e-fullrow, .accordiontree .e-list-
item.e-level-1.e-active > .e-fullrow, .accordiontree .e-list-item.e-level-
1.e-hover > .e-fullrow, .accordiontree .e-list-item.e-level-1 > .e-fullrow,
.accordiontree .e-list-item.e-level-1.e-active.e-hover > .e-fullrow {
background-color: darkslateblue;
border-color: darkslateblue;
}
/*To change the text color for the first level nodes*/
.accordiontree .e-list-item.e-level-1 > .e-text-content .e-list-text,
.accordiontree .e-list-item.e-level-1.e-active > .e-text-content .e-list-
text, .accordiontree .e-list-item.e-level-1.e-hover > .e-text-content .e-
list-text, .accordiontree .e-list-item.e-level-1.e-active.e-hover > .e-text-
content .e-list-text {
color: white;
font-size: 16px;
}
/*To hide the expand and collapse icon*/
.accordiontree .e-list-item.e-level- .e-icons.e-icon-collapsible,
.accordiontree .e-list-item.e-level-1 .e-icons.e-icon-collapsible,
.accordiontree .e-list-item.e-level-1 .e-icon-expandable {
display: none
}
/*To change the background color for the second level nodes*/
.accordiontree .e-list-item.e-level-2 > .e-fullrow, .accordiontree .e-list-
item.e-level-2.e-active > .e-fullrow, .accordiontree .e-list-item.e-level-
2.e-hover > .e-fullrow, .accordiontree .e-list-item.e-level-2 > .e-fullrow,
.accordiontree .e-list-item.e-level-2.e-active.e-hover > .e-fullrow {
background-color: white;
border-color: white;
}
/*To change the text color for the second level nodes*/
.accordiontree .e-list-item.e-level-2 > .e-text-content .e-list-text,
.accordiontree .e-list-item.e-level-2.e-active > .e-text-content .e-list-
text, .accordiontree .e-list-item.e-level-2.e-hover > .e-text-content .e-
list-text, .accordiontree .e-list-item.e-level-2.e-active.e-hover > .e-text-
content .e-list-text {
color: blue;
font-size: 14px;
}
</style>
```

Set tooltip for tree nodes in Blazor TreeView Component

TreeView control allows to set tooltip option to tree nodes using the **Tooltip** property. The following code example demonstrates how to set tooltip for TreeView nodes.

ASPX-CS

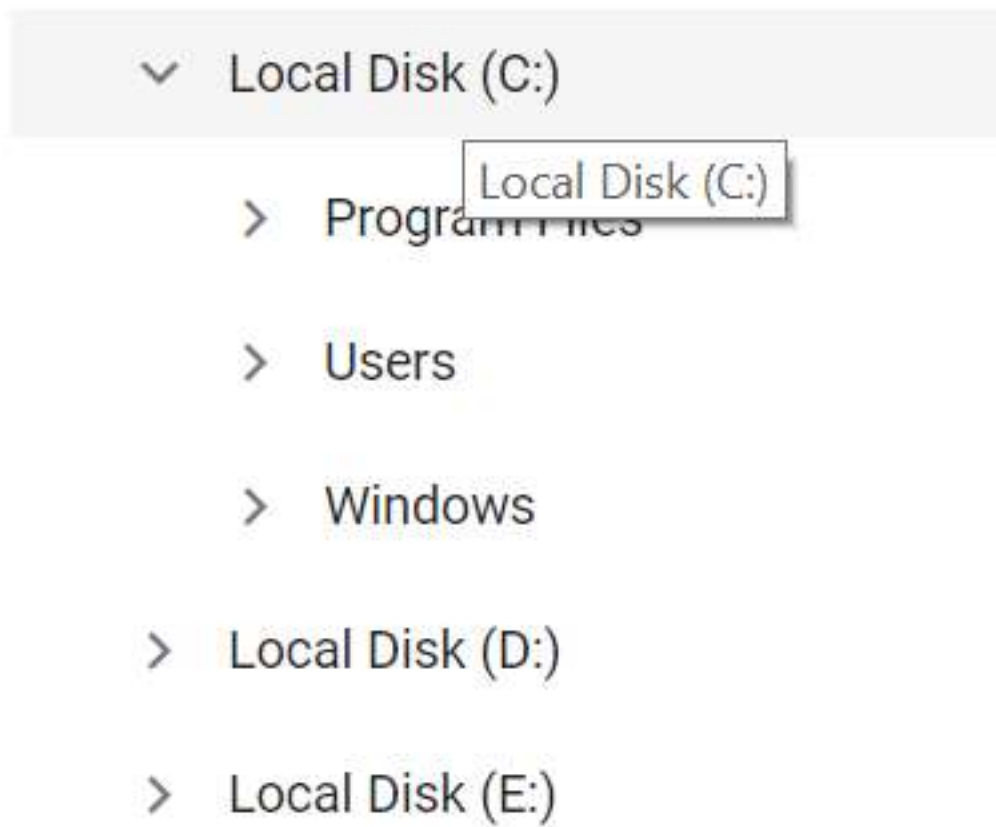
```
@using Syncfusion.Blazor.Navigations
<SfTreeView TValue="DriveData">
  <TreeViewFieldsSettings DataSource="@Drive" Id="NodeId" Text="NodeText"
  Tooltip="Tooltip" Expanded="Expanded"
  Child="@("Child") "></TreeViewFieldsSettings>
</SfTreeView>
@code{
List<DriveData> Drive = new List<DriveData>();
protected override void OnInitialized()
{
  base.OnInitialized();
  Drive.Add(new DriveData
  {
    NodeId = "01",
```

```

NodeText = "Local Disk (C:)",
Tooltip = "Local Disk (C:)",
Expanded = true,
Child = new List<DriveData>()
{
    new DriveData { NodeId = "01-01", NodeText = "Program Files", Tooltip =
        "Program Files",
        Child = new List<DriveData>()
        {
            new DriveData { NodeId = "01-01-01", NodeText = "Windows NT" , Tooltip =
                "Windows NT"},
        },
        new DriveData { NodeId = "01-02", NodeText = "Users", Tooltip="Users",
        Child = new List<DriveData>()
        {
            new DriveData { NodeId = "01-02-01", NodeText = "Smith", Tooltip= "Smith" },
            new DriveData { NodeId = "01-02-02", NodeText = "Public", Tooltip="Public"
            },
        },
        new DriveData { NodeId = "01-03", NodeText = "Windows", Tooltip= "Windows",
        Child = new List<DriveData>()
        {
            new DriveData { NodeId = "01-03-01", NodeText = "Boot", Tooltip = "Boot" },
        },
        },
    });
Drive.Add(new DriveData
{
    NodeId = "02",
    NodeText = "Local Disk (D:)",
    Tooltip = "Local Disk (D:)",
    Child = new List<DriveData>()
    {
        new DriveData { NodeId = "02-01", NodeText = "Personals",
        Tooltip="Personals"
        },
        new DriveData { NodeId = "02-02", NodeText = "Projects", Tooltip =
        "Projects"
        },
        new DriveData { NodeId = "02-02", NodeText = "Office", Tooltip = "Office"
        }
    }
});
Drive.Add(new DriveData
{
    NodeId = "03",
    NodeText = "Local Disk (E:)",
    Tooltip = "Local Disk (E:)",
    Child = new List<DriveData>()
    {
        new DriveData { NodeId = "03-01", NodeText = "Pictures", Tooltip =
        "Pictures"
        },
    },

```

```
new DriveData { NodeId = "03-02", NodeText = "Documents", Tooltip = "Documents"
},
new DriveData { NodeId = "03-03", NodeText = "Study Materials", Tooltip = "Study Materials"
},
},
});
}
class DriveData
{
public string NodeId { get; set; }
public string NodeText { get; set; }
public string Tooltip { get; set; }
public string Icon { get; set; }
public bool Expanded { get; set; }
public bool Selected { get; set; }
public List<DriveData> Child { get; set; }
}
}
```



Remove the checkbox of the parent node in Blazor TreeView Component

By enabling the **ShowCheckBox** property, check box could be rendered before each node of TreeView. However, some application needs to render check box in child nodes alone. In such case, remove the check box of the parent node by customizing the CSS.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
<SfTreeView TValue="Country" ShowCheckBox="true" CssClass="CustomTree">
<TreeViewFieldsSettings TValue="Country" Id="Id" DataSource="@Countries"
Text="Name" ParentID="ParentId" HasChildren="HasChild" Expanded="Expanded"
Selected="IsSelected"></TreeViewFieldsSettings>
</SfTreeView>
@code{
public class Country
{
public int Id { get; set; }
public int? ParentId { get; set; }
public string Name { get; set; }
public bool HasChild { get; set; }
public bool Expanded { get; set; }
public bool IsSelected { get; set; }
}
List<Country> Countries = new List<Country>();
protected override void OnInitialized()
{
base.OnInitialized();
Countries.Add(new Country
{
Id = 1,
Name = "Australia",
HasChild = true,
Expanded = true
});
Countries.Add(new Country
{
Id = 2,
ParentId = 1,
Name = "New South Wales",
IsSelected = true
});
Countries.Add(new Country
{
Id = 3,
ParentId = 1,
Name = "Victoria",
IsSelected = true
});
Countries.Add(new Country
{
Id = 4,
ParentId = 1,
Name = "South Australia"
});
Countries.Add(new Country
{
```

```
Id = 5,
ParentId = 1,
Name = "Western Australia"
});
Countries.Add(new Country
{
Id = 6,
Name = "Brazil",
HasChild = true
});
Countries.Add(new Country
{
Id = 7,
ParentId = 6,
Name = "Paraná"
});
Countries.Add(new Country
{
Id = 8,
ParentId = 6,
Name = "Ceará"
});
Countries.Add(new Country
{
Id = 9,
Name = "China",
HasChild = true
});
Countries.Add(new Country
{
Id = 10,
ParentId = 9,
Name = "Guangzhou"
});
Countries.Add(new Country
{
Id = 11,
ParentId = 9,
Name = "Shantou"
});
}
}
<style>
.CustomTree .e-list-item.e-level-1 .e-text-content.e-icon-wrapper
.e-checkbox-wrapper {
display: none
}
</style>
```

▼ Australia

☐ New South Wales

☐ Victoria

☐ South Australia

☐ Western Australia

> Brazil

> China

Get iconCss dynamically in Blazor TreeView Component

In the TreeView component, get the original bound data using the `GetTreeData` method. For this method, if the id of the tree node is passed, it returns the corresponding node information, or otherwise the overall tree nodes information will be returned. This method can be used to get the bound `IconCss` class in the `NodeChecking` event.

CSHARP

```
@using Syncfusion.Blazor.Navigations
<SfTreeView TValue="TreeItem"
SortOrder="@Syncfusion.Blazor.Navigations.SortOrder.Ascending"
ShowCheckBox="true" AutoCheck="false" @ref="tree">
<TreeViewEvents TValue="TreeItem"
NodeChecking="BeforeCheck"></TreeViewEvents>
<TreeViewFieldsSettings DataSource="@TreeDataSource" Id="NodeId"
Text="NodeText" Expanded="Expanded" Child="Child"
IconCss="Icon"></TreeViewFieldsSettings>
</SfTreeView>
@if (SelectedIcon != null)
{
    @SelectedIcon;
}
```

```

@code{
SfTreeView<TreeItem> tree;
List<TreeItem> TreeDataSource = new List<TreeItem>();
string SelectedIcon = null;
string SelectedID = null;
async void BeforeCheck(NodeCheckEventArgs args)
{
    if (args.Action == "check")
    {
        SelectedID = args.NodeData.Id;
        List<TreeItem> IconData = this.tree.GetTreeData(SelectedID);
        SelectedIcon = $"Icon class is: {IconData[0].Icon}";
        StateHasChanged();
    }
}

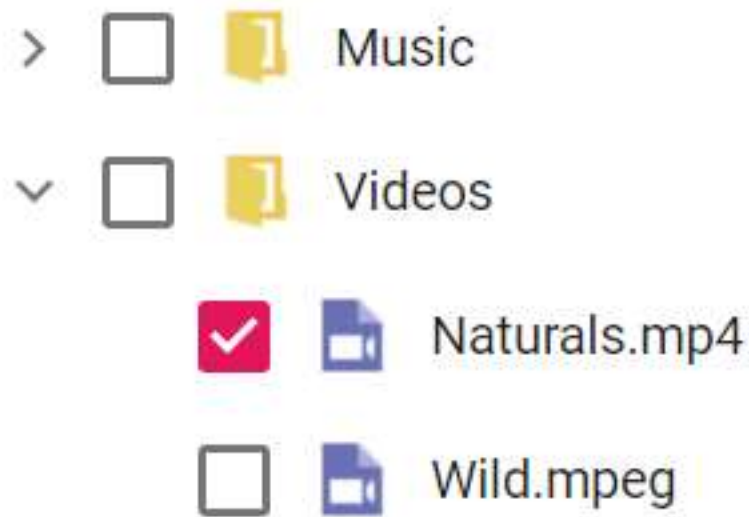
protected override void OnInitialized()
{
    base.OnInitialized();
    TreeDataSource.Add(new TreeItem
    {
        NodeId = "01",
        NodeText = "Music",
        Icon = "folder",
        Child = new List<TreeItem>()
        {
            new TreeItem { NodeId = "01-01", NodeText = "Gouttes.mp3", Icon = "audio" }
        },
    });
    TreeDataSource.Add(new TreeItem
    {
        NodeId = "02",
        NodeText = "Videos",
        Icon = "folder",
        Child = new List<TreeItem>()
        {
            new TreeItem { NodeId = "02-01", NodeText = "Naturals.mp4", Icon = "video" },
            new TreeItem { NodeId = "02-02", NodeText = "Wild.mpeg", Icon = "video" },
        },
    });
}

class TreeItem
{
    public string NodeId { get; set; }
    public string NodeText { get; set; }
    public string Icon { get; set; }
    public bool Expanded { get; set; }
    public bool Selected { get; set; }
    public List<TreeItem> Child;
}

<style>
.e-treeview .e-list-img {
width: 25px;
height: 25px;
}
/* Loading sprite image for TreeView */

```

```
.e-treeview .e-list-icon {
background-repeat: no-repeat;
background-image: url("css/treeview/images/file_Icons.png");
height: 20px;
}
/* Specify the Icon positions based upon class name */
.e-treeview .e-list-icon.folder {
background-position: -197px -552px
}
.e-treeview .e-list-icon.docx {
background-position: -197px -20px
}
.e-treeview .e-list-icon.ppt {
background-position: -197px -48px
}
.e-treeview .e-list-icon.pdf {
background-position: -197px -104px
}
.e-treeview .e-list-icon.images {
background-position: -197px -132px
}
.e-treeview .e-list-icon.zip {
background-position: -197px -188px
}
.e-treeview .e-list-icon.audio {
background-position: -197px -244px
}
.e-treeview .e-list-icon.video {
background-position: -197px -272px
}
.e-treeview .e-list-icon.exe {
background-position: -197px -412px
}
</style>
```

Icon class is: video

Get all child nodes through parentID in Blazor TreeView Component

This section shows how to get the child nodes from corresponding Parent ID. Using the `GetTreeData` method, the node details of the TreeView is achieved.

ASPX-CS

```
@using Syncfusion.Blazor.Navigations
@using Syncfusion.Blazor.Inputs
@using Newtonsoft.Json;
<SfTreeView TValue="TreeData" @ref="tree">
  <TreeViewFieldsSettings DataSource="@TreeDataSource" Id="Code" Text="Name"
    Selected="Selected" Expanded="Expanded"
    Child="Child"></TreeViewFieldsSettings>
</SfTreeView>
<SfMaskedTextBox @ref="mask" Placeholder="Enter the ID (ex: NA)"
  FloatLabelType="@FloatLabelType.Always" Width="250"></SfMaskedTextBox>
<button class="e-btn e-info" @onclick="@GetDetails">Submit</button>
<br />
@if (TreeNodeDetails != null)
{
  @TreeNodeDetails
}
@code{
  List<TreeData> TreeDataSource = new List<TreeData>();
  SfTreeView<TreeData> tree;
  SfMaskedTextBox mask;
  string TreeNodeDetails = null;
  async void GetDetails()
  {
    String EnteredText = mask.Value.ToString();
    List<TreeData> treeData = tree.GetTreeData(EnteredText);
    TreeNodeDetails = $"NodeData: {JsonConvert.SerializeObject(treeData[0])}";
    StateHasChanged();
  }
}
```

```
}
protected override void OnInitialized()
{
    base.OnInitialized();
    TreeDataSource.Add(new TreeData
    {
        Code = "NA",
        Name = "North America",
        Child = new List<TreeData>()
        {
            new TreeData { Code = "USA", Name = "United States of America", Selected =
            true },
            new TreeData { Code = "CUB", Name = "Cuba" },
            new TreeData { Code = "MEX", Name = "Mexico" },
        },
    });
    TreeDataSource.Add(new TreeData
    {
        Code = "AF",
        Name = "Africa",
        Child = new List<TreeData>()
        {
            new TreeData { Code = "NGA", Name = "Nygeria" },
            new TreeData { Code = "EGY", Name = "Egypt" },
            new TreeData { Code = "ZAF", Name = "South Africa" },
        },
    });
    TreeDataSource.Add(new TreeData
    {
        Code = "AS",
        Name = "Asia",
        Child = new List<TreeData>()
        {
            new TreeData { Code = "CHN", Name = "China" },
            new TreeData { Code = "IND", Name = "India" },
            new TreeData { Code = "JPN", Name = "Japan" },
        },
    });
    TreeDataSource.Add(new TreeData
    {
        Code = "EU",
        Name = "Europe",
        Child = new List<TreeData>()
        {
            new TreeData { Code = "DNK", Name = "Denmark" },
            new TreeData { Code = "AUT", Name = "Austria" },
            new TreeData { Code = "FIN", Name = "Finland" },
        },
    });
    TreeDataSource.Add(new TreeData
    {
        Code = "SA",
        Name = "South America",
        Child = new List<TreeData>()
        {
            new TreeData { Code = "BRA", Name = "Brazil" },
            new TreeData { Code = "COL", Name = "Colombia" },
        },
    });
}
```

```
new TreeData { Code = "ARG", Name = "Argentina" },
},
});
TreeDataSource.Add(new TreeData
{
    Code = "OC",
    Name = "Oceania",
    Child = new List<TreeData>()
    {
        new TreeData { Code = "AUS", Name = "Australia" },
        new TreeData { Code = "NZL", Name = "Newzealand" },
        new TreeData { Code = "WSM", Name = "Samoa" },
    },
});
TreeDataSource.Add(new TreeData
{
    Code = "AN",
    Name = "Antartica",
    Child = new List<TreeData>()
    {
        new TreeData { Code = "BVT", Name = "Bouvet Island" },
        new TreeData { Code = "ATF", Name = "French Southern Lands" },
    },
});
}
public class TreeData
{
    public string Code { get; set; }
    public string Name { get; set; }
    public bool Expanded { get; set; }
    public bool Selected { get; set; }
    public List<TreeData> Child;
}
}
```