FlexChart for WinForms

2022.04.19 更新

グレープシティ株式会社

FlexChart for WinForms

目次

FlexChart for WinFormsの概要

Charts are the perfect tools to visualize data regardless of domain you are working in. They are very effective in presenting huge data in condensed and easy to understand format, thus making it easy to put your point across without actually going through the huge data sets.

Modern looking, high performance **FlexChart for WinForms** brings you a powerful yet easy-to use API to configure charts. Starting from simple line and bar charts to specialized treemap and sunburst charts, FlexChart API provides a wide range with 80+ chart types. GDI+ and DirectX rendering, flexible data-binding, automatic rendering and adjustment of various chart elements such as data labels, and axis labels are just to name a few. With such wide variety of chart types and the rich feature set, you are just left with selection of a chart type that is appropriate for the data and the concept you want to present. Not just this, with FlexChart, you can empower the end-user to customize the chart at run-time using the built-in **FlexChart Designer**.

Documentation	**Blogs**
Create Your First Application with FlexChart	Get Started With FlexChart
FlexChart Designer	Evolving Best .NET Chart Control
End-user Interaction	Modern Data Visualization With FlexChart
Draw Gridlines on Chart Data	Three Ways To Use FlexChart
Videos	**Demo Samples**
Get Started With FlexChart	WinForms and WPF Chart Demo
Universal API Demo	
40+ .NET Chart Types	

Note: ComponentOne FlexChart for WinForms is compatible with both .NET 4.5.2 and .NET 5.

API References	
C1.Win.FlexChart.4.5.2 Assembly	C1.Win.FlexChart.5 Assembly

主な特長

Broad Spectrum of Chart Types

With 80+ chart types, FlexChart provides a wide range of charts to choose from. Starting from simple line chart and bar chart that come with several variations including stacked and 100% stacked options, the spectrum widens to the specialized charts such as treemap and sunburst.

GDI+ and DirectX Rendering

Render high quality charts with FlexChart as it provides support for DirectX and GDI+ rendering engine.

Flexible Data Binding

When it comes to binding, FlexChart provides enough flexibility as it gives you options to bind at the axis level, series level as well as chart level, to static arrays and dynamic lists, with single data source or multiple.

Simple yet Powerful API

With clear object model, FlexChart provides a simple and easy-to-use but powerful and efficient API that will not make your business applications slow.

Built-in Chart Designer

Provides great flexibility as you can edit various chart aspects such as appearance, binding, axes, legend, data labels etc. at run-time using the built-in FlexChart Designer.

Enhanced User Experience

FlexChart comes with many interactivity features such as selection, drilldown, scrolling, zooming, line markers, range selector and animation which enhance the overall user experience.

Pre-defined Color Palettes

With 16 pre-defined color palettes to choose from and an option to design custom palettes, FlexChart lets you create charts matching with UI of your business applications.

Serialization and De-serialization

Along with saving a chart as image, FlexChart lets you save an existing chart as an XML file or load it from the same to generate a new chart.

Automatic Adjustment of Chart Elements

FlexChart comes with built-in intelligence to automatically adjust the elements such as axis labels, data labels, legend as per the available real estate until or unless their position is specifically defined.

Rich Feature Set

Apart from the features mentioned above, FlexChart provides a rich feature set in order to provide better presentation and ease-of-use while analyzing charts. Multiple axes, axis grouping, legend grouping, multiple plot areas, annotations, custom content in data labels and line markers, options to manage overlapping of axis or data labels etc. are just to name a few.

機能比較

This section compares the features of various FlexCharts available across different platforms as well as shows a comparison with MS Chart.

Topic	Content
Comparing FlexCharts	Shows comparison between features of FlexChart for WinForms, WPF and UWP
Comparing WinForms Charts	Shows comparison between features of FlexChart for WinForms and MS Chart available for Windows platform.

FlexChartの比較

Apart from WinForms, ComponentOne provides FlexChart on all other popular platforms as well including WPF, UWP, ASP.NET MVC, JavaScript, and Xamarin. Although most of the features are common across all the platforms, this topic gives you a quick comparison between the FlexCharts available for desktop platforms, that is, WinForms, WPF and UWP. For easy navigation, we have broadly divided these features into different categories. Click on the corresponding category in the right-hand pane or scroll down to know more about availability of a particular feature.

Chart Types

Chart Types	Win	WPF	UWP
Line Charts			
Line	○	○	○
LineStacked	○	○	○
LineStacked100	○	○	○
Spline	○	○	○
SplineStacked	○	○	○
SplineStacked100	○	○	○
LineSymbols	○	○	○
StackedLineSymbols	○	○	○
StackedLineSymbols100	○	○	○
SplineSymbols	○	○	○
SplineSymbolsStacked	○	○	○
StackedSplineSymbols100	○	○	○
Step	○	○	○
StepSymbols	○	○	○
Area Charts			
Area	○	○	○
StackedArea	○	○	○
StackedArea100	○	○	○

SplineArea	○	○	○
StackedSplineArea	○	○	○
StepArea	○	○	○
StackedSplineArea100	○	○	○
Bar and Column Charts			
Bar	○	○	○
StackedBar	○	○	○
StackedBar100	○	○	○
Column	○	○	○
StackedColumn	○	○	○
StackedColumn100	○	○	○
Point Charts			
Bubble	○	○	○
Scatter	○	○	○
Pie Charts			
Pie	○	○	○
Doughnut	○	○	○
PieExploded	○	○	○
DoughnutExploded	○	○	○
Financial Charts			
CandleStick	○	○	○
Stock/ HighLowOpenClose	○	○	○
Heikin-Ashi	Available in Financial Chart of respective platforms: • Financial Chart for WinForms • Financial Chart for WPF • Financial Chart for UWP		
LineBreak/ThreeLineBreak			
Renko			
Kagi			
ColumnVolume			
EquiVolume			
CandleVolume			
ArmsCandleVolume			
Statistical Charts			
Box-and-Whisker	○	○	○
ErrorBar	○	○	○
Histogram	○	○	○

Cumulative Histogram	○	○	○
Frequency Polygon	○	○	○
Cumulative Frequency Polygon	○	○	○
RangedHistogram	○	○	○
Gaussian Curve	○	○	○
Pareto	○	○	○
Waterfall	○	○	○
Specialized Charts			
Floating Bar	○	○	○
Radar	○	○	○
Polar	○	○	○
Funnel	○	○	○
Heat Map	○	○	
Gantt	○	○	○
Sunburst	○	○	○
TreeMap	○	○	○
Trendlines	○	○	○
2D	○	○	○

Data Binding

Data Binding	Win	WPF	UWP
Objects implementing IEnumerable	○	○	○
Batch Updates	○	○	○

Look & Feel

Look & Feel	Win	WPF	UWP
Predefined Palettes	16	16	16
Custom Palette	○	○	○
Background Color	○	○	○
Background Image	○	○	○
Background Gradient/	○	○	○

HatchStyles			
Border and Border Styles	○	○	○
Render Modes	Native/DirectX	Native/Direct2D/Direct3D	Native/Direct3D

Chart Area

Chart Area	Win	WPF	UWP
Header	○	○	○
Footer	○	○	○
Header/Footer Borders	○	○	○
Header/Footer Alignment	○	○	○
Rotate ChartArea	○	○	○
Coordinate Conversion Methods	○	○	○

Plot Area

Plot Area	Win	WPF	UWP
Plot margins	○		
Markers for PlotElements	Supported for all symbol and point chart types	Supported for all symbol and point chart types	Supported for all symbol and point chart types
Markers Size	○	○	○
Markers: Border and Border styling	○	○	○
Background Image/ Gradient/ HatchStyle for PlotElements	○	○	○

Data Labels

DataLabels	Win	WPF	UWP
Offset	○	○	○
Connecting Lines	○	○	○
Borders and Border styling	○	○	○
Styling	○	○	○
Format String	○	○	○
Custom Content	○	○	○

Manage overlapping	○	○	○
Positions for Cartesian charts	Bottom/ Center/ Left/ None/ Right/ Top	Bottom/ Center/ Left/ None/ Right/ Top	Bottom/ Center/ Left/ None/ Right/ Top
Positions for Pie charts	Center/ Inside/ Outside/ None	Center/ Inside/ Outside/ None	Center/ Inside/ Outside/ None

Annotations

Annotations	Win	WPF	UWP
Pre-defined Shapes	○	○	○
Position	○	○	○
Attaching Annotations	○	○	○
Offset	○	○	○
Styling	○	○	○
Tooltip	○	○	○
Connector	○	○	○
Customization	○	○	○

Axis

Axis	Win	WPF	UWP
Axis: Primary X/Y	○	○	○
Axes: Secondary X/Y	○	○	○
Axes: Multiple Secondary X/Y	○	○	○
Axis Label: Format strings	○	○	○
Axis Label: Hide	○	○	○
Axis Label: styling	○	○	○
Axis Range (Min/Max) values	○	○	○
Axis: Hide	○	○	○
Axis: Logarithmic	○	○	○
Axis: Overlapping	○	○	○
Axis: Reverse	○	○	○

Axis Grouping	○	○	○
AxisLine Styling	○	○	○
Labels: Alignment	○	○	○
Labels: Angle	○	○	○
Labels: Overlapping	○	○	○
Major/ Minor GridLines	○	○	○
Major/ Minor TickMarks	○	○	○
Major/ Minor Units	○	○	○
Title and Title styling	○	○	○
Configure Origin	Any value	Any value	○
TickMarks Position	Cross/ Inside/ Outside/ None	Cross/ Inside/ Outside/ None	Cross/ Inside/ Outside/ None
Position	Top/ Bottom/ Left/ Right/ Auto/ None	Top/ Bottom/ Left/ Right/ Auto/ None	Top/ Bottom/ Left/ Right/ Auto/ None

Series

Series	Win	WPF	UWP
Multiple Series	○	○	○
Data binding	○	○	○
Chart types at series level	○	○	○
Styling	○	○	○
Visibility	Plot/ Legend/ Both Plot and Legend/ Hidden	Plot/ Legend/ Both Plot and Legend/ Hidden	Plot/ Legend/ Both Plot and Legend/ Hidden
Conditional Formatting	○	○	○
Stacked Groups	○	○	○
Handle Empty/ Null Data Points	○	○	○

Legends

Legends	Win	WPF	UWP
Title	○	○	○
Title Style	○	○	○

Toggle Series Visibility from legend	○	○	○
Orientation	Auto/ Vertical/ Horizontal	Auto/ Vertical/ Horizontal	Auto/ Vertical/ Horizontal
Position	Left/ Top/ Right/ Bottom	Left/ Top/ Right/ Bottom	Left/ Top/ Right/ Bottom
TextWrap	○	○	○
Custom Legend Icon	○	○	○

Marker Symbols

Marker Symbols	Win	WPF	UWP
Box	○	○	○
Dot	○	○	○

User Interactions

User Interactions	Win	WPF	UWP
Animation	○	○	○
Tooltips	○	○	○
Series selection	○	○	○
Point selection	○	○	○
LineMarkers aka Crosshairs	○	○	○
Range Selector	○	○	○
Zooming	○	○	○
Scrolling	○	○	○
Hit Test	○	○	○
Sunburst Drilldown	○	○	○
TreeMap Drilldown	○	○	○

Tooltips

Tooltips	Win	WPF	UWP
Auto tooltips	○	○	○
Custom content	○	○	○
Show Delay	○	○	○
Styling	○	○	○

Pie Charts

Pie Charts	Win	WPF	UWP
Exploded slices	○	○	○
Inner Radius	○	○	○
Starting Angle of first slice	○	○	○

Exporting/Importing & Printing

Exporting/Importing & Printing	Win	WPF	UWP
Export to JPEG/ JPG	○	○	○
Export to PNG	○	○	○
Export to SVG	○		
Export to BMP		○	○
Serialization Support	○	○	○
Printing support	○	○	○

Footprint

Footprint	Win	WPF	UWP
Assembly Size	229KB	183KB	218KB

MSChart の比較

This topic gives you a quick comparison between the FlexChart for WinForms and the standard MS Chart available for WinForms platform. For easy navigation, we have broadly divided these features into different categories. Click on the corresponding category in the right-hand pane or scroll down to know more about availability of a particular feature.

Chart Types

Chart Types	FlexChart	MS Chart
Area	○	○
StackedArea	○	○
StackedArea100	○	○
SplineArea	○	○
StackedSplineArea	○	
StackedSplineArea100	○	
StepArea	○	
Bar	○	○
StackedBar	○	○
StackedBar100	○	○

Bubble	○	○
CandleStick	○	○
Column	○	○
Combination	○	○
StackedColumn	○	○
StackedColumn100	○	○
Histogram	○	○
Ranged Histogram	○	○
Pareto	○	
Gantt	○	
Floating Bar	○	
Stock/ HighLowOpenClose	○	○
Line	○	○
LineStacked	○	
LineStacked100	○	
Spline	○	○
SplineStacked	○	
SplineStacked100	○	
LineSymbols	○	
StackedLineSymbols	○	
StackedLineSymbols100	○	
SplineSymbols	○	
SplineSymbolsStacked	○	
StackedSplineSymbols100	○	
Step	○	
StepSymbols	○	
Pie	○	○
Doughnut	○	○
PieExploded	○	○
DoughnutExploded	○	○
Point/ Scatter	○	○
Radar	○	○
Polar	○	○
Box-and-Whisker	○	○
ErrorBar	○	○

Funnel	○	○
Sunburst	○	
TreeMap	○	
Trendlines	○	
Waterfall	○	
2D	○	○
Heikin-Ashi	Available in Financial Chart for WinForms	
LineBreak/ThreeLineBreak		○
Renko		○
Kagi		○
ColumnVolume		
EquiVolume		
CandleVolume		
ArmsCandleVolume		
FastLine		○
FastPoint		○
StepLine		○
RangeBar		○
RangeColumn		○
SplineRange		○
Range		○
PointAndFigure		○
Pyramid		○
3D		○

Data Binding

Data Binding	FlexChart	MS Chart
MS SQL Server	○	○
OLEDB	○	○
ODBC	○	○
Object Data Source	○	○
Oracle	○	○
SharePoint Objects	○	○
Unbound Data		○

Chart Features

Chart Features	FlexChart	MS Chart
Annotations	○	○
Axis binding	○	○
Customizable data labels	○	○
Customizable headers and footers	○	○

Data Manipulations

Data Manipulations	FlexChart	MS Chart
Aggregation	With custom code	○
Sorting	With custom code	○
TopN	With custom code	○

Look & Feel

Look & Feel	FlexChart	MS Chart
Predefined Palettes	16	12
Custom Palette	○	○
Background Color	○	○
Background Image	○	○
Background Gradient/ HatchStyles		○
Border and Border Styles		○
Themes	16 themes	12 themes
DirectX Rendering, GDI+, SVG Rendering	○	

Chart Area

Chart Area	FlexChart	MS Chart
Header	○	○
Footer	○	
Multiple Headers		○
Header/ Footer Borders	○	○
Header/ Footer Alignment	○	○
Rotate ChartArea	○	
Multiple Chart or Plot Areas	○	○
Toggle Visibility		○
Coordinate Conversion Methods	○	○

Plot Area

Plot Area	FlexChart	MS Chart
Plot position configuration		○
Plot margins	○	
Markers for PlotElements	Supported for all symbol and point chart types	○
Markers Size	○	○
Markers as Images		○
Markers: Border and Border styling		○
Background Image/ Gradient/ HatchStyle for PlotElements		○
Border and Border styling for Plot Elements		○
Empty Points Styling		○

Data Labels

Data Labels	FlexChart	MS Chart
Offset	○	
ConnectingLines	○	○
Borders and Border styling	○	○
Styling	○	○
Format String	○	○
Custom Content	○	
Manage Overlapping	○	
Positions for Cartesian charts	Bottom/ Center/ Left/ None/ Right/ Top	Only applies to bar charts. Outside/ Left/ Right/ Center
Positions for Pie charts	Center/ Inside/ Outside/ None	Inside/ Outside/ None

Axis

Axis	FlexChart	MS Chart
Axis: Primary X/Y	○	○
Axis: Secondary X/Y	○	○
Axis: Multiple Secondary X/Y	○	
Axis Label: Format strings	○	
Axis Label: Hide	○	○
Axis Label: styling	○	○
Axis Range (Min/Max) values	○	
Axis: Hide	○	○

Axis: Logarithmic	○	○
Axis: Reverse	○	○
Axis: Grouping	○	○
AxisLine Styling	○	○
Labels: Alignment	○	
Labels: Angle	○	○
Labels: Hide overlapping	○	
Major/ Minor GridLines	○	○
Major/ Minor TickMarks	○	○
Major/ Minor Units	○	○
TickMarks Length	○	
TickMarks Styling	○	○
Title and Title Styling	○	○
Configure Origin	Any value	Auto/ Min/ Max
TickMarks Position	Cross/ Inside/ Outside/ None	Cross/ Inside/ Outside/ None
Position	Top/ Bottom/ Left/ Right/ Auto/ None	
ArrowHead styling		○
Axis Label: Custom		○
Axis Label: Staggered		○
ScaleBreaks		○
Multiple Axis	○	

Series

Series	FlexChart	MS Chart
Multiple Series	○	○
Data binding	○	○
Chart types at series level	○	○
Styling	○	○
Visibility	Plot/ Legend/ Both Plot and Legend/ Hidden	Plot/ Plot and Legend/ Hidden
Conditional Formatting	○	
Stacked charts	○	
Copy/ Merge/ Split Series Data		○
Handle Empty/ Null Data Points	○	○

Legends

Legends	FlexChart	MS Chart

Title	○	○
Title Style	○	○
TextWrap	○	
Toggle Series Visibility from legend	○	
Orientation	Auto/ Vertical/ Horizontal	Vertical/ Horizontal/ Table
Position	Left/ Top/ Right/ Bottom	Left/ Top/ Right/ Bottom
Custom Legend Items		○
Items Ordering		○
Multiple Legends		○
Custom Legend Icon	○	

Marker Symbols

Marker Symbols	FlexChart	MS Chart
Box	○	○
Dot	○	○
Diamond		○
Triangle		○
Cross		○
Star4		○
Star5		○
Star6		○
Star10		○

User Interactions

User Interactions	FlexChart	MS Chart
Tooltips	○	○
Series selection	○	○
Point selection	○	○
Range Selection	○	
Zooming	○	○
Scrolling	○	○
Drilldown	With custom code	With custom code
Sunburst Drilldown	○	
TreeMap Drilldown	○	
Draggable Annotations		○

Statistical Analysis using predefined formulas		○
Financial Analysis using predefined Indicators and Overlays	○	○
DataPoint Filtering		○
DataPoint Sorting		○
Grouping		○
LineMarkers	○	○
DataPoints Searching		○
HitTest	○	○

Tooltips

Tooltips	FlexChart	MS Chart
Auto tooltips	○	
Custom content	○	String type only
Show Delay	○	
Styling	○	
String type only		○
Tooltips for different chart elements		○

Financial Chart Features

Financial Chart Features	Financial Chart	MS Chart
Bollinger bands overlays	Available in Financial Chart for WinForms	
Envelope overlay		
Fibonacci tools		
Indicators		
MACD Indicator		
Moving averages		
Overlays		
Stochastic Oscillator Indicator		
Fibonacci tools		

Pie Charts

Pie Charts	FlexChart	MS Chart
Exploded slices	○	○
Inner Radius	○	○
Starting Angle of first slice	○	○

Exporting, Importing & Printing

Exporting, Importing & Printing	FlexChart	MS Chart
Export to JPEG/ JPG	○	○
Export to PNG	○	○
Export to SVG	○	
Export to BMP		○
Export to EMF		○
Export to EMfDual		○
Export to EMfPlus		○
Export to GIF		○
Export to TIFF		○
Save/ Load to Binary files		○
Save/ Load to memory streams		○
Save/ Load to XML		○
Export to Dataset		○
Printing	○	○
Print Support	○	○

Footprint

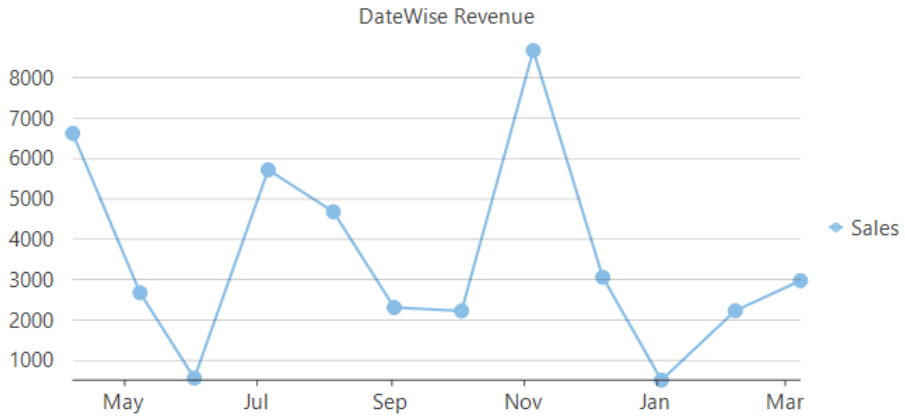
Footprint	FlexChart	MS Chart
Assembly Size	330KB	904KB

クイックスタート

The following quick-start guide will take you through the basics of FlexChart control.

.NET 4.5.2

This quick start will guide you through the steps of creating a simple line symbol chart using the FlexChart control. Follow the steps below to get started:



Set up the application

1. Create a new Windows Forms app.
2. Drag and drop the FlexChart control from the toolbox onto the form.
   Observe: A column type chart is drawn with a default data.

Bind the FlexChart control to a data source

1. Create a data source.

```
```

2. Bind the FlexChart to this data source by setting the DataSource property.

```
```

Configure the FlexChart control

1. Clear the default series getting displayed in the chart.
2. Add a new series using Add method and configure the X and Y axes by setting the BindingX and Binding property.
3. Configure the chart by setting the ChartType and other required properties.

```
```

.NET 5

To create a simple WinForms application in .NET 5 Core for FlexChart control, complete the following steps:

1. Create a new **Windows Forms** application.
2. Switch to code editor. Initialize and add the FlexChart control to the form.

```
C#                                                                    copyCode
// コントロールを初期化します
FlexChart flexChart = new FlexChart();
// フォームにコントロールを追加します
this.Controls.Add(flexChart);
```

3. Create a data source:

```
C#                                                                    copyCode
// データソースを作成します
public List<Product> GetProductRevenue()
{
    List<Product> _list = new List<Product>();
    _list.Add(new Product() { Name = "Desktop", Date = new DateTime(2018, 04, 07), Orders = 265, Revenue = 6625 });
    _list.Add(new Product() { Name = "Desktop", Date = new DateTime(2018, 05, 08), Orders = 107, Revenue = 2675 });
    _list.Add(new Product() { Name = "Mouse", Date = new DateTime(2018, 06, 02), Orders = 56, Revenue = 560 });
    _list.Add(new Product() { Name = "Mouse", Date = new DateTime(2018, 07, 06), Orders = 572, Revenue = 5720 });
    _list.Add(new Product() { Name = "Mouse", Date = new DateTime(2018, 08, 05), Orders = 468, Revenue = 4680 });
    _list.Add(new Product() { Name = "Printer", Date = new DateTime(2018, 09, 02), Orders = 154, Revenue = 2310 });
    _list.Add(new Product() { Name = "Desktop", Date = new DateTime(2018, 10, 03), Orders = 89, Revenue = 2225 });
    _list.Add(new Product() { Name = "Desktop", Date = new DateTime(2018, 11, 05), Orders = 347, Revenue = 8675 });
    _list.Add(new Product() { Name = "Printer", Date = new DateTime(2018, 12, 07), Orders = 204, Revenue = 3060 });
    _list.Add(new Product() { Name = "Printer", Date = new DateTime(2019, 01, 03), Orders = 34, Revenue = 510 });
    _list.Add(new Product() { Name = "Mouse", Date = new DateTime(2019, 02, 06), Orders = 223, Revenue = 2230 });
    _list.Add(new Product() { Name = "Desktop", Date = new DateTime(2019, 03, 08), Orders = 119, Revenue = 2975 });
    return _list;
}
```

4. Bind the FlexChart control with the data source.

```
C#                                                                    copyCode
// データをFlexChartに渡します
flexChart.DataSource = GetProductRevenue();
```

5. Configure the control and add a chart type. Set other properties of the control.

```
C#                                                                    copyCode
// FlexChartを構成します
flexChart.Series.Clear();
// チャートタイプを設定します
flexChart.ChartType = C1.Chart.ChartType.LineSymbols;
// チャートのヘッダーを設定します
flexChart.Header.Content = "DateWise Revenue";
```

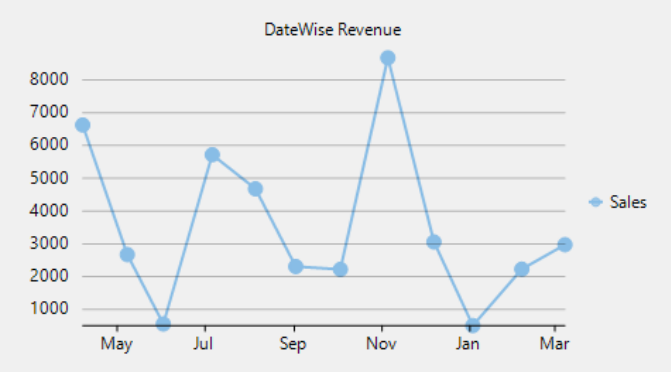
6. Add a Series to the chart and bind it (AxisY) to the 'Revenue' field of DataCollection.

```
C#                                                                    copyCode
// チャートに系列を追加し、DataCollectionの「Revenue」フィールドにバインドします
flexChart.Series.Add(new C1.Win.Chart.Series
{
    // Nameプロパティを使用して、凡例の系列に対応して表示される文字列を指定します
    Name = "Sales",
    Binding = "Revenue"
});
```

7. Bind the chart's AxisX to 'Date'.

```
C#                                                                    copyCode
// チャートのX軸を「Date」にバインドして、日付が横軸に表示されるようにします
flexChart.BindingX = "Date";
```

Run the code and observe the output:



Note: WinForms .NET 5 Edition does not include rich design-time support yet. We will enhance it in future releases.

設計時サポート

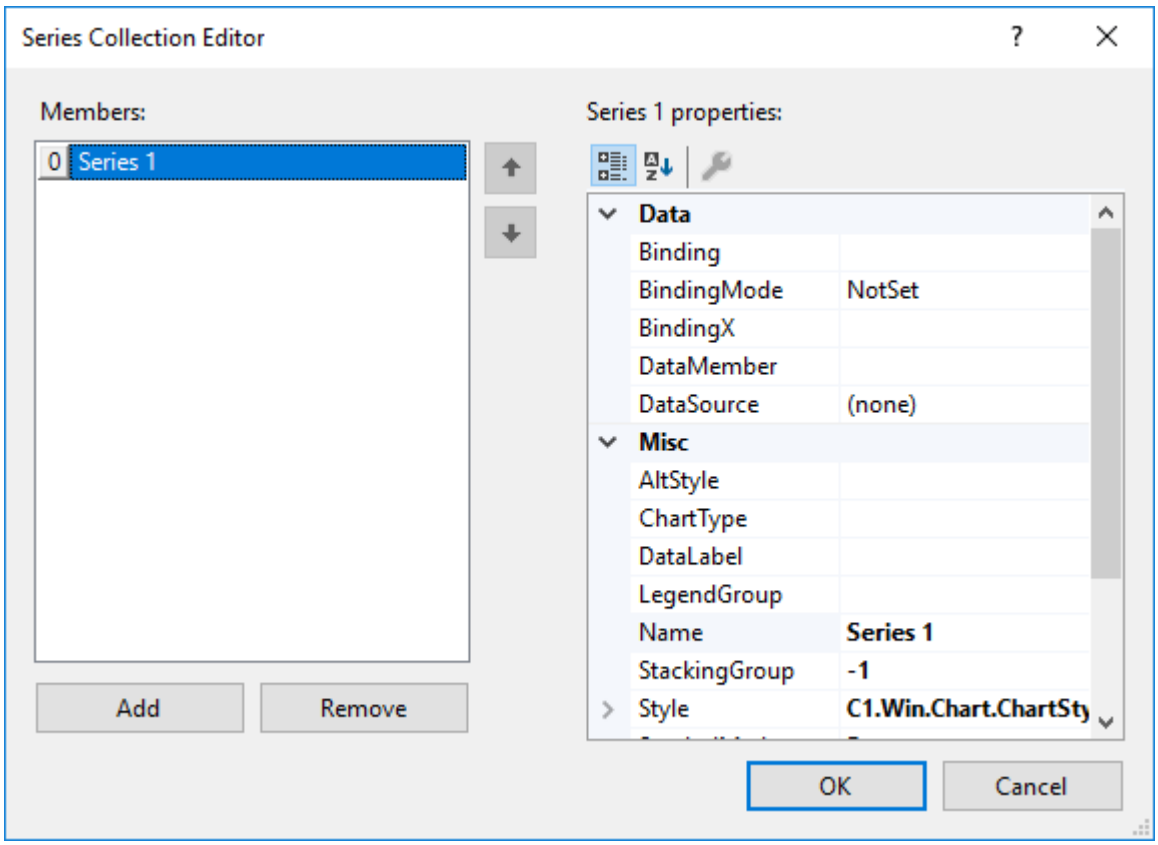
FlexChart lets you configure the charts easily at design-time using the property grid, collection editors etc. in Visual Studio. There are a lot of options available through property grid such as configuring chart axes, setting chart type, configuring data source, applying animation, customizing look and feel etc. Apart from the property grid, FlexChart also provides following collection editors to make your coding task easier:

- **Series Collection Editor**
- **PlotArea Collection Editor**

Series Collection Editor

Series Collection Editor of FlexChart can be used to add the series and set the related properties at design-time. You can set the data source, chart type, tooltip, visibility etc. of each series using this collection editor. Following are the steps to access the same:

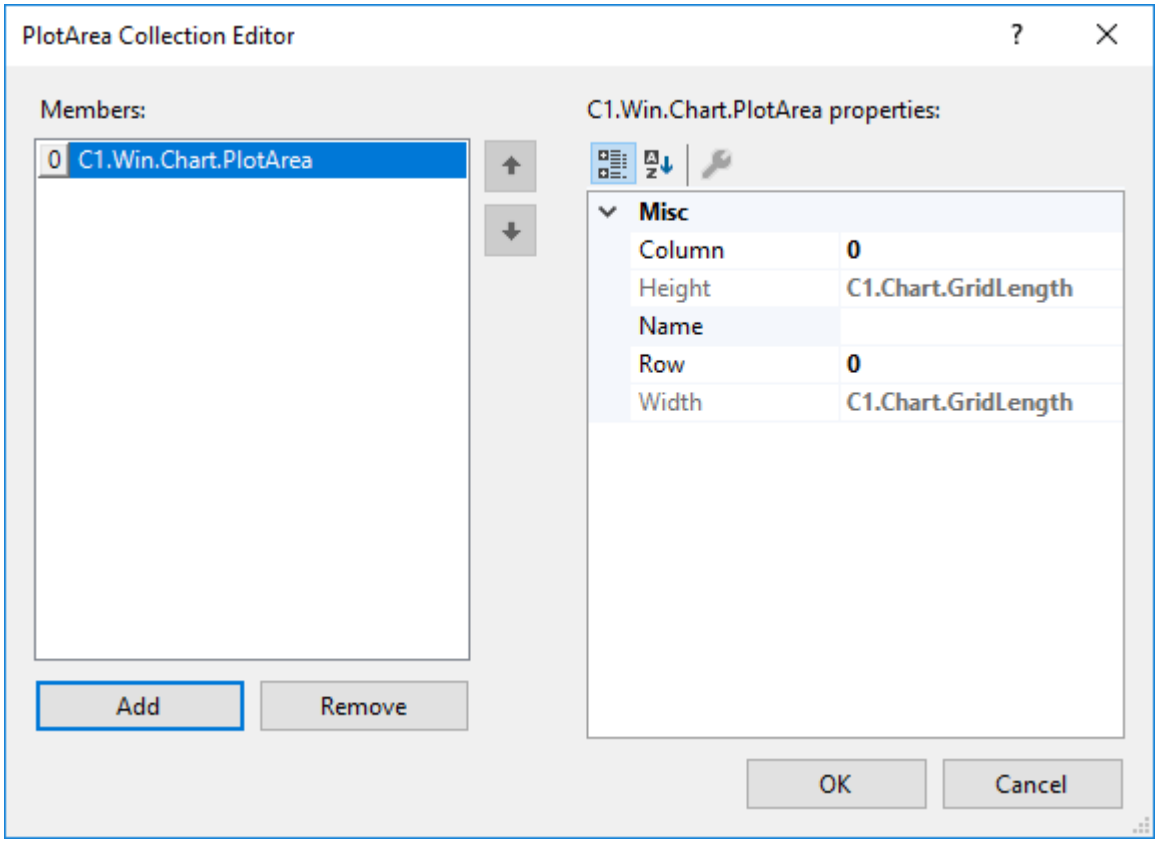
1. Right-click the FlexChart control on form.
2. Select **Properties** from the context menu.
3. In the **Properties** window, click the ellipsis button next to the **Series** property.



PlotArea Collection Editor

PlotArea Collection Editor of FlexChart can be used to add the plot area and set the related properties at design-time. You can set the height, width, row, column etc. of each plot area using this collection editor. Following are the steps to access the same:

1. Right-click the FlexChart control on form.
2. Select **Properties** from the context menu.
3. In the **Properties** window, click the ellipsis button next to the **PlotAreas** property.

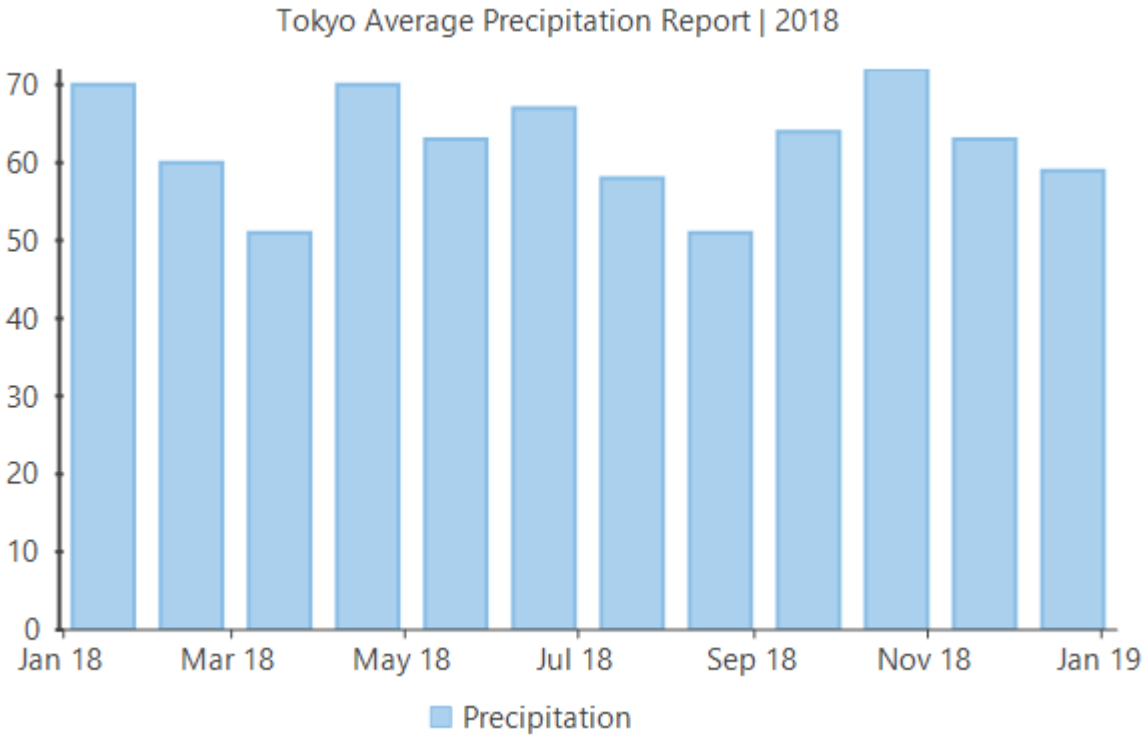


結合

Binding provides a way to create a link between FlexChart and the application data . FlexChart allows you to bind data coming from databases as well as data stored in other structures, such as arrays and collections. This topic discusses three ways to provide data to a chart.

Chart Binding

Chart binding is the simplest form of binding in which you just have to set a data source by setting the DataSource property and then map the Binding and BindingX properties to fields that you want to plot on Y-axis and X-axis of the chart respectively.



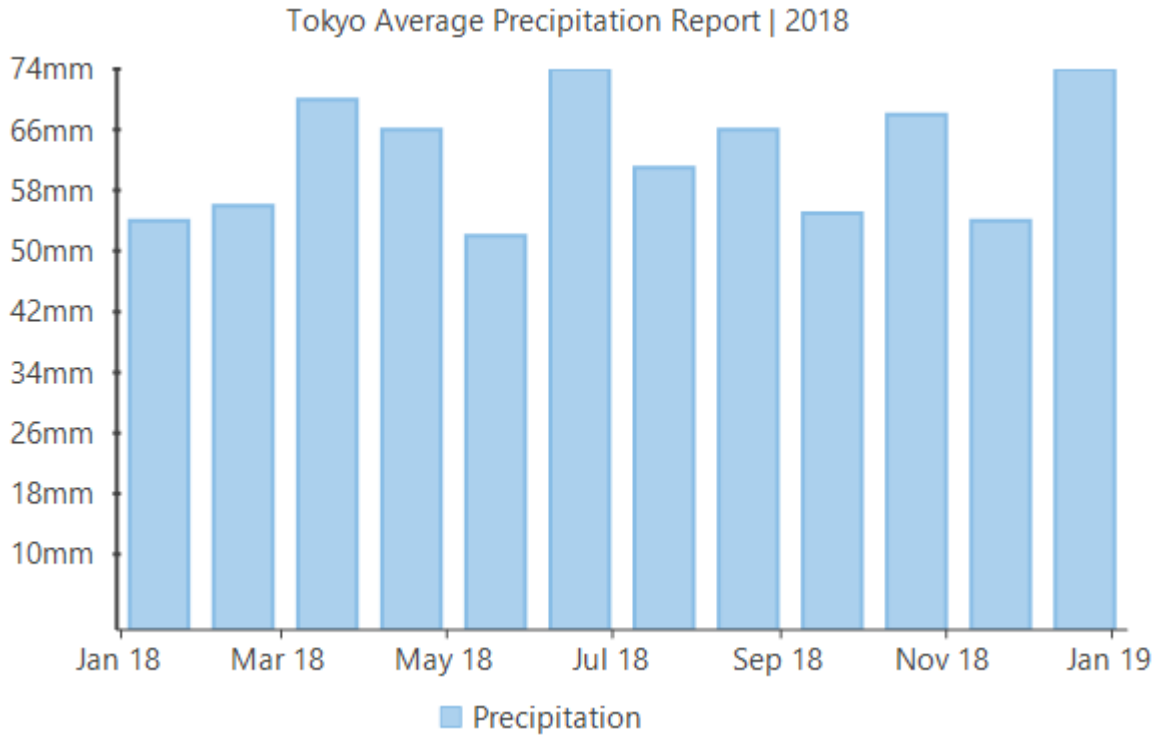
CS VB

Note that the above sample code uses a custom method namedGetTemperatureData tosupply data. You can set up the data source as per your requirements.

CS VB

Axis Binding

Axis binding refers to binding an axis with a separate data source to render axis labels different from what gets displayed on binding the chart with the chart data source. For instance, in the example above, we canoverride the Y-axis labels that are rendered automatically on binding with chart data source to display the precipitation values in mm. FlexChart providesDataSource andBinding properties so that you can bind your axes to another data source.



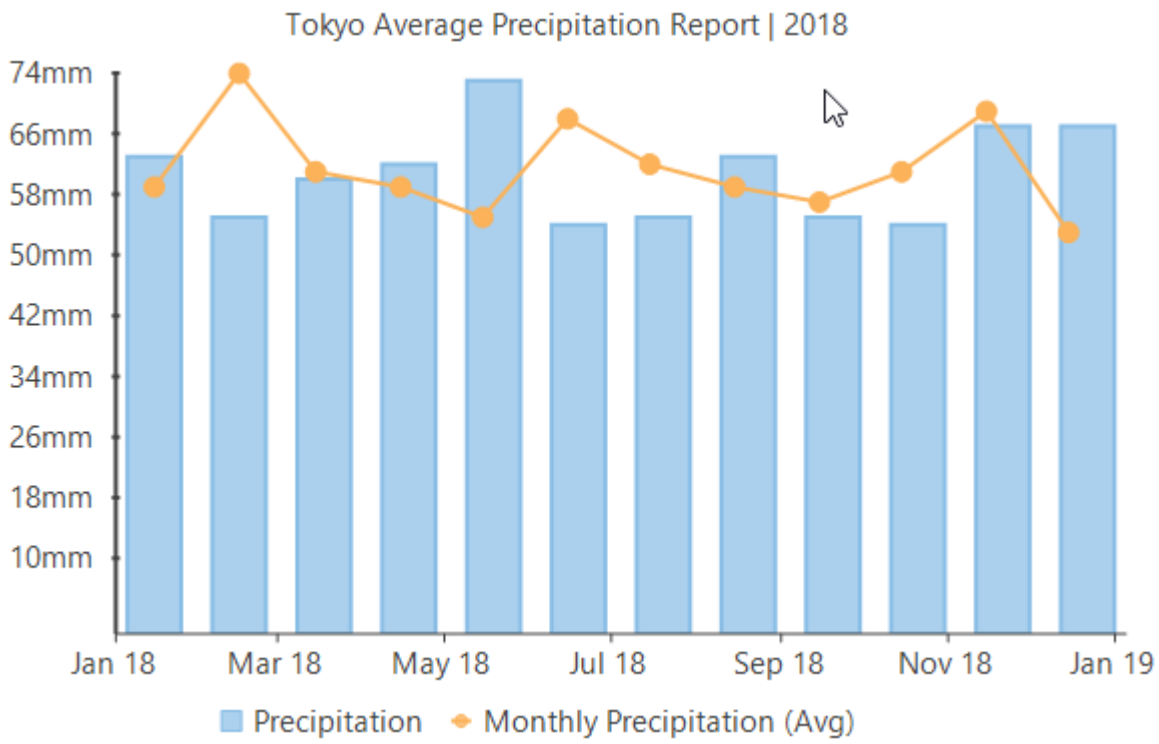
CS VB

Note that the above sample code uses a custom method namedGetAxisBindinglabels tosupply data. You can set up the data source as per your requirements.

CS VB

Series Binding

In series binding, you can bind a particular series to a separate data source and display the data which is not there in the originaldata source. For instance, in the example from above two sections, we can further add another series to show the average monthly precipitation in the same chartfrom a different data source.

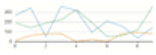
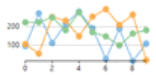
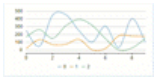
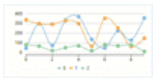
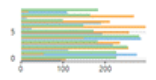
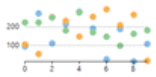
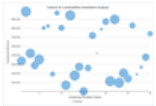
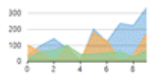
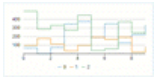


CS VB

チャートタイプ

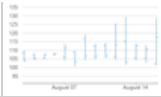
FlexChart supports a wide range of chart types including various variations. For instance, along with line charts, you can also create stacked line chart, stacked 100 line chart etc. using properties available in the FlexChart API. This topic gives a quick snapshot of all the major chart types along with their use cases. The thumbnails below let you have a glance of how a particular chart type looks like. To know about variations of these chart types, you can navigate to corresponding topics by clicking on the hyperlinks.

Basic Charts

Chart Type	Chart Snapshot	Use Case
Line Chart		Use line charts to plot the continuously changing data against a periodic interval. They are generally used to show trends over a period of time. However, they can also be used to plot data against other continuous periodic values such as temperature, distance etc.
Line Symbol Chart		Use line symbol charts when you want to indicate the exact data points plotted on a line chart.
Spline Chart		Like line charts, use the spline charts to demonstrate the data changing against a periodic interval. However, unlike line chart, they show the gradual change in trend. They are also aesthetically better than a line chart.
Spline Symbol Chart		Use spline symbol charts when you want to indicate the exact data points plotted on a spline chart.
Bar Chart		Use bar charts for demonstrating patterns and trends across different categories. In these charts, each horizontal bar corresponds to a category and its length corresponds to the value or measure of that category.
Column Chart		Very similar in usage with bar charts, column charts use vertical columns instead of horizontal bars for identifying the trends. These charts are generally used in the case of fewer categories that can be plotted easily on X-axis.
Scatter Chart		Use the scatter chart to graph the pairs of numerical data and hence to identify the relationship between the two variables.
Bubble Chart		Use the bubble charts to plot three dimensional data. Bubble charts are often used to present financial data.
Area Chart		Use area charts to emphasize the magnitude of trend along with demonstrating the trend. Based on line charts, area charts are distinct because of the filled area between the line segments and the x-axis.
Step Chart		Use step charts to plot the data that shows sudden changes on irregular intervals and remains constant between these intervals.
Combination Chart		Use the combination of two or more different charts in same plot area to compare the different data sets that are related to each other.
Pie Chart		Use pie charts to demonstrate the relative contribution of various categories.
Donut/Doughnut Chart		Similar to pie charts, use donut charts to display proportional data with just a visual difference of a hole in the center.

Financial Charts

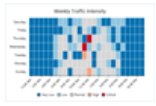
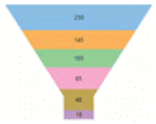
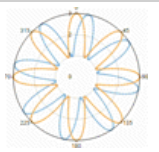
Chart Type	Chart Snapshot	Use Case
Candlestick		Use candlestick charts to plot high, low, open and close values of stock over a period of time. In these charts, area between open and close values forms body of a candle and, vertical lines between high and low values form the wick and tail respectively.

HighLowOpenClose (HLOC)		Similar to candlestick charts, use HLOC charts to show the stock's high, low, open and close values. The only difference between the two is that there is no candlestick body in this case.

Statistical Charts

Chart Type	Chart Snapshot	Use Case
Box-and-Whisker		Use box-and-whisker charts to display the distribution of numerical data in the form of quartiles, means and outliers.
Error Bar		Use error bar charts to indicate the estimated error in the measured data through standard deviation of the data set.
Histogram		Use histograms to present the data distribution over a continuous interval to know which sub-interval gets the maximum or minimum frequency.
Ranged Histogram		Use Excel-like histogram to visualize the frequency distribution against ranged X-axis in category or non-category mode.
Pareto Chart		Use pareto chart to analyze the frequency of problems or causes in a process.
Waterfall Chart		Use waterfall charts to indicate the fluctuation in data over a period of time and its cumulative impact on an initial value.

Specialized Charts

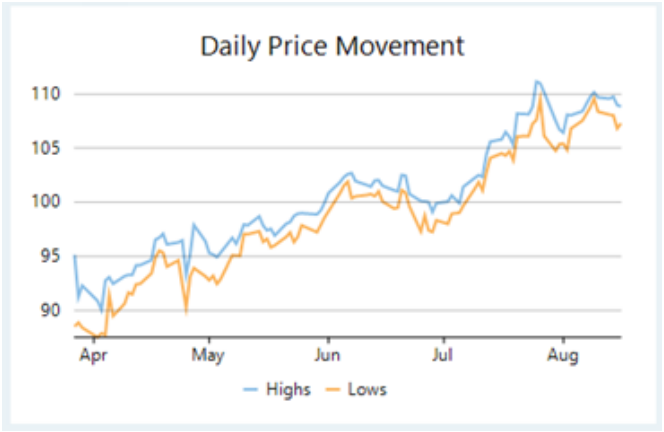
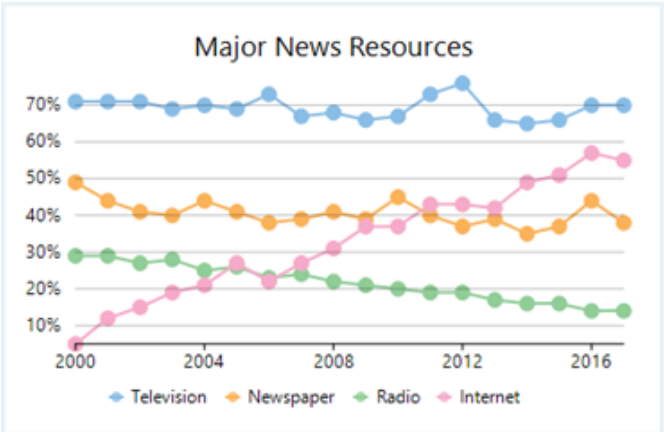
Chart Type	Chart Snapshot	Use Case
Heat Map		Use heat maps to identify the data variance with the help of color coding.
Floating Bar Chart		Use floating bar charts when you have two Y-values (high and low) for each X-value. As the name suggests, these charts plot floating bars between these Y-values to indicate maximum and minimum readings.
Gantt Chart		Use gantt charts to demonstrate the project schedules with project activities listed on Y-axis and time scale on X-axis. These charts present each activity as a horizontal bar starting at the start time of the activity with length equivalent to the duration of same.
Funnel Chart		Use funnel chart to demonstrate the gradual reduction in data as it moves through various stages of a linear process. These charts also help in identifying the problem areas in such processes.
Radar Chart		Use radar chart to visually compare multiple quantitative variables against various features where each variable is placed radially around a common center point.
Polar Chart		Similar to radar charts, use the polar charts to compare multivariate data with a difference that in this case, X-values are numeric values specifying angles in degrees.
Sunburst Chart		Use sunburst charts to present data with multiple hierarchy levels with the help of stacked slices arranged in multiple rings.
TreeMap		Use treemaps to display the hierarchical data in the form of nested rectangles, hence saving a lot of screen estate, especially in the case of huge hierarchical data sets.

折れ線チャート

Line Charts

Line charts are the most basic charts that are created by connecting the data points with straight lines. These charts are used to visualize a trend in data as they compare values against periodic intervals such as time, temperature etc. Closing prices of a stock in a given time frame, monthly average sale of a product are some good example that can be well demonstrated through a line chart.

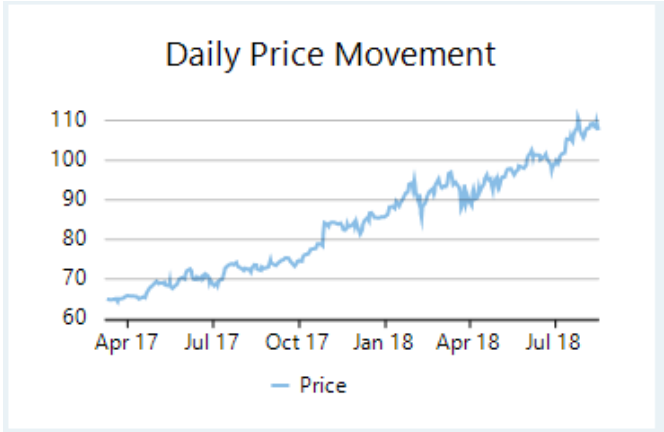
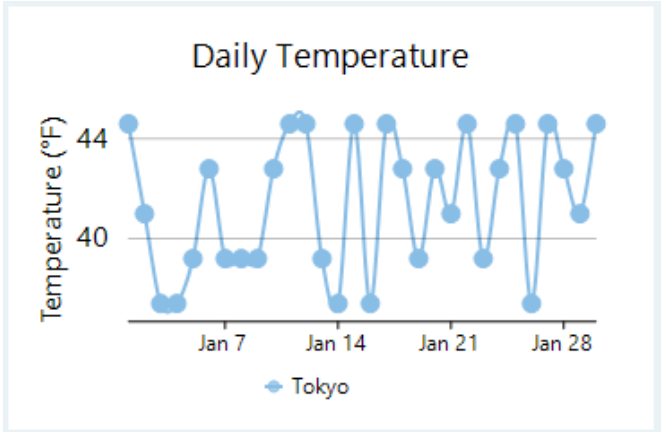
Line symbol chart is a slight variation of line chart and is displayed with markers on the data points. These charts are used when it is required to know the exact data points which have been used to plot the chart.

Line Chart	Line Symbol Chart
	

Spline Charts

Spline charts are a variation of line charts as they connect the data points using smooth curves instead of straight lines. Apart from the aesthetic aspect, these charts are preferred for displaying a gradual change in trend. Just like line chart, closing prices of a stock or life cycle of a product can be well demonstrated through a spline chart.

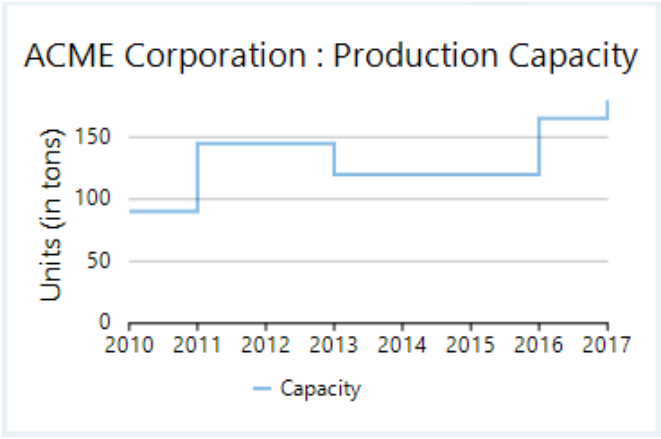
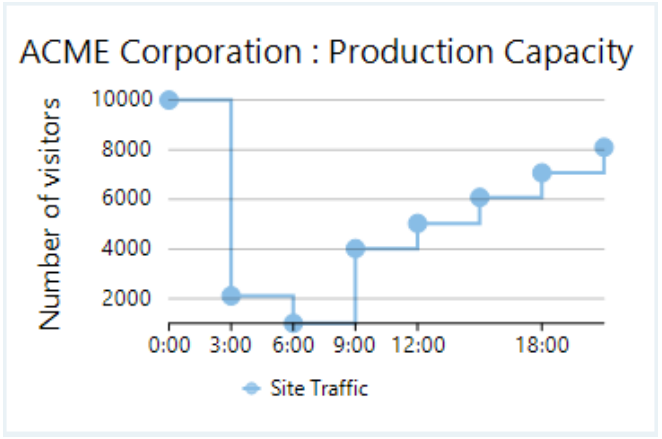
Spline symbol chart is a version of spline chart that is displayed with markers on the data points. These charts are used when it is required to know the exact data points which have been used to plot the chart.

Spline Chart	Spline Symbol Chart
	

Step Charts

Step charts use vertical and horizontal lines to connect the data points to form a step-like progression. These charts are generally used to demonstrate the sudden change in data that might occur on irregular intervals and remains constant till the next change. These charts can be used to plot the change in commodity prices, change in tax rate against income etc.

Step symbol chart is a slight variation of step chart and is displayed with markers on the data points. These charts are used when it is required to know the exact data points which have been used to plot the chart.

Step Chart	Step Symbol Chart
	

Create a Line Chart

With FlexChart, you can create these line charts by setting the ChartType property of FlexChart class as shown in the table below. This property accepts the values from **ChartType** enumeration of C1.Chart namespace.

Chart Type	Value of ChartType property
Line	Line
Line Symbol	LineSymbols
Spline	Spline
Spline Symbol	SplineSymbols
Step	Step
Step Symbol	StepSymbols

FlexChart also provides option to stack and rotate the line charts. You can set the Stacking property of FlexChart class to **Stacked** or **Stacked100pc** for stacked or stacked100 line charts respectively. This property accepts the values from Stacking enumeration of C1.Chart namespace. To rotate the line chart, that is to render the X axis vertically and Y axis horizontally, you can set the Rotated property to **true**.

For symbol charts, by default, FlexChart renders a circular symbol and a standard size and style. However, you can change the same by setting the **SymbolMarker**, **SymbolSize** and **SymbolStyle** properties of the series.

To create a line chart using FlexChart:

At design-time

1. Right click the FlexChart control on form to open the **Properties** window.
2. Navigate to the **ChartType** property and set its value to **Line** to create a simple line chart.
3. Set the data source using the DataSource property.
4. Configure X-axis and Y-axis values by setting the BindingX and Binding property respectively.

Using code

To create a line chart through code, the first step after initializing the control is to clear the default series and add a new series using the **Add** method. Set up the data source through the **DataSource** property and configure the X and Y axes by setting the **BindingX** and **Binding** property. You also need to set up the chart by setting the **ChartType** property and other required properties.

CS VB

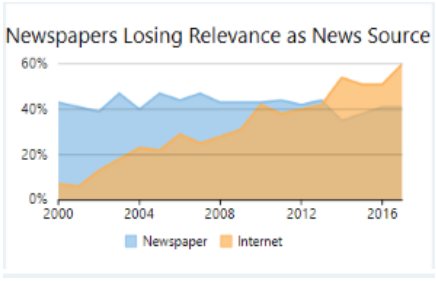
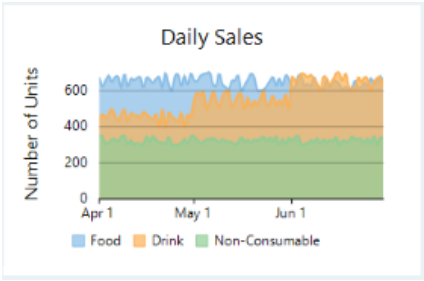
Note that the above sample code uses a custom method named GetProductRevenue to supply data. You can set up the data source as per your requirements.

CS VB

面チャート

Area charts are line charts with area between the line chart and axis filled with a color or shading. However, while line charts simply present the data values to demonstrate the trend, filled portion of the area chart helps in communicating the magnitude of the trend as well. These charts are also used to analyze a simple comparison between the trend of each category. For instance, an area chart can easily depict how internet is gradually taking over the newspaper as a source of getting news.

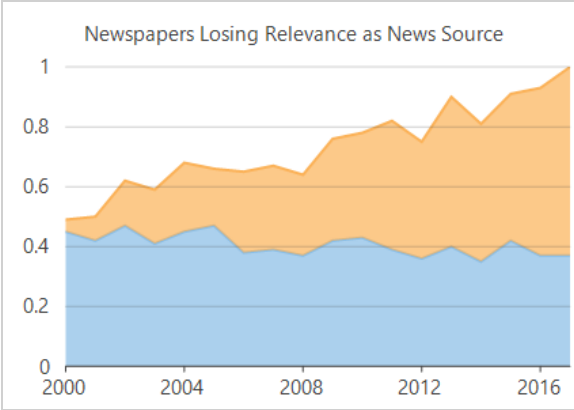
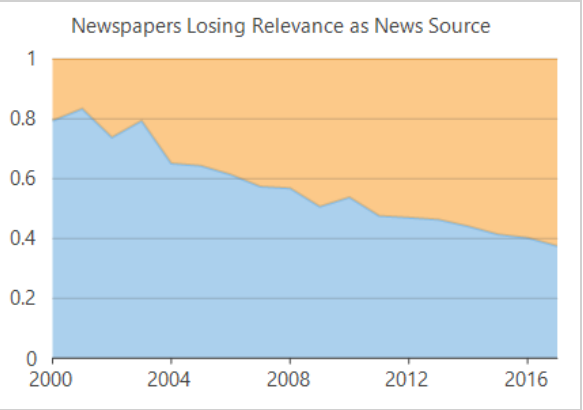
Spline area charts and **step area charts** are other modifications of area charts where area between spline chart or step chart and axis is filled with a color or shading to indicate the magnitude of change. For more information on spline chart and step chart, see Line Charts.

Area Chart	Spline Area Chart	Step Area Chart
		

Create an Area Chart

With FlexChart, you can create these charts by setting the ChartType property of FlexChart class as shown in the table below. This property accepts the values from **ChartType** enumeration of **C1.Chart** namespace.

FlexChart also provides options to stack the area charts. You can set the Stacking property of FlexChart class to **Stacked** or **Stacked100pc** for stacked or stacked100 area charts respectively. This property accepts the values from **Stacking** enumeration of C1.Chart namespace. Stacked charts are used for demonstrating the part-to-whole relationship that is, for displaying the cumulative values of categories. On the other hand, stacked 100 charts are used to present the percentage share of the values.

Stacked Area Chart	Stacked 100 Area Chart
	

To create an area chart using FlexChart:

At design-time

1. Right click the FlexChart control on form to open the Properties window.
2. Navigate to the ChartType property and set its value to "Area".
3. Set the data source using the DataSource property.
4. Configure X-axis and Y-axis values by setting the BindingX and Binding property respectively.

Using code

To create an area chart through code, the first step after initializing the control is to clear the default series and add a new series using the **Add** method. Set up the data source through the **DataSource** property and configure the X and Y axes by setting the **BindingX** and **Binding** property. You also need to set up the chart by setting the **ChartType** property and other required properties.

CS VB

Note that the above sample code uses a custom method named GetNewsSourcesInfo to supply data to the chart. You can set up the data source as per your requirements.

CS VB

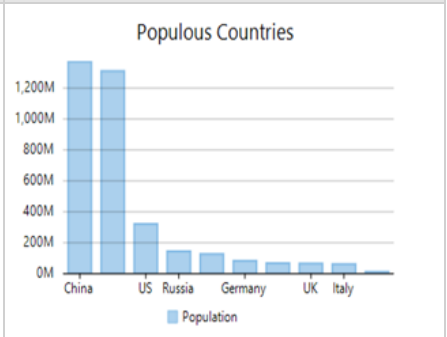
横棒および縦棒チャート

Bar Chart

Bar charts compare categorical data through the horizontal bars, where length of each bar represents the value of the corresponding category. Y-axis in these charts is a category axis. For example, sales of various product categories can be well presented through a bar chart.

Column Chart

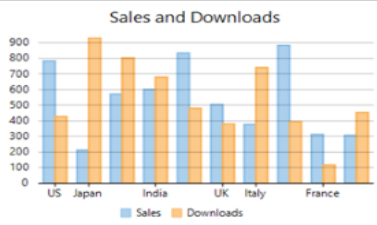
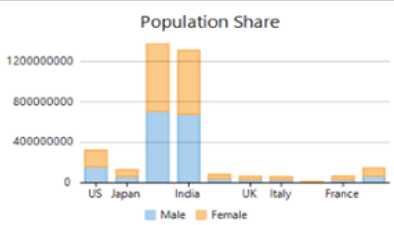
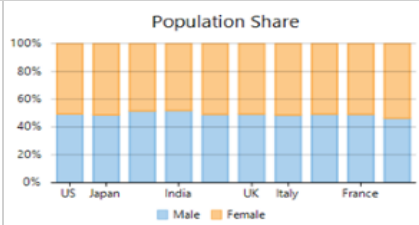
Column charts are simply vertical version of bar charts and they use X- axis as a category axis. Though bar charts and column charts can be used interchangeably, column charts are preferred where number of values is too large to be accommodated on an X-axis while bar charts are preferred in the case of long category titles which are difficult to fit on an X-axis or in the case of large number of categories. For example, population share of different countries across the globe is good example that can be demonstrated using a column chart.

Bar Chart	Column Chart
	

Create a Bar or Column Chart

With FlexChart, you can create a bar chart and a column chart by setting the ChartType property of FlexChart class to **Bar** or **Column**. This property accepts the values from **ChartType** enumeration of C1.Chart namespace.

FlexChart also provides options to cluster and stack the bar and column charts. While cluster is simply created by adding the multiple series with same base category to the chart, stacking can be done by setting the Stacking property of FlexChart class to **Stacked** or **Stacked100pc**. Stacked charts are used for demonstrating the part-to-whole relationship that is, for displaying the cumulative values of categories. For example, population of each gender in different countries can be presented by a stacked column chart. On the other hand, stacked 100 charts are used to present the percentage share of the values.

Clustered Column Chart	Stacked Column Chart	Stacked 100 Column Chart
		

To create a bar or column chart using FlexChart:

At design-time

1. Right click the FlexChart control on form to open the Properties window.
2. Navigate to the ChartType property and set its value to "Bar" or "Column".
3. Set the data source using the DataSource property.
4. Configure X-axis and Y-axis values by setting the BindingX and Binding property respectively.

Using code

To create a bar chart through code, the first step after initializing the control is to clear the default series and add a new series using the **Add** method. Set up the data source through the **DataSource** property and configure the X and Y axes by setting the **BindingX** and **Binding** property. You also need to set up the chart by setting the **ChartType** property and other required properties.

CS VB

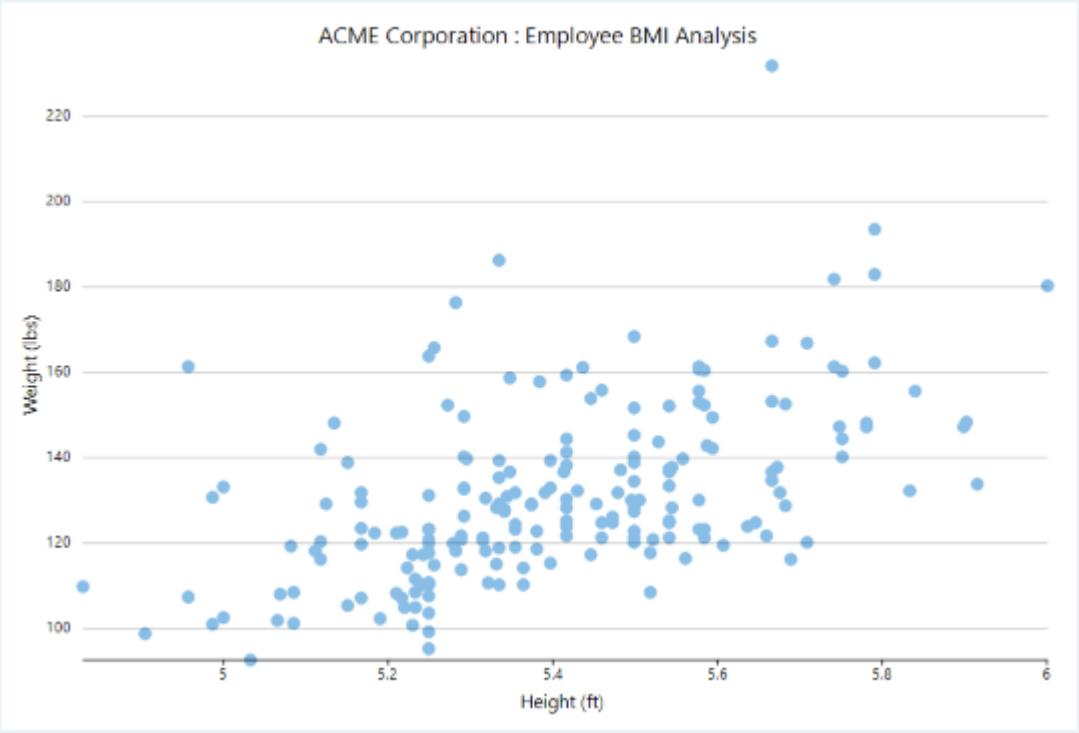
Note that the above sample code uses a custom method named GetData to supply data. You can set up the data source as per your requirements.

CS VB

点チャート

Scatter Chart

Scatter charts are very powerful type of charts which plot pairs of numerical data in order to identify a relationship between the two variables. Scatter charts show how much one variable is affected by another. They are best for comparing large number of data points. For instance, following chart displays the relationship between weight and height of the employees working in a company.



Create a Scatter Chart

With FlexChart, you can create these charts by setting the ChartType property of FlexChart class to **Scatter**. This property accepts the values from **ChartType** enumeration of C1.Chart namespace. By default, FlexChart renders the scatter chart series with a circular symbol and a standard size and style. However, you can change the same by setting the SymbolMarker, SymbolSize and SymbolStyle properties of the series. To rotate a scatter chart, that is to

render the X axis vertically and Y axis horizontally, you can set the Rotated property to **true**.

To create a scatter chart using FlexChart:

At design-time

1. Right click the FlexChart control on form to open the **Properties** window.
2. Navigate to the **ChartType** property and set its value to **Scatter**.
3. Set the data source using the DataSource property.
4. Configure X-axis and Y-axis values by setting the BindingX and Binding property respectively.

Using code

To create a scatter chart through code, the first step after initializing the control is to clear the default series and add a new series using the **Add** method. Set up the data source through the **DataSource** property and configure the X and Y axes by setting the **BindingX** and **Binding** property. You also need to set up the chart by setting the **ChartType** property and other required properties.

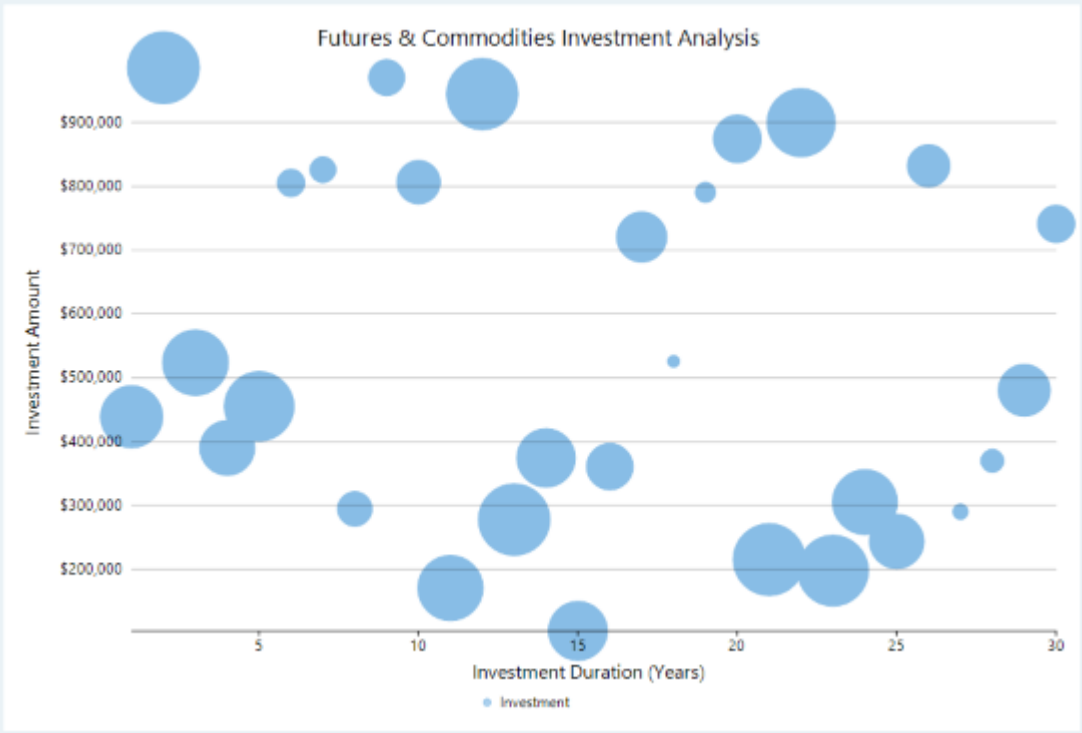
CS VB

Note that the above sample code uses a custom method named GetPointData to supply data. You can set up the data source as per your requirements.

CS VB

Bubble Chart

Bubble charts are similar to scatter charts and are used when you have a third dimension to plot. This third dimension is depicted through the size of the bubbles. These charts are generally used to plot financial data, for example, below chart presents a relationship between investment duration, amount of investment and return on investment.



Create a Bubble Chart

With FlexChart, you can create these charts by setting the ChartType property of FlexChart class to **Bubble**. This property accepts the values from **ChartType** enumeration of C1.Chart namespace. By default, FlexChart renders the bubble chart series with a circular symbol and a standard style. However, you can change the same by setting the SymbolMarker and SymbolStyle properties of the series. You can also define the maximum and minimum size of the bubbles by setting the BubbleMaxSize and BubbleMinSize properties respectively.

FlexChart also provides option to rotate the bubble charts, that is to render the X axis vertically and Y axis horizontally. To rotate a bubble chart, you can set the Rotated property to **true**.

To create a bubble chart using FlexChart:

At design-time

1. Right click the FlexChart control on form to open the **Properties** window.
2. Navigate to the **ChartType** property and set its value to **Bubble**.
3. Set the data source using the DataSource property.
4. Configure X-axis and Y-axis values by setting the BindingX and Binding property respectively.

Using code

To create a bubble chart through code, the first step after initializing the control is to clear the default series and add a new series using the **Add** method. Set up the data source through the **DataSource** property and configure the X and Y axes by setting the **BindingX** and **Binding** property. You also need to set up the chart by setting the **ChartType** property and other required properties.

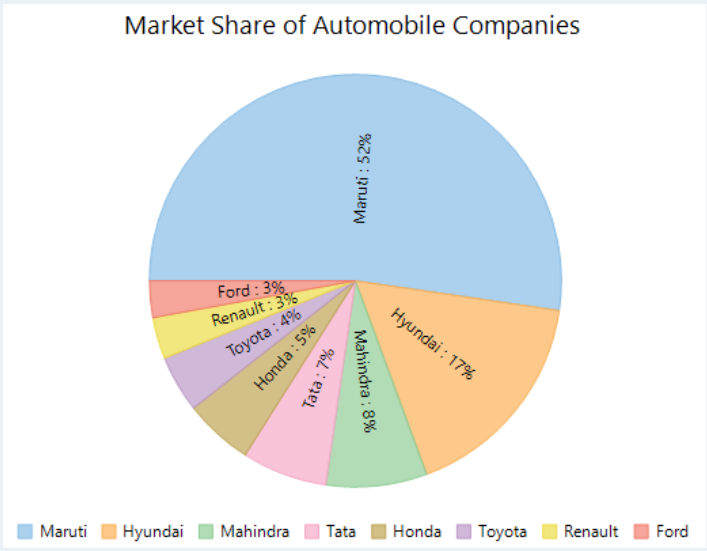
CS | VB

Note that the above sample code uses a custom method named GetData to supply data. You can set up the data source as per your requirements.

CS VB

円チャート

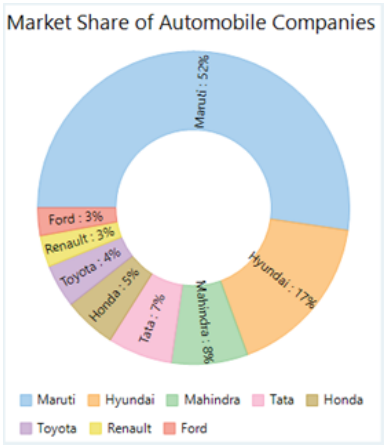
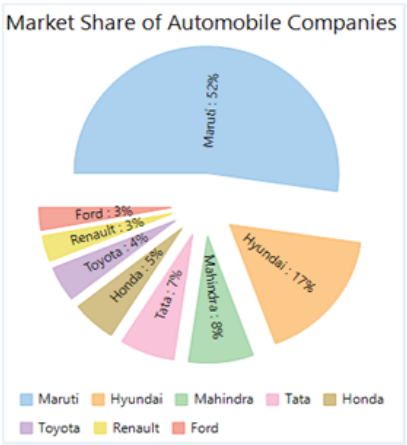
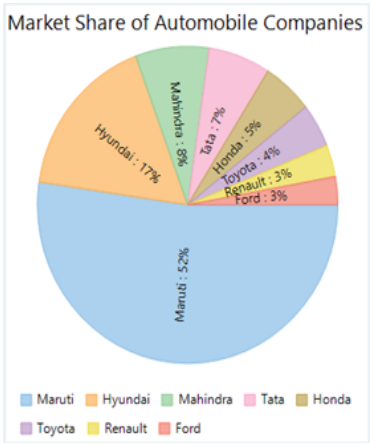
Pie charts, the most common tools used for data visualization, are circular graphs that display the proportionate contribution of each category which is represented by a pie or a slice. The magnitude of the dependent variable is proportional to the angle of the slice. These charts can be used for plotting just one series with non-zero and positive values. For example, you can use a pie chart to show the market share of various automobile companies.



Create a Pie Chart

FlexChart for WinForms provides pie chart through a stand alone control which is represented by the FlexPie class. You can bind the chart with data using the FlexPie.DataSource property provided by the **FlexPie** class. This class also provides FlexPie.Binding and FlexPie.BindingName properties for setting numeric values and labels of the pie chart slices. You can also specify the angle from where you want to start drawing the slices in the clock wise direction by setting the StartAngle property. FlexChart also lets you create following variations of pie chart:

- **Doughnut Chart:** Set the InnerRadius property to a value greater than zero to create a hole in the center. By default, this property is set to **zero**.
- **Exploded Pie Chart:** Set the **Offset** property to a value greater than zero to push the slices away from the center. By default, this property is set to **zero**.
- **Reversed Pie Chart:** Set the **Reversed** property to **true** that creates the chart with angles drawn in the counter clockwise direction. By default, this property is set to **false**.

Doughnut Chart	Exploded Pie Chart	Reversed Pie Chart
		

To create a pie chart using FlexChart:

At design-time

1. Drag and drop the FlexPie control to the form.
2. Right click the FlexPie control on form to open the Properties window.
3. Set the data source using the DataSource property.
4. Set the Binding and BindingName property.

Using code

To create a pie chart through code, the first step after initializing the FlexPie control is to clear the default series and add a new series using the Add method. Set up the data source through the **DataSource** property and set the **BindingName** and **Binding** property. You can also set up the chart further by setting the other required properties such as header content, data label content etc.

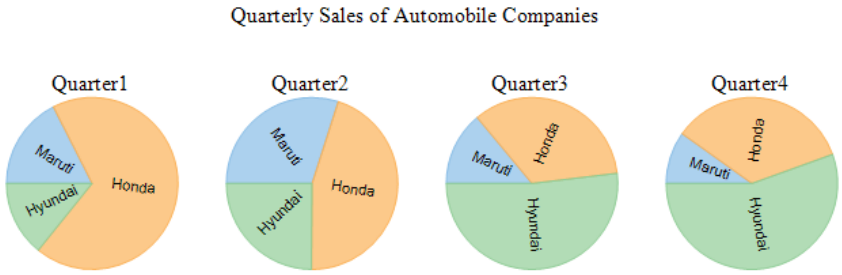
CS VB

Note that the above sample code uses a custom method named GetCarSales to supply data. You can set up the data source as per your requirements.

CS VB

Create Multiple Pie Charts

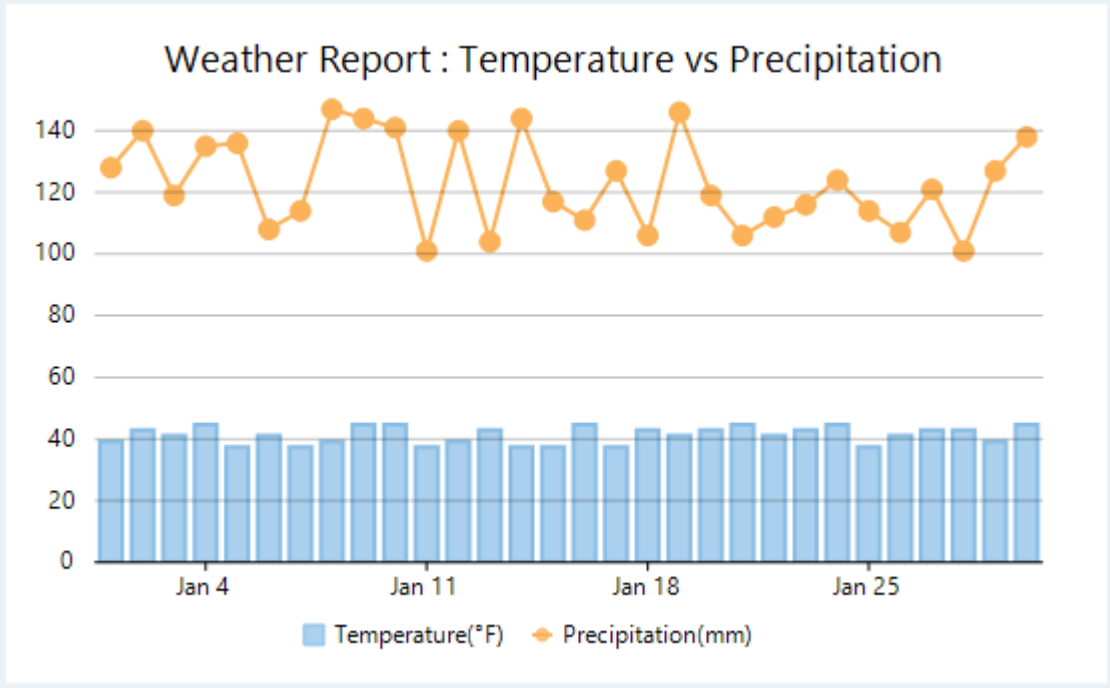
While a single pie chart is used to present the part-to-whole relationship in a group, the multiple pies are required for presenting or comparing data across various groups. For example, you can use multiple pies to compare quarterly sales figures of various car brands. With FlexChart, you can achieve this by simply setting the comma-separated field names from a data source in Binding property of the FlexPie class.



CS VB

複合チャート

Combination charts are the combination of two or more chart types in a single plot area, for instance, a bar and a line chart laid in a single plot. Combination charts are best used to compare the different data sets that are related to each other such as actual and target values, total revenue and profit, temperature and precipitation etc. Note that these charts may require multiple axes to cater different scales for different values. To know more about multiple axes, see Axes.



Create a Combination Chart

With FlexChart, you can create a combination chart by specifying a chart type for every series. You can do this by setting the ChartType property of the SeriesBase class. Setting this property overrides the ChartType property set for the chart.

To create a mixed chart using FlexChart:

At design-time

1. Right click the FlexChart control on form to open the **Properties** window.
2. Set values of DataSource property and BindingX property.
3. Navigate to the **Series** field and click ellipsis button to open the **Series Collection Editor**.
4. In Series Collection Editor, click **Series 1** to set its Binding and **ChartType** properties.
5. Click the **Add** button to add a new series **Series 2**.
6. Set the Binding and ChartType properties for Series 2.

Using code

To create a mixed chart through code, the first step after initializing the control is to clear the default series. Set up the data source through the **DataSource** property and configure the X axis by setting the **BindingX** property. Now, add a new series using the **Add** method and set the **Binding** and **ChartType** property for same. Similarly, add another series to the **Series** collection and set the ChartType, Binding and other relevant properties as required for this series.

CS VB

```
```

Note that the above sample code uses a custom method named GetTemperatureData to supply data to the chart. You can set up the data source as per your requirements.

CS VB

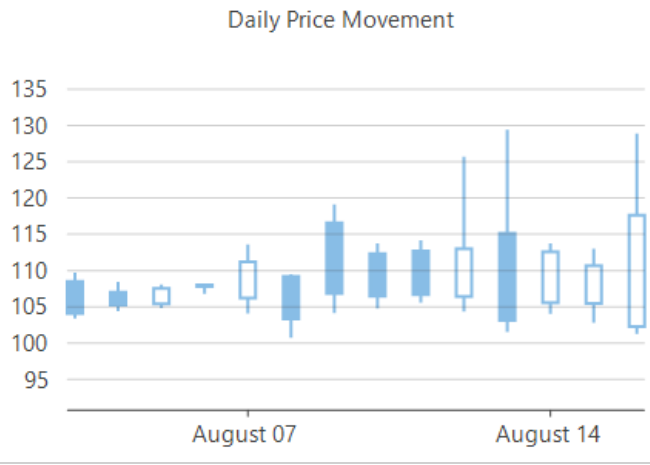
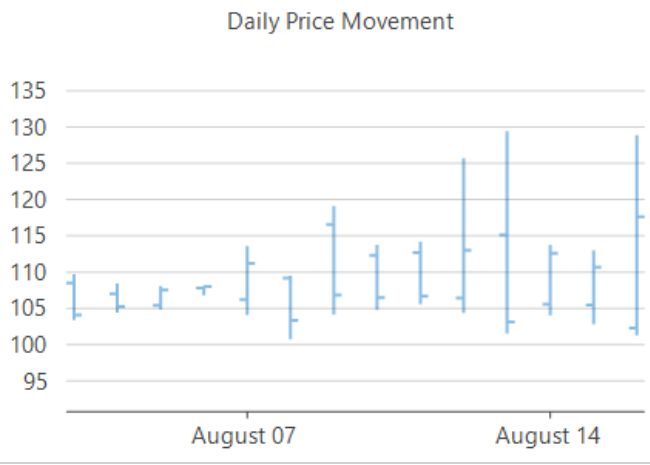
財務チャート

Candlestick Chart

Candlestick charts are financial charts to plot open, close, high and low values of a stock over short periods of time. Each candlestick in these charts consist of a body, a wick and a tail. While body of a candlestick chart represents the opening and closing values of the stock in a particular span of time, wick and tail, the vertical lines above and below the body, indicate the highest and lowest value of stock in that time span respectively. Being packed with so much information in a single chart, these charts are used in tracking the price movements of stock.

HighLowOpenClose (HLOC) Chart

Like Candlestick charts, **HighLowOpenClose charts** or **HLOC charts** are also used to plot high, low, open and close values of stocks in a specified time frame and are used for stock analysis. The only difference from a candlestick chart is that the HLOC charts are drawn without candlestick 'body'.

Candlestick Chart	HighLowOpenClose Chart
	

Create a Candlestick or HLOC Chart

With FlexChart, you can create Candlestick or HLOC chart by setting the ChartType property of FlexChart class to **Candlestick** or **HighLowOpenClose**. This property accepts the values from ChartType enumeration of C1.Chart namespace. You can also change the size of the candle or symbols in HLOC chart by setting the **SymbolSize** properties of the series.

To create a candlestick or HLOC chart using FlexChart:

At design-time

1. Right click the FlexChart control on form to open the **Properties** window.
2. Navigate to the **ChartType** property and set its value to **Candlestick** or **HighLowOpenClose**.
3. Set the data source using the DataSource property.
4. Configure X-axis and Y-axis values by setting the BindingX and Binding property respectively.

Using code

To create a HLOC or candlestick chart through code, the first step after initializing the control is to clear the default series and add a new series using the **Add** method. Set up the data source through the **DataSource** property and configure the X and Y axes by setting the **BindingX** and **Binding** property. You also need to set up the chart by setting the **ChartType** property and other required properties.

CS VB

Note that the above sample code uses a custom method named GetQuotes to supply data to the chart. You can set up the data source as per your requirements.

CS VB

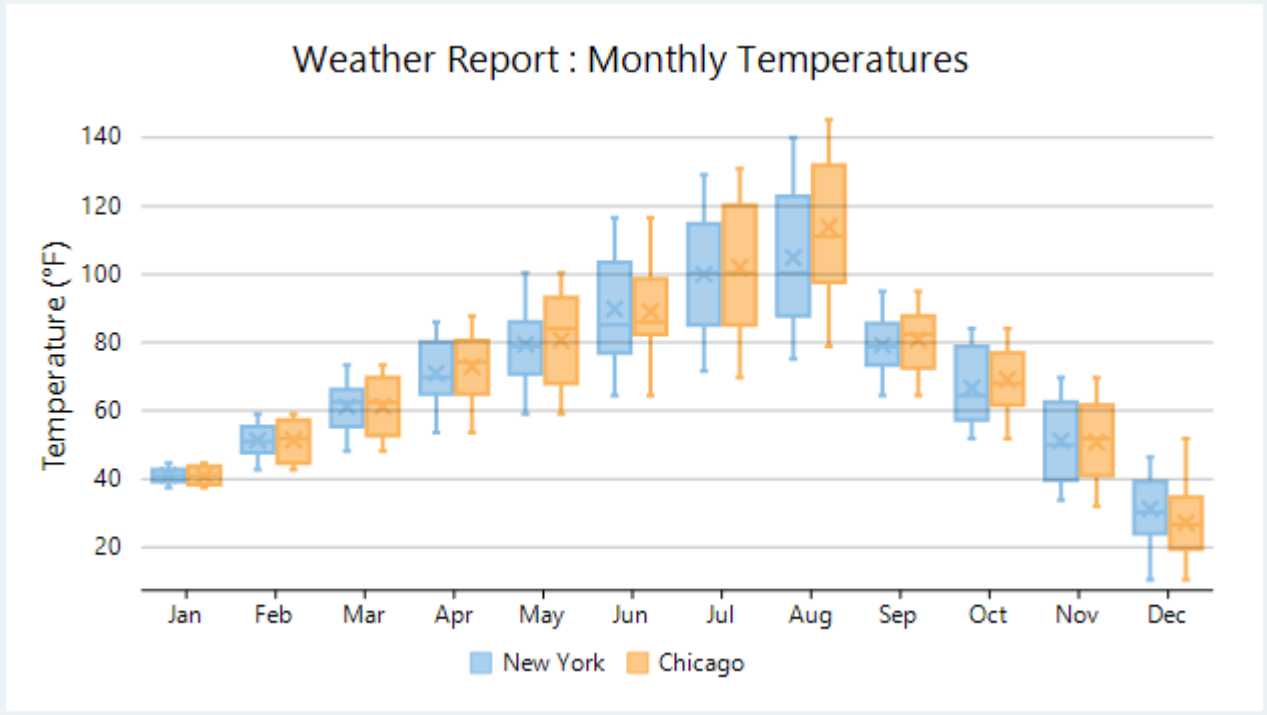
統計チャート

Statistical charts help in presenting the statistical data in graphical format and make it easier to understand and interpret. This section discusses various statistical charts which can be created using the FlexChart control.

Topic	Content
Box-and-Whisker	Discusses how to create a Box-and-Whisker chart using FlexChart.
Error Bar	Discusses how to create an Error Bar chart using FlexChart.
Histogram	Discusses how to create a Histogram chart using FlexChart.
Ranged Histogram	Discusses how to create a Ranged Histogram chart using FlexChart.
Pareto Chart	Discusses how to create a Pareto chart using FlexChart.
Waterfall Chart	Discusses how to create a Waterfall chart using FlexChart.

箱ひげ図

Box-and-Whisker charts are the statistical charts that display the distribution of numerical data through quartiles, means and outliers. As the name suggests, these values are represented using boxes and whiskers where boxes show the range of quartiles (lower quartile, upper quartile and median) while whiskers indicate the variability outside the upper and lower quartiles. Any point outside the whiskers is said to be an outlier. These charts are useful for comparing distributions between many groups or data sets. For instance, you can easily display the variation in monthly temperature of two cities.



Create a Box-and-Whisker Chart

In FlexChart, Box-and-Whisker can be implemented using the BoxWhisker class which represents a Box-and-Whisker series. Apart from other series related properties, this class provides properties specific to Box-and-Whisker series such as the **QuartileCalculation** property, which lets you specify whether to include the median in quartile calculation or not. This property accepts the values from QuartileCalculation enumeration. FlexChart also provides options to specify whether to display outliers, inner points, mean line and mean marks through ShowOutliers, ShowInnerPoints, ShowMeanLine and ShowMeanMarks properties respectively.

To create a box-and-whisker chart through code, the first step after initializing the control is to clear the default series and add a new series using the **Add** method. Set up the data source through the **DataSource** property and configure the X and Y axes by setting the **BindingX** and **Binding** property. You also need to set up the chart by setting the **ChartType** property and other required properties.

CS VB

```
```

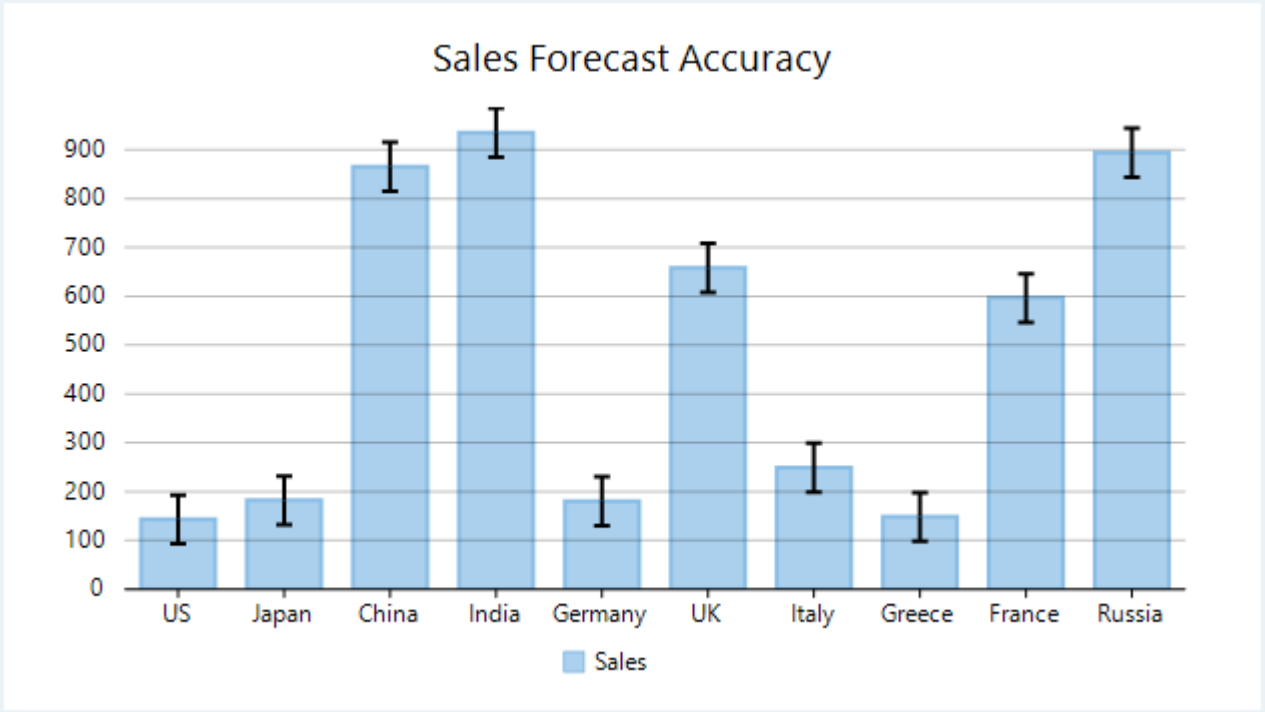
Note that the above sample code uses a custom methods named GetTemperatureData and GetMonthAxisDataSource to supply data to the chart. You can set up the data source as per your requirements.

CS VB

```
```

エラーバー(誤差範囲)

Error bars are the charts that indicate the estimated error or uncertainty in the measured data to give an idea about how precise that data is. They most often represent this through the standard deviation of data set. Error bars are, generally useful while plotting results of scientific studies, experiments, or any other measurements that show variation in data from original values. Deviation in sales forecast is one of the good examples that can be presented using these error bars.



Create an Error Bar Chart

In FlexChart, error bars can be shown along with various chart types such as line, area, column, scatter and, spline charts. Error bars can be implemented using the ErrorBar class which represents an error bar series. Apart from other series related properties, this class provides properties specific to error bar series such as the **ErrorAmount** property, which lets you specify the error amount of the series as a standard error amount, a percentage or a standard deviation. This property accepts the values from ErrorAmount enumeration. FlexChart also provides options to specify whether to display error bars in plus, minus or both directions using the Direction property. EndStyle property of FlexChart lets you specify whether to display caps or not at the ends of error bars. To customize the appearance of bar style, you can use the ErrorBarStyle property.

CS VB

```
```

Note that the above sample code uses a custom method named GetCountrySales to supply data to the chart. You can set up the data source as per your requirements.

CS VB

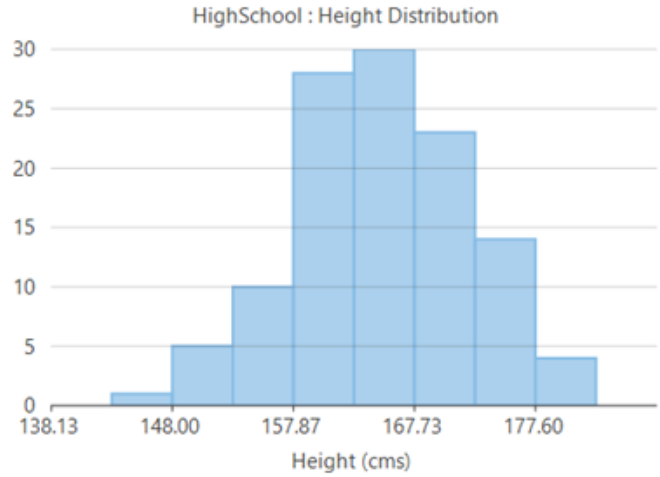
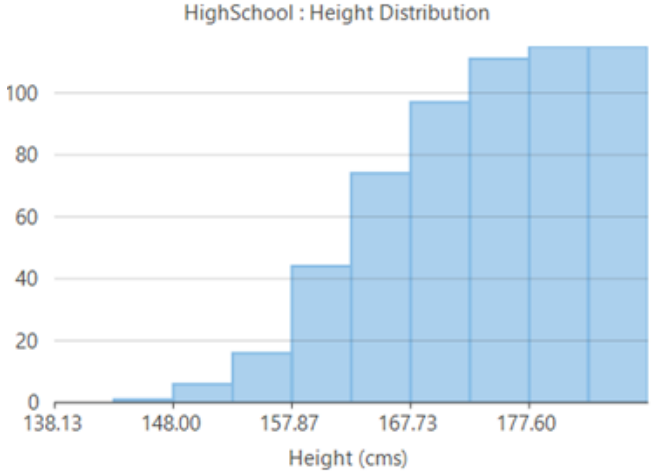
```
```

ヒストグラム

Histogram

Histograms are visual representation of data distribution over a continuous interval or certain time period. These charts comprise of vertical bars to indicate the frequency in each interval or bin created by dividing the raw data values into a series of consecutive and non-overlapping intervals. Hence, histograms help in estimating the range where maximum values fall as well as in knowing the extremes and gaps in data values, if there are any. For instance, histogram can help you find the range of height in which maximum students of a particular age group fall.

Cumulative histograms are a variation of histograms which plot cumulative frequencies instead of frequencies and hence, demonstrate the cumulative number of readings in all the bins up to a specified bin.

Histogram	Cumulative Histogram
	

Create a Histogram

With FlexChart, you can create a histogram by setting the ChartType property to **Histogram** and adding a Histogram series to the chart. Once data is provided, FlexChart automatically calculates the bins to group the data and create a histogram. However, if required, you can also set width of these bins by setting the **BinWidth** property. To create a cumulative histogram, you need to set the **CumulativeMode** property to **true**.

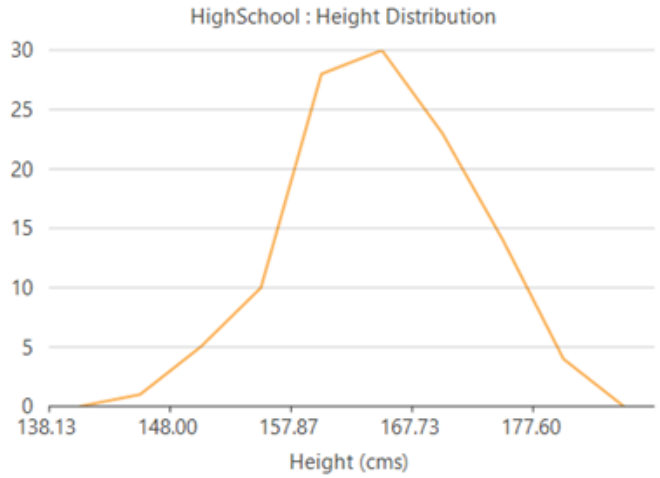
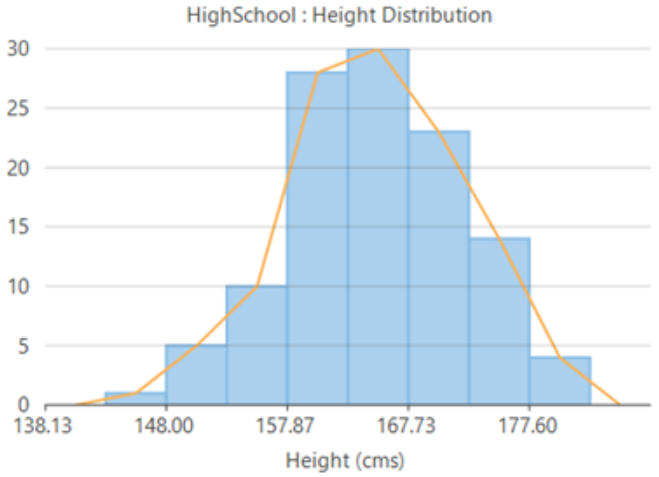
CS VB

Note that the above sample code uses a custom method named GetPointData to supply data to the chart. You can set up the data source as per your requirements.

CS VB

Frequency Polygon

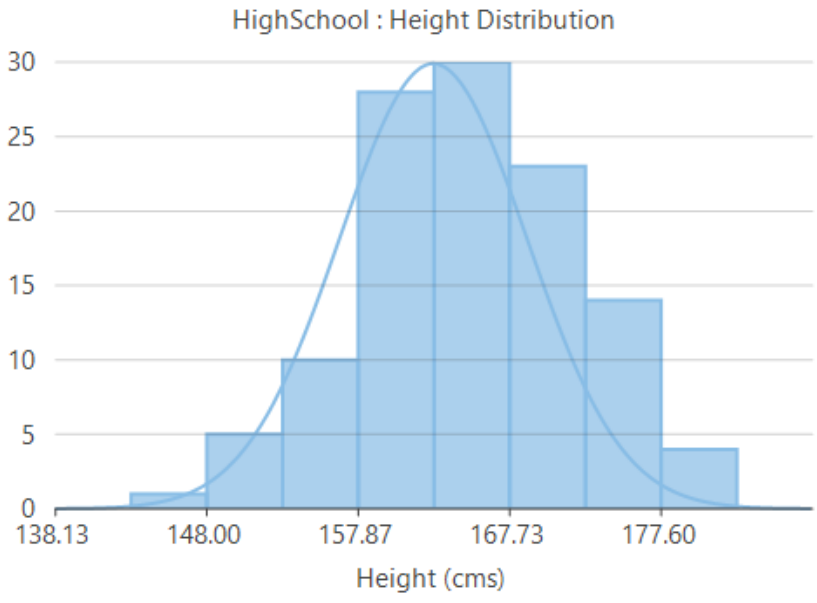
Frequency polygons are another variation of histograms that are used to depict the overall pattern of frequency distribution in the data. These are created by joining the mid points of the top of the bars of histogram through straight line segments to form a closed two-dimensional figure. With FlexChart, you can create a frequency polygon by setting the **HistogramAppearance** property to **FrequencyPolygon** or **HistogramAndFrequencyPolygon**, depending on whether you want to display it alone or along with a histogram. This property accepts values from the **HistogramAppearance** enumeration. FlexChart also lets you change the appearance of frequency polygon by setting the **FrequencyPolygonStyle** property.

Frequency Polygon	Frequency Polygon with Histogram
	

CS VB

Gaussian Curve or Normal curve

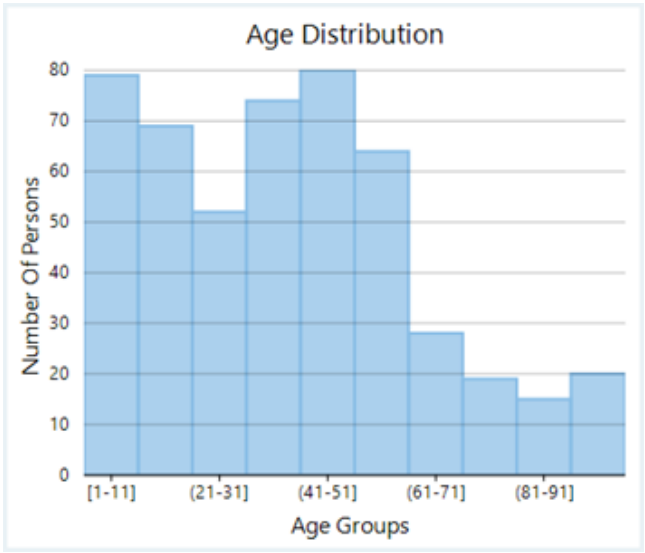
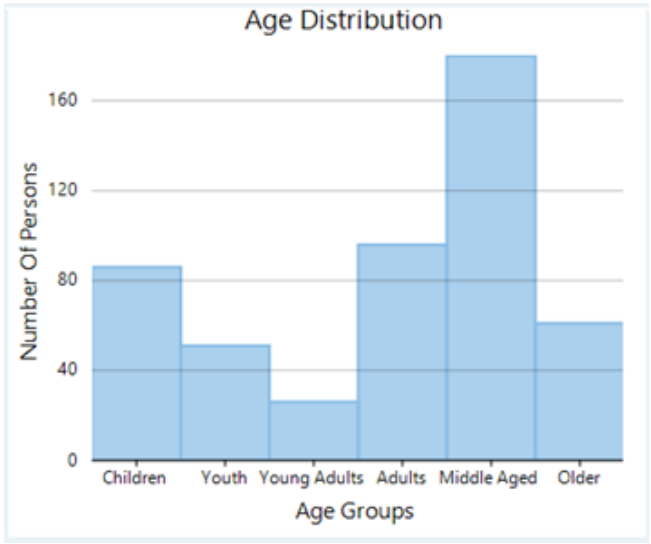
Gaussian curve is a bell shaped curve, also known as **normal curve**, which represents the probability distribution of a continuous random variable. This curve is another variation of histogram and can be created on a histogram by setting the **Visible** property of **NormalCurve** class to **true**. You can also change the appearance of normal curve by setting the **LineStyle** property of this class.



CS VB

RangedHistogram

Ranged histogram is a modern Excel-like histogram chart that helps visualize frequency distribution on Y-axis, against ranged X-axis. Like histogram chart type, bins are created by dividing the raw data values into a series of consecutive, non-overlapping intervals. Based on the number of values falling in a particular bin or category, frequencies are then plotted as rectangular columns against X-axis. FlexChart lets you render the ranged histogram in non-category or category mode depending on what do you want to display on the X-axis.

Ranged Histogram in Non-category Mode	Ranged Histogram in Category Mode
	

Non-category Mode

In non-category mode, the original data points are binned into intervals or ranges. For instance, the age distribution of a population plotted against age divided into various ranges is one good example of histogram in non-category mode.

With FlexChart, you can create a ranged histogram in non-category mode by setting the ChartType property to **RangedHistogram** and adding a RangedHistogram series to the chart. Once data is provided, FlexChart automatically calculates the bins to group the data and create a ranged histogram. However, you can also choose whether you want to render the chart with a specified bin width or a specified bin count by setting the BinMode property which accepts the values from HistogramBinning enumeration. You can further set BinWidth, NumberOfBins, UnderflowBin, OverflowBin and can even specify whether to ShowUnderflowBin and ShowOverflowBin in the ranged histogram.

CS VB

Category Mode

In category mode, frequency data is exclusively grouped in categories (which are plotted on X-axis) as provided by the original

data and Y-axis depicts cumulative frequency for the respective categories. For instance, the age distribution of a population can also be plotted by dividing the data into age groups as shown in the image above.

In FlexChart, you need to set the ChartType property to **RangedHistogram**, add a RangedHistogram series to the chart and set the BindingX property to enable the category mode. After this, properties such as BinMode, BinWidth, NumberOfBins, OverflowBin, and UnderflowBin are ignored.

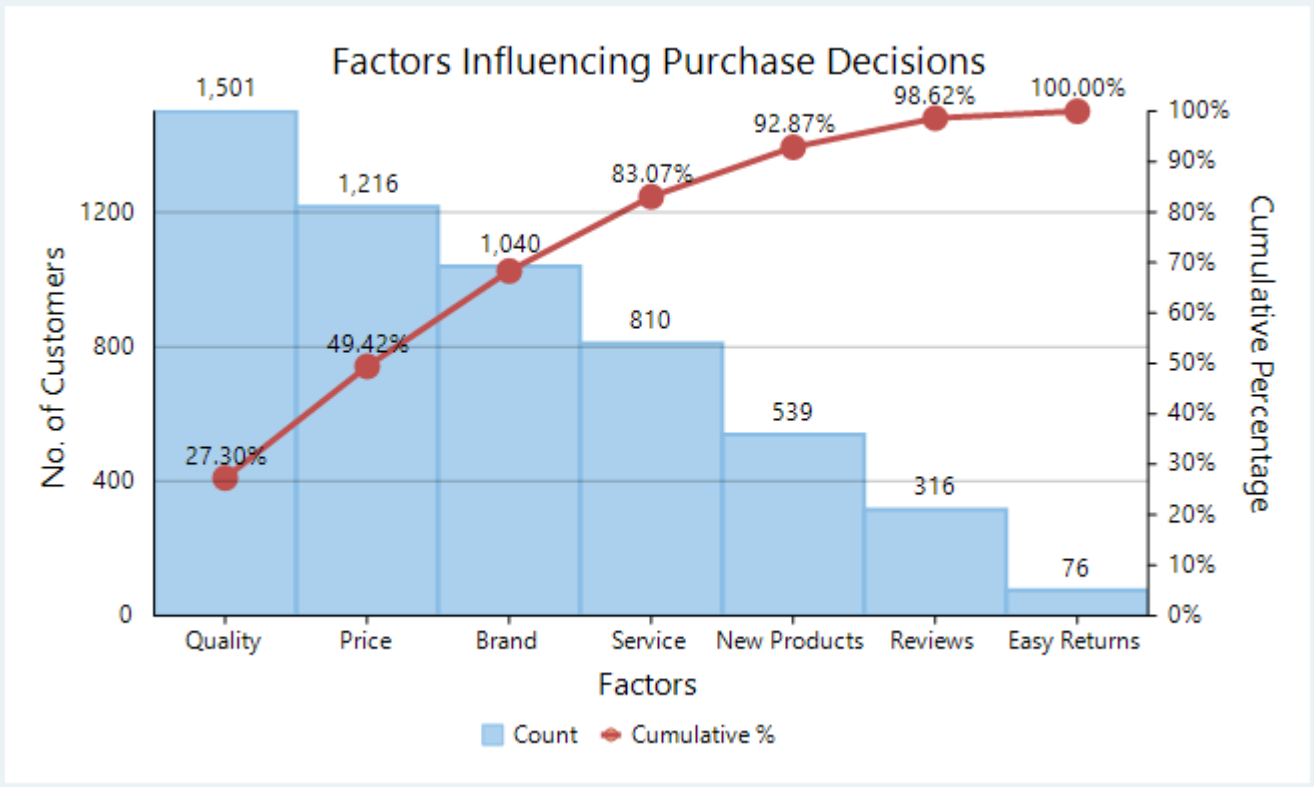
CS VB

Note that the above sample codes use a custom method named GetAgeData to supply data to the chart. You can set up the data source as per your requirements.

CS VB

パレート図

Pareto charts graphically summarize the process problems in ranking order from the most frequent to the least one. These charts comprise of both a bar and a line chart where bar chart values are plotted in decreasing order of relative frequency while the line chart plots the cumulative total percentage of frequencies. Pareto chart is essentially used in scenarios where the data is broken into different categories, and when the developer needs to highlight the most important factors from a given set of factors. For example, quality control, inventory control, and customer grievance handling are some areas where Pareto chart analysis can be frequently used.



With FlexChart, Pareto chart can be easily created by combining ranged histogram chart with any of line, spline, line symbol, or spline symbol chart. First, plot the relative frequency on a ranged histogram in descending order. Then, calculate the cumulative relative frequency in percentage using original data to create another series which is plotted using any of the line, spline, line symbol, or spline symbol chart. This forms Pareto line of the chart which helps in identifying the added contribution of each category.

CS VB

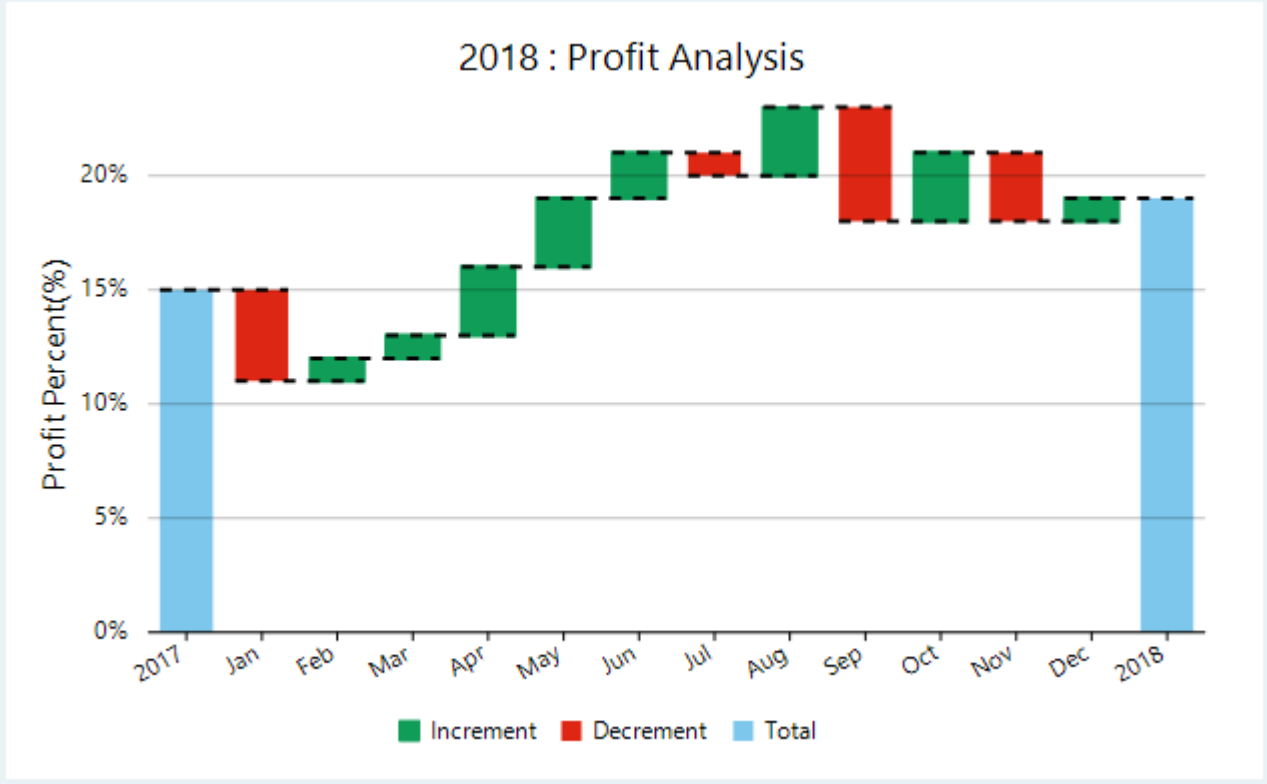
Note that the above sample code uses a custom method named GetPurchaseFactorsData to supply data to the chart. You can set up the data source as per your requirements.

CS VB

ウォータフォール

Waterfall charts are the statistical charts that demonstrate the cumulative effect of increasing and decreasing intermediate values on an initial value to result in a final value. These charts generally represent the initial and final values as blue colored total columns and intermediate values as green and red floating columns for increments and

decrements respectively. These charts are helpful in scenarios such as viewing fluctuations in product earnings or for profit analysis as shown in the chart below.



Create a Waterfall Chart

In FlexChart, a waterfall chart can be implemented using the Waterfall class which represents a waterfall series. Apart from other series related properties, this class provides properties specific to waterfall series such as the ShowTotal or ShowIntermediateTotal properties, which let you specify whether to display the total and intermediate total columns or not. You can also choose whether to display the ConnectorLines. FlexChart also allows you to change the style of these columns by setting the RisingStyle, FallingStyle, TotalStyle, and StartStyle properties respectively.

CS VB

Note that the above sample code uses a custom method named GetProfitStatement to supply data to the chart. You can set up the data source as per your requirements.

CS VB

特集なチャート

This section discusses the specialized chart types provided by the FlexChart control.

Topic	Content
Heat Map	Discusses different types of heat maps, color axis and how to implement them.
Floating Bar Chart	Discusses floating bar chart and how to implement them.
Gantt Chart	Discusses gantt chart and how to implement them.
Funnel Chart	Discusses funnel charts, related properties and how to implement them.
Radar Charts	Discusses radar charts, related properties and how to implement them.
Sunburst	Discusses sunburst charts, related properties and how to implement them.
TreeMap	Discusses treemap chart, related properties and how to implement them.

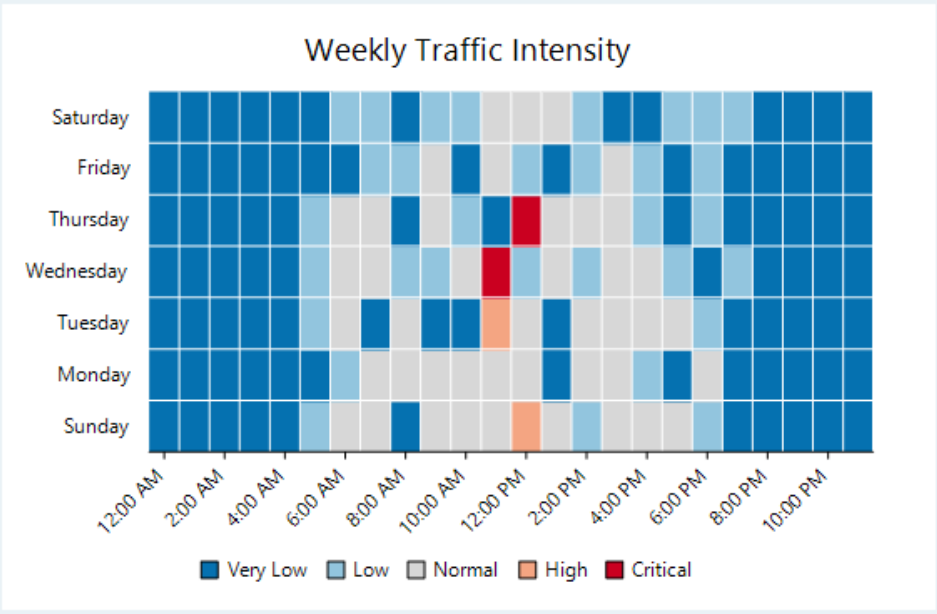
ヒートマップ

Heat maps are the graphical representations that use color coding to plot the two dimensional data. The primary purpose of heat maps is to identify patterns, areas of concentration, and data variance. Some of the real life examples where heat maps could be used are temperature records or demonstrating weekly traffic intensity.

Depending on the type of data used, heat maps can be created using two types of visual representations: category-based heat maps and value-based heat maps.

Category-based Heat Map

Category-based heat maps are used to plot data against intervals or specific numeric ranges. To create a category-based heat map using FlexChart, you need to create an instance of the DiscreteColorScale class. This class provides the Intervals property which allows you to get or set the collection of intervals. You can define the intervals by setting the Min, Max, Color and Name properties of each interval. These user-defined intervals form legend of the heat map. Next step is to create a Heatmap series, supply data to it by setting the DataSource property and set its ColorScale property to the **DiscreteColorScale** object.



CS VB

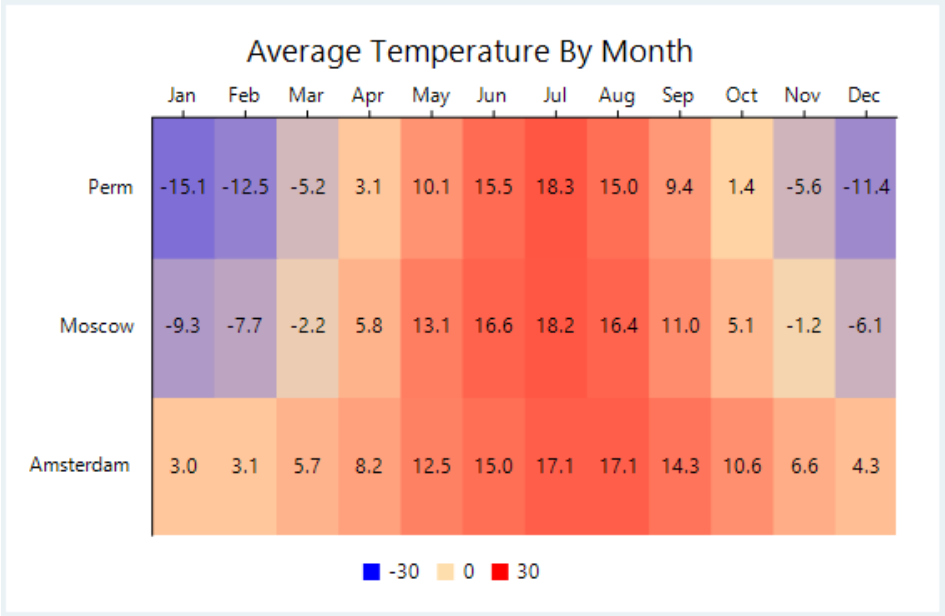
Note that the above sample code uses a custom method named Get2DTrafficData to supply data to the chart. You can set up the data source as per your requirements.

CS VB

Value-based Heat Map

Value-based heat maps are used to plot the raw data values without dividing them into the intervals. To create a value-based heat map using FlexChart, you need to create an instance of the GradientColorScale class. The legend of a value based heat map is generated with each entry corresponding to a numeric value between the Min and Max values. The interval between these values is calculated based on the formula (|Min| + |Max|)/(n-1) where **n** refers to the number of colors specified in the Colors property. Next step is to create a Heatmap series, supply data to it by setting the DataSource property and set its ColorScale property to the **GradientColorScale** object.

For example, a simple custom palette containing red, white and blue color maps the values from -30 to 30 to shades of red, white, blue, where -30 is mapped to red and 30 to blue.



CS VB

Note that the above sample code uses a custom method named Get2DTempData to supply data to the chart. You can set up the data source as per your requirements.

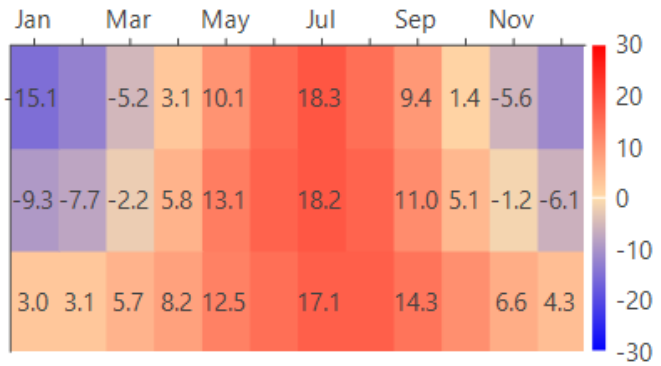
CS VB

Use Gradient Legend

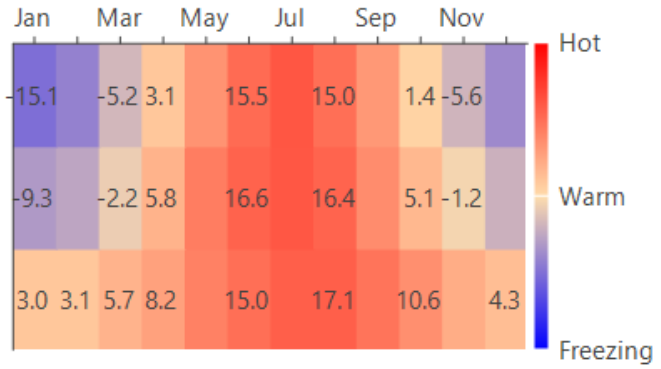
By default, like any other chart, a heat map also gets generated with a legend. However, the default chart legend of a heat map can be replaced with a gradient chart legend by using the ColorAxis class. Gradient chart legend is a small bar of integrated colors obtained from the Colors definitions. Each color is for a specific numeric range and it integrates with the next color, creating a gradient legend. Every point on the gradient legend represents a distinctive color and value. Therefore, all the dissimilar values in the chart appear in distinctive colors as per the position on the gradient legend.

To implement heat map with a gradient legend, you need to provide an instance of **ColorAxis** class to the Axis property provided by the GradientColorScale class. Moreover, you also need to specify the Min, Max and Colors property to set up a gradient legend. You can also choose to label the gradient legend with defined categories instead of values by providing a data source for the color axis.

Gradient Legend with Values



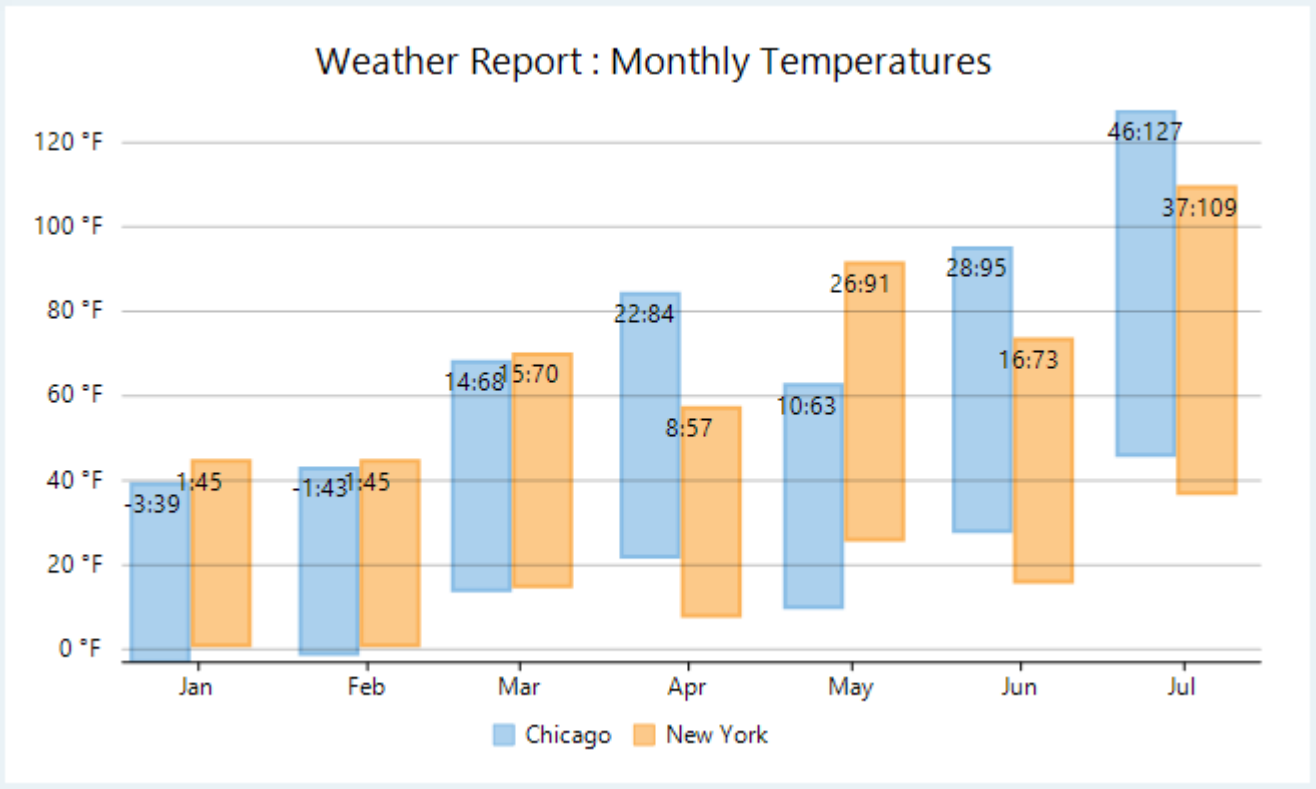
Gradient Legend with Categories



CS VB

フローティングバー

Floating bar charts are charts with a single or multiple bars floating between a minimum and maximum value instead of being connected to the axis. It displays information as a range of data by plotting two Y-values(low and high) per data point. Floating bars can be useful to show highs and lows in a data set, such as daily high and low temperatures, stock prices, blood pressure readings, etc.



In FlexChart, floating bar chart can be implemented using the Series class. To begin with, create a new **Series** object and specify its properties. Then, use the **SymbolRendering** event provided by the Series class to plot the data points on the chart.

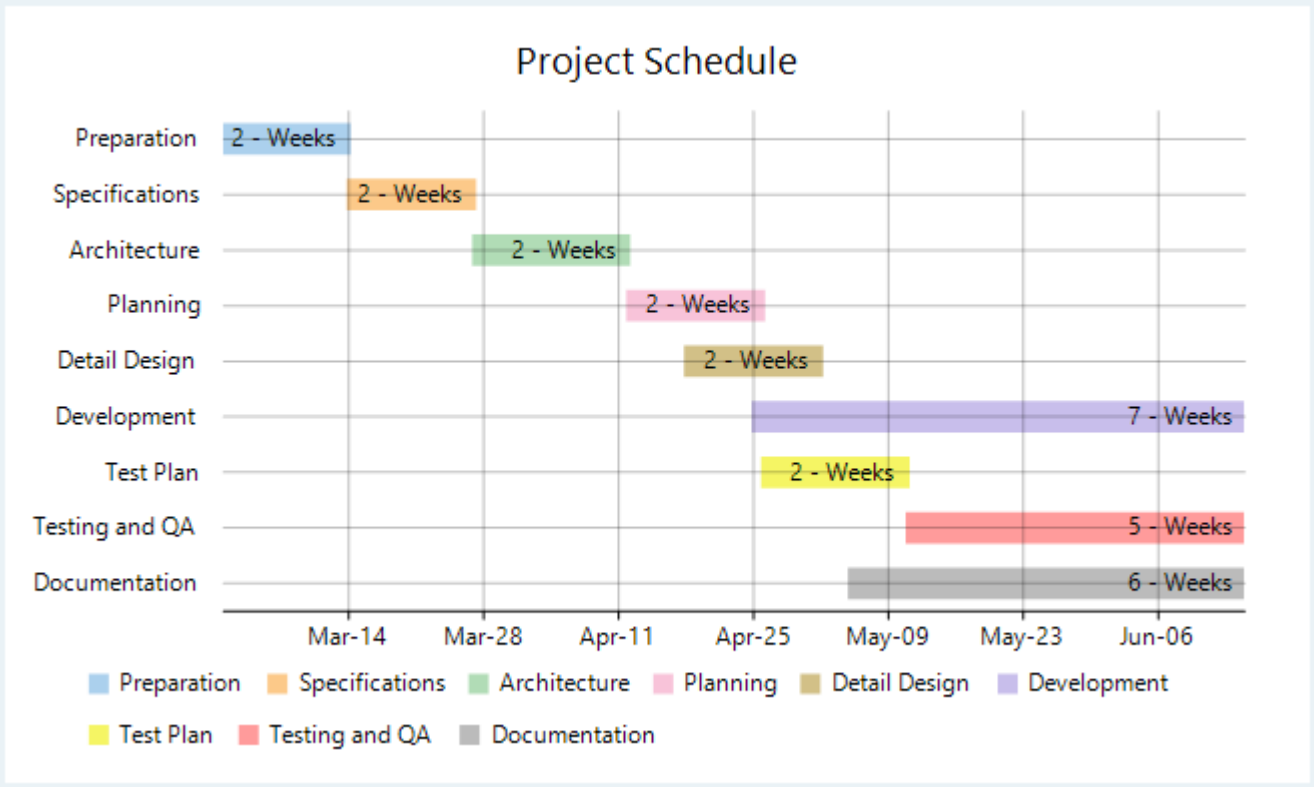
CS VB

Note that the above sample code uses a custom method named GetTemperatureData to supply data to the chart. You can set up the data source as per your requirements.

CS VB

ガントチャート

Gantt charts are the charts which demonstrate the project schedules and plans by plotting the project activities against time. With activities listed on Y-axis and time scale along the X-axis, these charts indicate the duration of each activity through length of horizontal bars positioned to start at the start time of an activity.As primary purpose of a Gantt chart is planning and scheduling, gantt charts are used for a range of projects and in various industries, such as construction, engineering, manufacturing, infrastructure, IT and more.



In FlexChart, gantt chart can be implemented using the Series class. To begin with, create a new **Series** object and specify its properties. Then, use the SymbolRendering event provided by the Series class to plot activity bars on the chart and the LabelRendering event provided by the FlexChart class to display the labels.

CS VB

Note that the above sample code uses a custom method named GetTasksData to supply data to the chart. You can set up the data source as per your requirements.

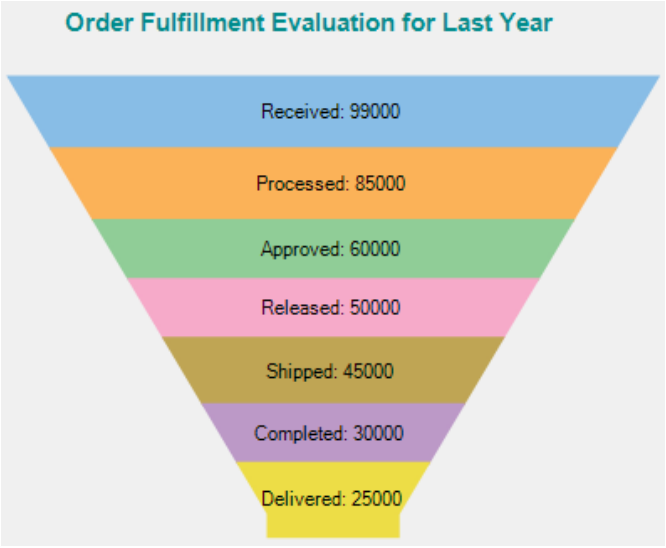
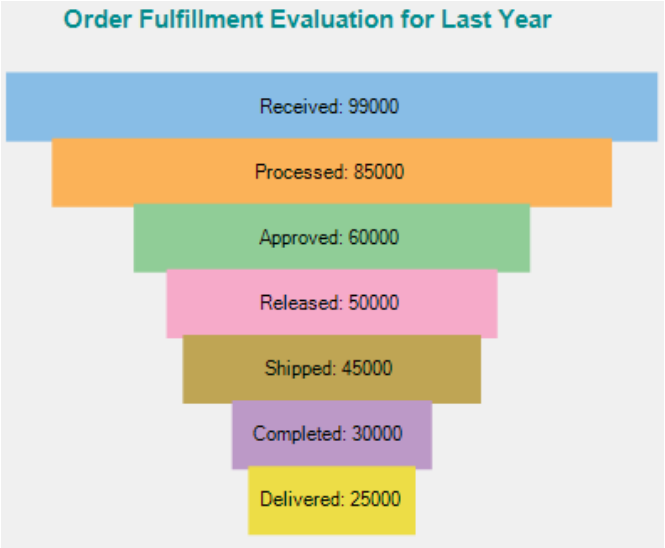
CS VB

ファンネル

Funnel charts are the charts that help in visualizing the sequential stages in a linear process such as order fulfillment. In such processes, each stage represents a proportion (percentage) of the total. Therefore, the chart takes the funnel shape with the first stage being the largest and each following stage smaller than the predecessor. Funnel charts are useful in identifying potential problem areas in processes where it is noticeable at what stages and rate the values decrease. For instance, a an order fulfillment process that tracks number of orders getting across a stage, such as orders received, processed, approved, released, shipped, completed and finally delivered.

FlexChart offers the funnel chart in two forms:

- **Trapezoid chart:** Arranges the related values on top of one another in the form of horizontal sections in a trapezium.
- **Stacked Bar chart:** Arranges related values on top of one another in the form of horizontal bars.

Trapezoid Funnel Chart	StackedBar Funnel Chart
	

Create a Funnel Chart

With FlexChart, you can create a funnel chart by setting the ChartType property of FlexChart class to **Funnel**. This property accepts the values from ChartType enumeration of C1.Chart namespace. To create a Trapezoid or StackedBar type funnel chart, you can set the FunnelType property to **Default** or **Rectangle** respectively.

In case of a trapezoid funnel chart, FlexChart also provides options to set the neck width and neck height. These properties are available in the ChartOptions class accessible through the Options property of the **FlexChart** class.

CS VB

Note that the above sample code uses a custom method named GetRecruitmentData to supply data to the chart. You can set up the data source as per your requirements.

CS VB

レーダーチャート

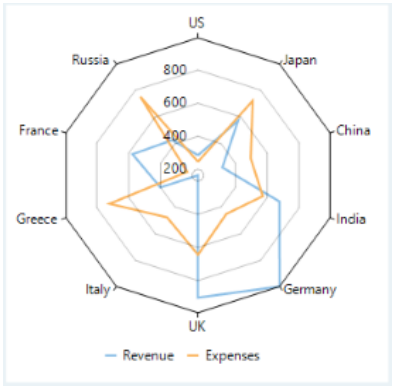
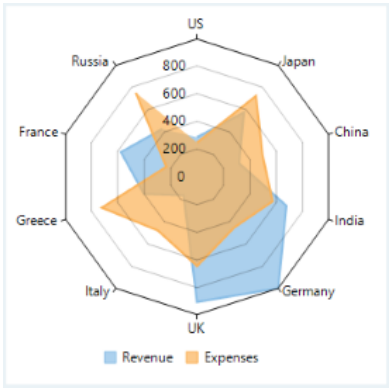
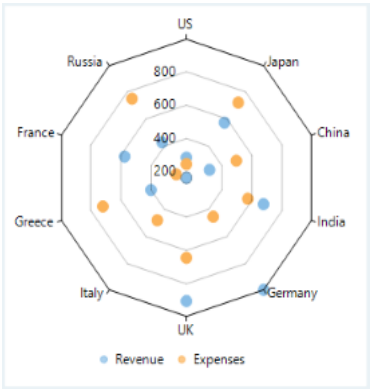
Radar charts are radial charts that help in visualizing comparison of two or more groups of values against various features or characteristics. These charts represent each variable on a separate axis which are arranged radially around a center at equal distances from each other. Each of these axes share the same tick marks and scale. The data for each observation is plotted along these axis and then joined to form a polygon. Multiple observations, that is, polygons plotted on a chart are also a great way to identify the outliers among each variable. Radar charts are generally used for analyzing performance or comparing values such as revenue and expense as shown in the charts below.

Polar charts are another variation of radar charts where X values are numbers that specify angular values in degrees.

Radar Chart	Polar Chart

Create a Radar Chart

FlexChart for WinForms provides radar chart through a stand alone control which is represented by the FlexRadar class. You can bind the chart with data using the DataSource property. You need to supply X and Y values to the chart by setting the Binding and BindingX properties. FlexChart also lets you specify the angle from where you want to start drawing the chart in the clock-wise direction by setting the StartAngle property. To render a radar chart in counter clock-wise direction, you need to set the Reversed property to **true**. FlexChart also provides ChartType property so that you can choose if you want to display a radar chart in a line, line symbol, scatter or an area chart format. This property accepts values from RadarChartType enumeration.

Radar Line Chart	Radar Area Chart	Radar Scatter Chart
		

To create a radar chart using FlexChart:

At design-time

1. Drag and drop the **FlexRadar** control to the form.
2. Right click the FlexRadar control on form to open the **Properties** window.
3. Set the data source using the DataSource property.
4. Configure the chart by setting the BindingX and Binding property respectively.

Using code

To create a radar chart through code, the first step after initializing the control is to clear the default series. Set up the data source through the **DataSource** property and configure the X axis (radial axis) by setting the **BindingX** property. Then, add new series using the **Add** method and set the **Binding** property for each added series.

CS VB

```
```

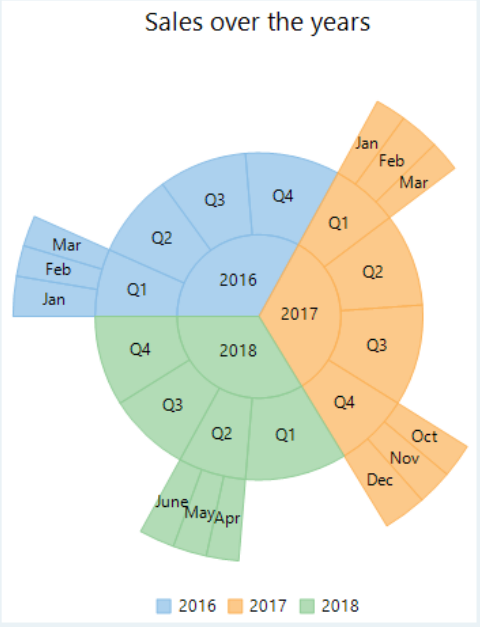
Note that the above sample code uses a custom method named GetData to supply data to the chart. You can set up the data source as per your requirements.

CS VB

```
```

サンバースト

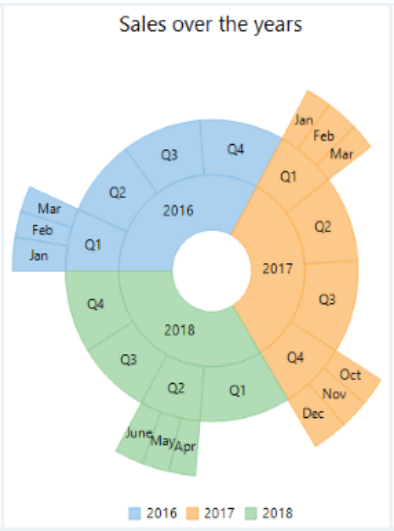
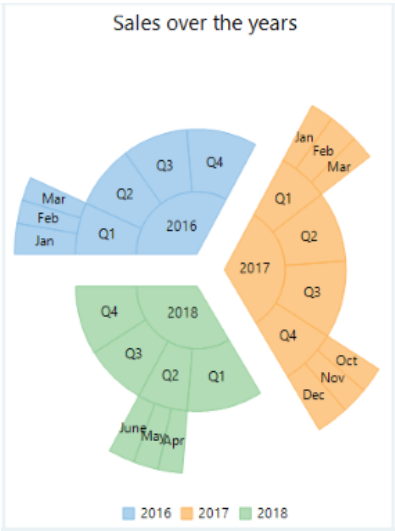
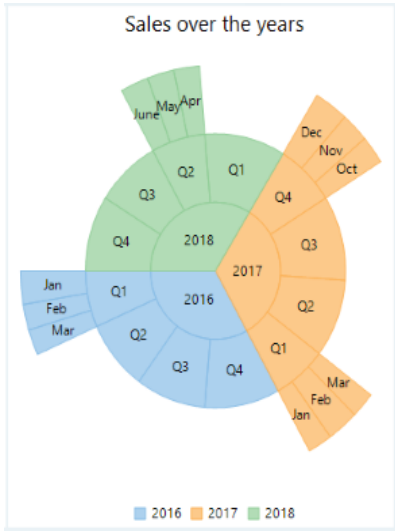
Sunburst, also known as a multi-level pie chart, is ideal for visualizing multi-level hierarchical data depicted by concentric circles. The circle in the center represents the root node surrounded by the rings representing different levels of hierarchy. Rings are divided based on their relationship with the parent slice with each of them either divided equally or proportional to a value. For instance, you can use sunburst to display sales over past few years or to display makes or models of a product.



Create a Sunburst Chart

FlexChart for WinForms provides sunburst chart through a stand alone Sunburst control which is represented by the Sunburst class. The control appears as a pie chart when dropped on the form until it is provided with a hierarchical data using the DataSource property provided by the FlexPie class. This class also provides Binding and BindingName properties for setting numeric values and labels of the sunburst slices. You can also specify the angle from where you want to start drawing the slices in the clock wise direction by setting the StartAngle property. FlexChart also provides properties to create following variations of sunburst chart:

- **Doughnut Sunburst Chart:** Set the InnerRadius property to a value greater than zero to create a hole in the center. By default, this property is set to **zero**.
- **Exploded Sunburst Chart:** Set the **Offset** property to a value greater than zero to push the slices away from the center. By default, this property is set to **zero**.
- **Reversed Sunburst Chart:** Set the **Reversed** property to **true** that creates the chart with angles drawn in the counter clockwise direction. By default, this property is set to **false**.

Doughnut Sunburst Chart	Exploded Sunburst Chart	Reversed Sunburst Chart
		

To create a sunburst chart using FlexChart:

At design-time

1. Drag and drop the **Sunburst** control to the form.
2. Right click the Sunburst control on form to open the **Properties** window.
3. Set the data source using the **DataSource** property.
4. Configure the chart by setting the **Binding** and **BindingName** property.
5. Set the **ChildItemsPath** property to name of the property that contains child items.

Using code

To create a sunburst chart through code, the first step after initializing the control is to clear the default series. Set up the data source through the **DataSource** property and configure the chart by setting the **Binding** and **BindingName** property. Also, set the **ChildItemsPath** property to generate child items in hierarchical data.

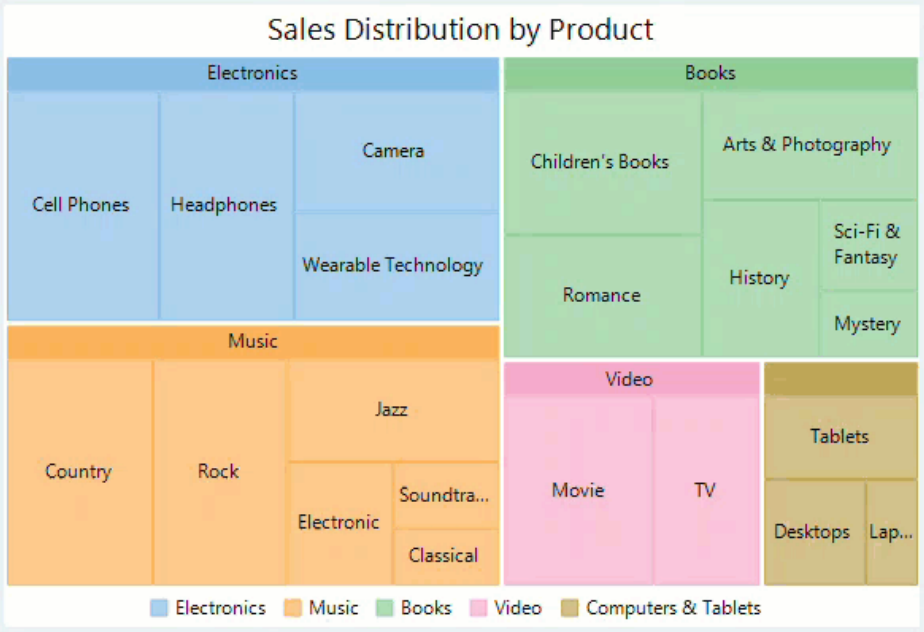
CS VB

Note that the above sample code uses a custom method named GetSunburstData to supply data to the chart. You can set up the data source as per your requirements.

CS VB

ツリーマップ

TreeMaps are the data visualization tools that display the hierarchical data as a set of nested rectangles, while displaying the quantities for each category through area size of the corresponding rectangles. These charts are useful in giving a quick glimpse of patterns in huge hierarchical data sets without costing you much of screen real estate. For instance, below treemap shows the product-wise sales distribution across various categories without occupying much space.

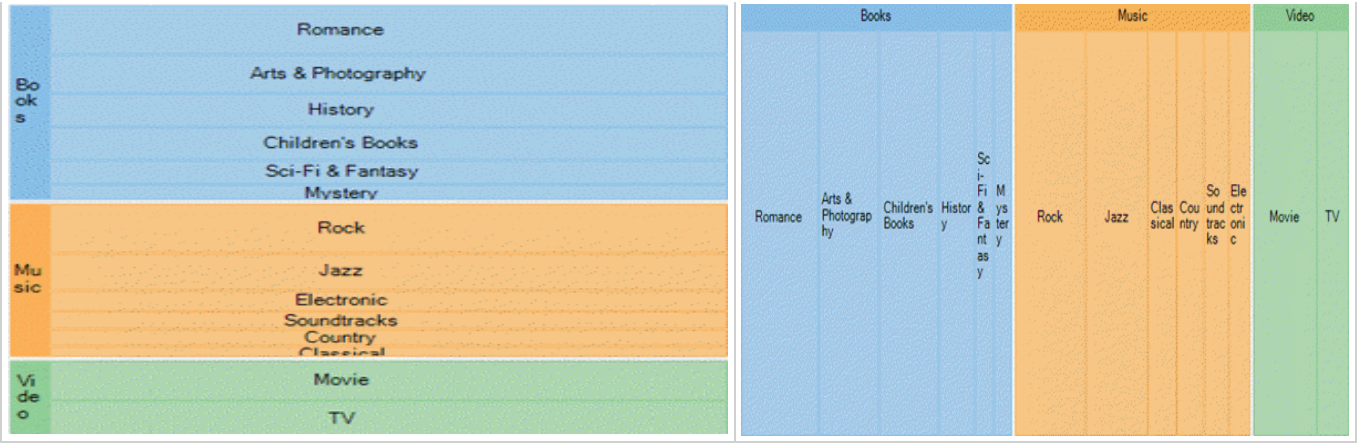


Create a TreeMap

FlexChart for WinForms provides treemap through a stand alone control which is represented by the TreeMap class. You can bind the chart with data using the DataSource property provided by the **TreeMap** class. This class also provides Binding and BindingName properties for generating rectangular nodes for data items and their respective categories or groups. While **Binding** property takes string value depicting the name of the property of data item that contains numeric data value, helpful in calculating the size of rectangular nodes, **BindingName** takes string value depicting the name of data items. **ChildItemPath** property ensures that a hierarchical structure of the provided data collection is maintained, by communicating to the control about the child items within the data.

By default, the TreeMap control displays the squarified layout (as shown in the image above) in which treemap rectangles are arranged as approximate squares. This layout is very useful for displaying large data sets and makes it easy to make comparisons and point patterns. However, you can also display your treemap in the horizontal or vertical layout by setting the **ChartType** property of **TreeMap** class.

Horizontal TreeMap	Vertical TreeMap



To create a treemap using FlexChart:

At design-time

1. Drag and drop the **TreeMap** control to the form.
2. Right click the TreeMap control on form to open the **Properties** window.
3. Set the data source using the **DataSource** property.
4. Set up chart by setting the **Binding** and **BindingName** property.
5. Set the **ChildItemsPath** property to name of the property that contains child items.

Using code

To create a treemap chart through code, the first step after initializing the control is to clear the default series. Set up the data source through the **DataSource** property and configure the chart by setting the **Binding** and **BindingName** property. Also, set the **ChildItemsPath** property to generate child items in hierarchical data.

CS VB

Note that the above sample code uses a custom method named GetTreeMapData to supply data to the chart. You can set up the data source as per your requirements.

CS VB

損益分岐点チャート

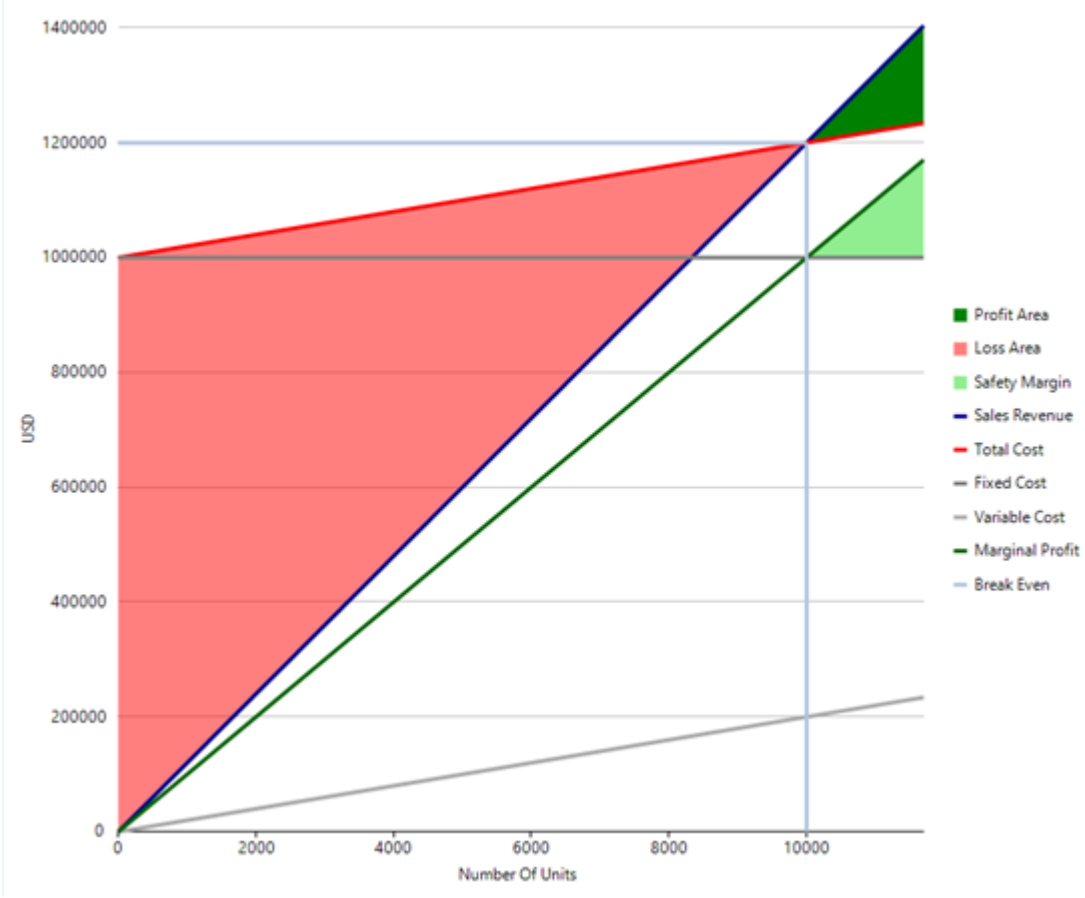
A break-even chart presents the relationship between cost and revenue to indicate profit and loss on different quantities with break-even point. It is also known as Cost Volume Profit graph.

Break-even point i.e., intersection point of the chart, shows the level of sales wherein total revenue is equal to the total costs and net income is equal to zero. Any number below the break-even point constitutes a loss while any number above it shows a profit. The chart plots revenue, fixed costs, and variable costs on the vertical axis, and volume on the horizontal axis.

The Break-Even chart can be highly useful in profit forecasting and planning to examine the effect of alternative business management decisions for your company. It allows end-user to see the unit volume sales level needed to achieve break even, based on which the user can be decided whether it is possible to reach this sales level.

The following example shows how you can create a break-even chart using the FlexChart control. In the example, based on the given data, the break-even point is 10,000 units sold. At this point, sales revenue, and total cost, both are

at 1,200,000 and equate each other. Till this point chart shows the loss and beyond this point it shows the profit.



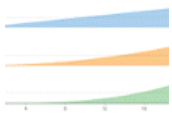
In FlexChart, WinForms break even chart is represented by the **BreakEven** class which inherited from standard Series class. To begin with, you create a new BreakEven object, set values of its main properties (SalesPrice, FixedCost, VariableCost) and add it to the FlexChart.Series collection to show the evolution of profits over time, and how long it takes to reach the break-even point where revenues surpass expenses.

C#

```
flexChart1.Series.Add(new BreakEven() { SalesPrice = 120, FixedCost = 1000000,
VariableCost = 20 });
flexChart1.Legend.Position = Position.Right;
flexChart1.AxisX.Title = "Number Of Units";
flexChart1.AxisY.Title = "USD";
```

FlexChart の要素

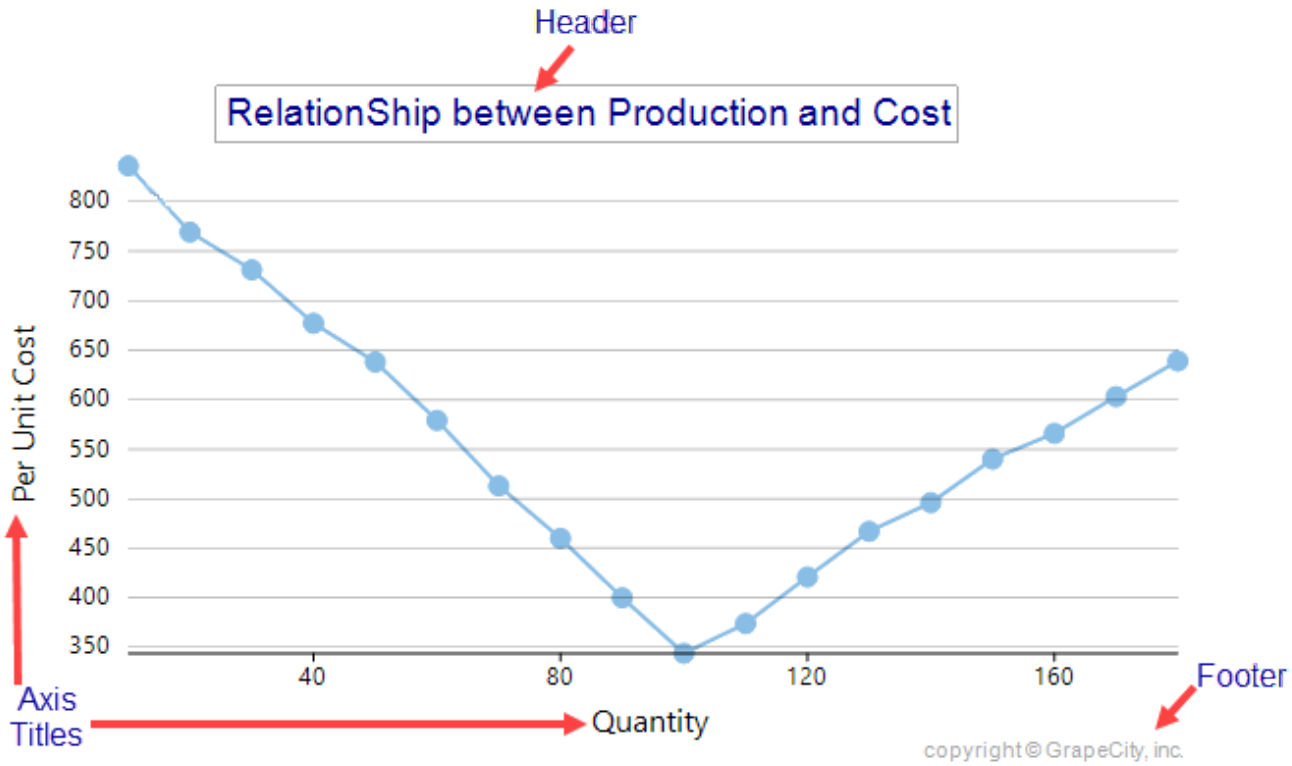
This section discusses the chart elements, their styling and other operations related to the elements.

Topic	Snapshot	Content
Title		Discusses chart titles, axis titles and their styling. • Set Chart Titles • Style Chart Titles • Set Axis Titles • Style Axis Titles
Series		Discusses how to add a series and customize it.
Axes		Discusses axis elements, axis binding, grouping and other axis related operations. • Position an Axis • Show or Hide an Axis Line • Set Axis Bounds • Display Reverse Axis • Display Multiple Axes • Style an Axis • Axis Elements • Axis Labels • Axis Grouping
Data Labels		Discusses data labels, their customization and how to manage the overlapping. • Create Custom DataLabels • Position Data Labels • Style Data Labels • Manage Overlapping Data Labels
Legends		Discusses legends and other related operations. • Toggle Series • Manage Long Legend Text • Style Legend • Legend Grouping
Plot Area		Discusses plot area, styling and rendering multiple plot areas. • Create Multiple Plot Areas
Tooltips		Discusses tooltip customizations and other related operations. • Custom Tooltip • Formatted Tooltip • Shared Tooltip
Line Markers		Discusses line marker and rendering custom content in line markers.
Annotation		Discusses different types of annotations and how to add or style them. • Add an Annotation • Position an Annotation • Style an Annotation

タイトル

Titles are the text fields that appear on a chart and are used to give description about the chart data. There are two type of titles that a chart can have:

- **Chart Titles**
- **Axis Titles**



Set Chart Titles

There are two types of chart titles that can be added to a FlexChart: a Header and a Footer. Headers are generally used for summarizing the chart and tell what a particular chart is all about. On the other hand, Footers are generally used for mentioning the copyright information or source of data used in the chart. You can set the header and footer of a chart by using the **Content** property which can be accessed through **Header** and **Footer** property which are of the type **ChartTitle** class.

CS VB

Style Chart Titles

To style the header and footer of a chart so that they match with chart and rest of the UI of your application, FlexChart provides various styling properties through the **ChartTitle** class. You can style the titles such as setting the font, fill or stroke colors using the **Style** property. You can also change its HorizontalAlignment, as well as set and customize the Border and BorderStyle.

CS VB

Set Axis Titles

Axis titles are the short text descriptions that explain the data displayed along a particular axis. In FlexChart, you can set the title for each axis by setting the **Title** property of Axis class.

CS VB

Style Axis Titles

In FlexChart, you can customize the axis titles by using the TitleStyle property which is of the type ChartStyle class.

CS VB

系列

Series is a set of related data points that are plotted on a chart. By default, FlexChart displays a column chart with dummy data series at design-time. However, you need to provide the control with data to render the chart at runtime.

In FlexChart, a series is represented by the **Series** class. Although, axis and chart type related properties are generally set on the whole chart, FlexChart also provides you AxisX, AxisY, ChartType, DataSource etc. for each series as well. This is helpful in scenarios such as rendering mixed charts, multiple axes etc. The **Series** class also provides the Name property whose text value represents that chart series in the legend. In case of line chart and area chart, you can also handle the null values in data to avoid gaps in plotting the chart series by setting the InterpolateNulls property to **true**.

Add a Series

FlexChart lets you add a series at design-time as well as through code. Follow the steps below to add a series:

At design-time

1. Open **Properties** window to view FlexChart properties.
2. Navigate to the **Series** field and click the **Ellipsis** button next to it.
   Series Collection Editor appears with a pre-added series, **Series 1**.
3. Click the **Add** button to add an additional series.

Using code

To add a series through code, first create a series by creating an instance of the Series class and then, add it to the

FlexChart series collection by using the **Add** method. FlexChart series collection can be accessed through the **FlexChart.Series** property.

CS VB

Add Data to Series

FlexChart provides multiple ways to add data to a chart series. You can opt for fetching all the data to plot a chart from a single data source or, bind a particular series or axis individually with separate data sources. For more information on binding and adding data to a chart series, see Binding.

Hide a Series

FlexChart provides flexibility to hide or display a series on plot area as well as legend through the **Visibility** property. This property accepts values from SeriesVisibility enumeration which lets you show or hide the series completely and also gives you options to display a series in legend or in plot area only.

CS VB

Moreover, FlexChart also provides LegendToggle property that allows the end user to toggle the visibility of series by clicking on the corresponding legend entry.

CS VB

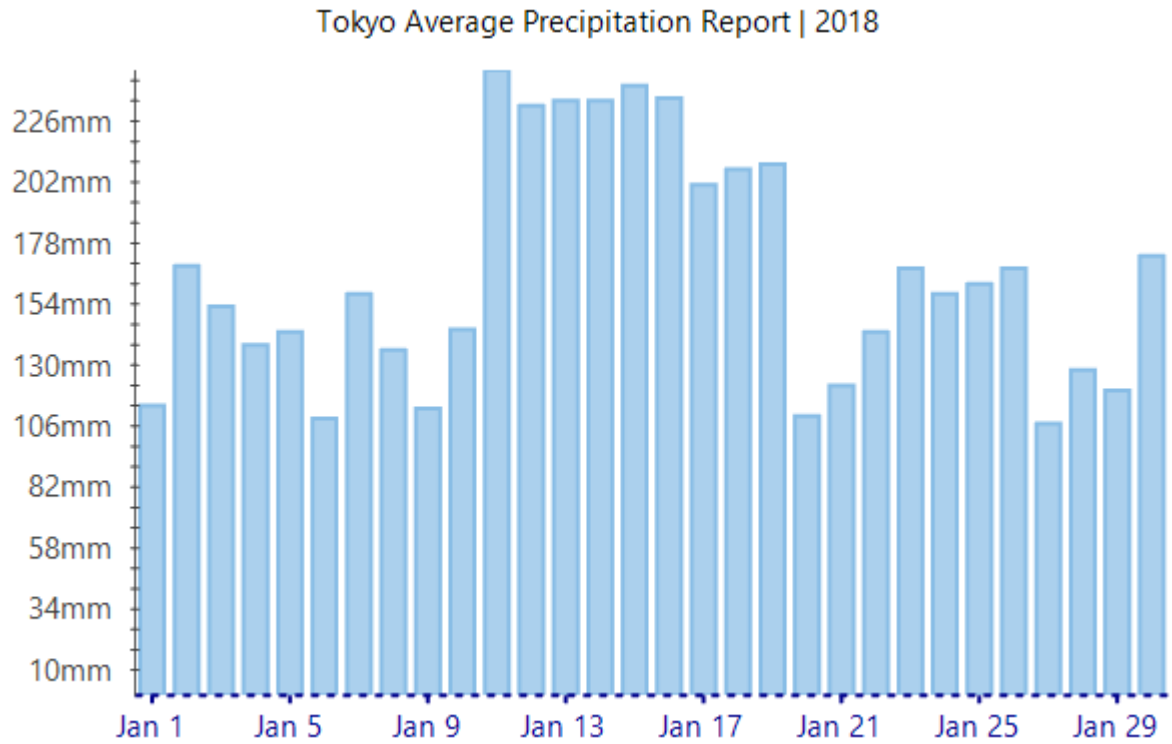
Style a Series

FlexChart provides the Style property of **Series** class to change the appearance of a series. For symbol charts such as scatter, line symbol etc., you can also change the markers, their size and style by setting the SymbolMarker, SymbolSize, SymbolStyle properties. Moreover, chart also provides built-in chart palettes so that you can easily customize the look of your chart by just setting the Palette property. For more information on palettes, see Appearance and Styling.

CS VB

軸

Most of the chart types are plotted between the two axes, horizontal axis and vertical axis commonly known as X and Y-axis respectively. Often X-axis is used to plot categories and Y-axis to plot values, however in some cases such as bar chart, Y-axis is used as a category axis while X-axis is used as a value axis. Similarly, depending on the data, there are cases when both of the axes are used as value axis. Note that charts such as pie chart, sunburst or treemap are some of the exceptions to this and do not possess any axis.



In FlexChart, the two axes are represented by the AxisX and AxisY property which return an object of the **Axis** class. Apart from binding the chart with a data source, FlexChart also lets you bind individual axes to separate data source and fields allowing you to display axis labels different from what is available in the chart data source.

CS VB

Note that the above code snippet uses a custom method GetAxisBindinglabels to supply data for axis binding. For information about axis binding, see Axis Binding.

Position an Axis

By default, FlexChart renders the X-axis at bottom and Y-axis on left side of the chart. However, you can change the position of these axes to top, right, center etc. by setting the **Position** property which accepts the values from **Position** enumeration. Setting this property to **None** hides the axis itself.

CS VB

Show or Hide an Axis Line

By default, FlexChart displays axis line for X-axis but doesn't display the same for Y-axis. To toggle the visibility of axis lines, you need to set the **AxisLine** property of the target axis.

CS VB

Set Axis Bounds

FlexChart also allows you to set the minimum and maximum values that can be plotted on an axis by using the Min and Max properties respectively. This helps you analyze a sub-set of data by plotting target data only instead of cluttering the chart plot area with whole data set.

CS VB

Display Reverse Axis

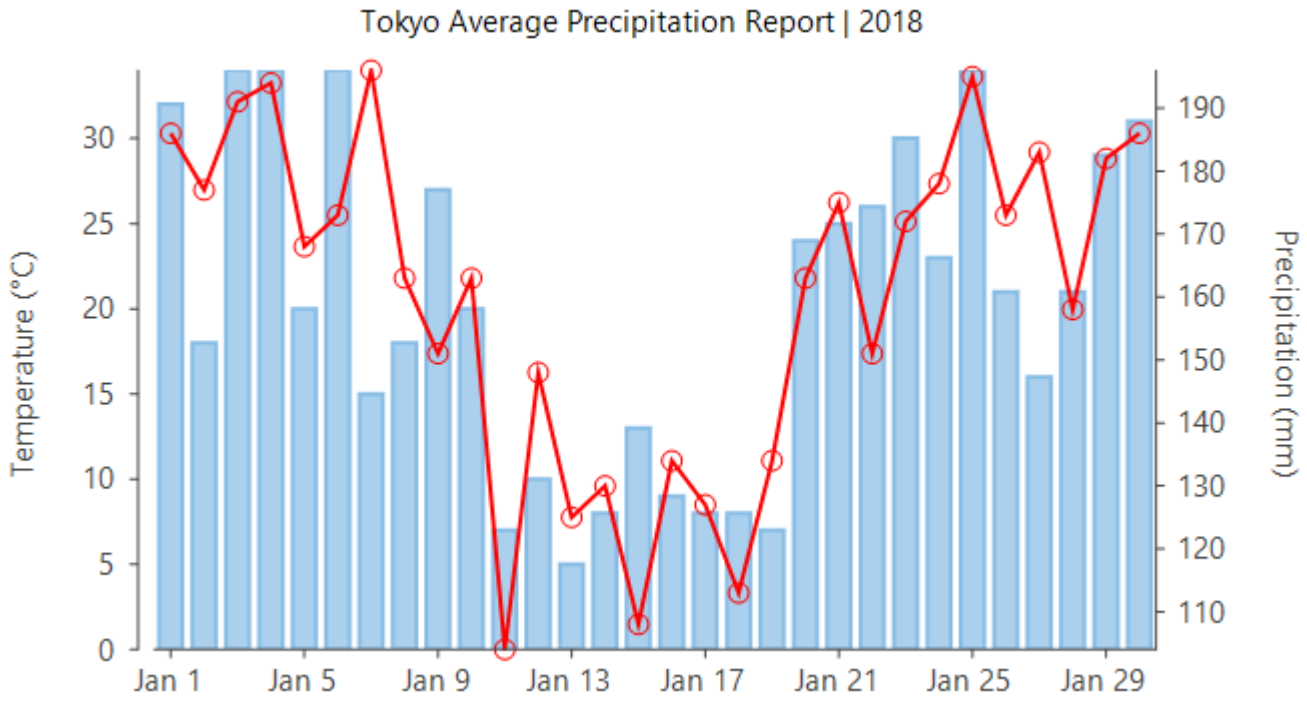
In some cases, flipping an axis to start displaying the values from maximum to minimum helps in better presentation of data such as height of various waterfalls or depth of an ocean at different levels. FlexChart lets you display the reversed axis by setting the **Reversed** property to **True**.

CS VB

Display Multiple Axes

Along with default X and Y-axis, FlexChart allows you to display multiple axes in the same chart. This allows you to

handle multiple series with significantly different range of values. To add a secondary axis to the chart, you need to add a new axis for the series with different data values.



CS VB

Style an Axis

FlexChart allows you to customize the axis lines and make your charts look attractive and in sync with rest of the application UI. The **Axis** class of FlexChart provides Style property that can be accessed to change the stroke color, stroke width, line pattern etc. of an axis.

CS VB

Apart from the features mentioned above, FlexChart also provides you options to customize axis elements such as titles, tick marks, units etc. and axis labels. You can also group the axis for better analysis and presentation needs. Following sections discuss more about these features related to axis and axis elements.

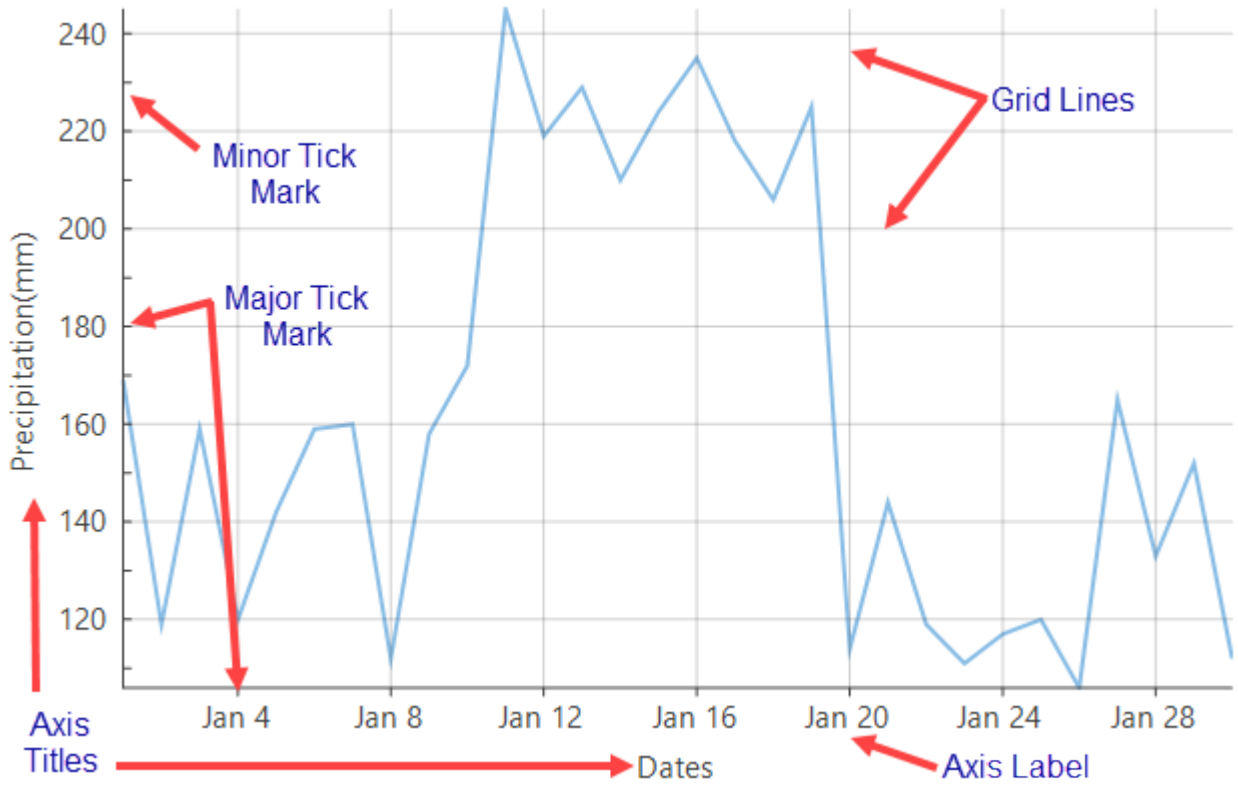
Axis Elements
: Discusses about axis elements and their customization.

Axis Labels
: Discusses axis labels, their customization and options to handle their overlapping.

Axis Grouping
Discusses various types of axis grouping: categorical, numerical and DateTime axis grouping

軸の要素

A chart axis constitutes of various elements such as axis title, tick marks, grid lines, axis units etc. FlexChart provides a number of properties to deal with and customize these elements, so that you can display your data in the most effective manner.



Axis Title

Axis title is the text that appears alongside each axis and summarizes the data shown on the same. In FlexChart, you can set the axis title by accessing the Title property of Axis class. FlexChart also lets you customize the title using the TitleStyle property.

CS VB

Axis Scale

Axis scale is an essential factor in determining how do you want to present your data and your audience to interpret it. Although FlexChart automatically creates an appropriate scale based on the data provided to it, you can also modify the scale by setting the Min, Max and MajorUnit properties. Clearly, the concept of axis scale is only applicable to the

value axis and is not valid for a categorical axis.

CS VB

Axis Units

As the scale is defined by FlexChart automatically when data is supplied to it, major and minor axis units of the value axis are also calculated as a part of the process. However, you can change the values of major and minor units by setting the MajorUnit and MinorUnit property of the **Axis** class. In case of a DateTime axis, FlexChart provides you an option to set the time unit as well using the TimeUnit property which lets you choose from day, month, quarter, week, and year options. This property accepts values from the TimeUnits enumeration. So, for setting the major interval on a DateTime axis to 3 months, you need to set the **TimeUnit** property to **Month** and **MajorUnit** property to **3**.

CS VB

Axis Tick Marks

Tick marks are the small marks or reference points to present the intervals created by dividing the axis according to major and minor units on a value axis. In case of category axis, these marks help in identifying the position of category values on the axis. Evidently, there are no minor tick marks on a category axis. By default, FlexChart sets up X-axis with major tick marks appearing outside the plot and Y-axis with no tick marks. However, you can change the position or visibility by setting the MajorTickMarks or MinorTickMarks properties which accept the value from TickMark enumeration. The enumeration lets you set the position of tick marks to appear inside, outside or crossing on the axis. You can also set this property to **None**, so the tick marks do not appear at all. Moreover, length of the tick marks can also be modified using the TickLength property.

CS VB

Axis Grid Lines

Grid lines are the lines that extend from tick marks perpendicular to the axis and facilitate the viewers with cues to know the unlabeled data points. By default, FlexChart renders the grid lines on a Y-axis but not on X-axis of the chart. However, you can choose to display or hide the same by setting the MajorGrid or MinorGrid properties of the **Axis** class. You can also customize the appearance of grid lines by setting the MajorGridStyle and MinorGridStyle properties.

CS VB

Axis Labels

Axis labels are the text referring to major divisions which appear along an axis. For information about axis labels in FlexChart, see Axis Labels.

軸ラベル

Axis labels are the text referring to major divisions which appear along an axis. On a category axis, axis labels display category names, while those on a value axis display values. FlexChart, by default, automatically generates the axis labels for both axes depending on the data and displays or hides them according to the available space along the axis line. However, you can set the chart to display axis labels for the maximum and minimum values always while automatic generation and placement, by setting the LabelMax and LabelMin properties respectively to **True**. You can also choose to hide all the labels of a particular axis by setting the Labels property of Axis class to **False**. FlexChart also lets you position the data labels with respect to the tick marks on the axis by setting the LabelAlignment property. You can change the format of axis labels by setting the Format property.

CS VB

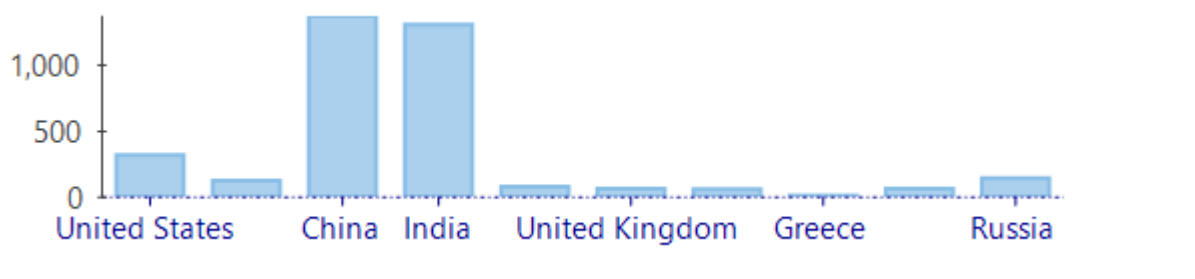
Manage Overlapping Axis Labels

Overlapping of axis labels, generally, occurs due to long axis label text or a large number of data points plotted on a chart. With FlexChart, you get many options to manage your axis labels. You can choose any of them according to the chart data and your requirement.

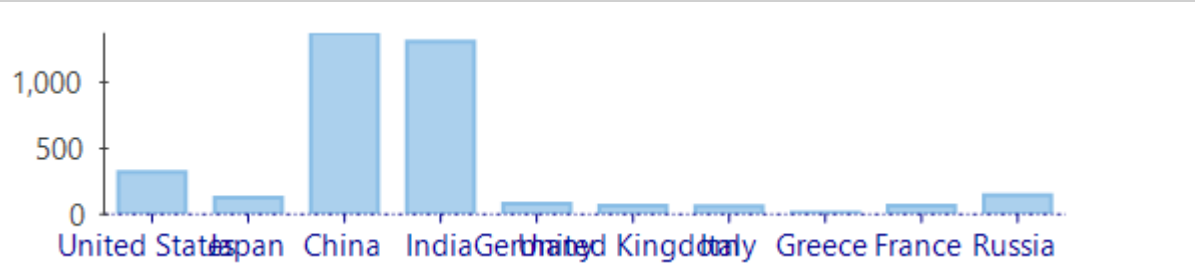
Overlapping label options

As mentioned earlier, by default, FlexChart automatically places the axis labels and hides the overlapping labels if the space does not allow to display them. However, FlexChart provides various options to handle the overlapping labels. The OverlappingLabels property of **Axis** class, which is set to **Auto** by default and is responsible for hiding the overlapping axis labels, also lets you show, trim or wrap the labels in the case of overlapping. This property accepts value from OverlappingLabels enumeration.

OverlappingLabel.Auto

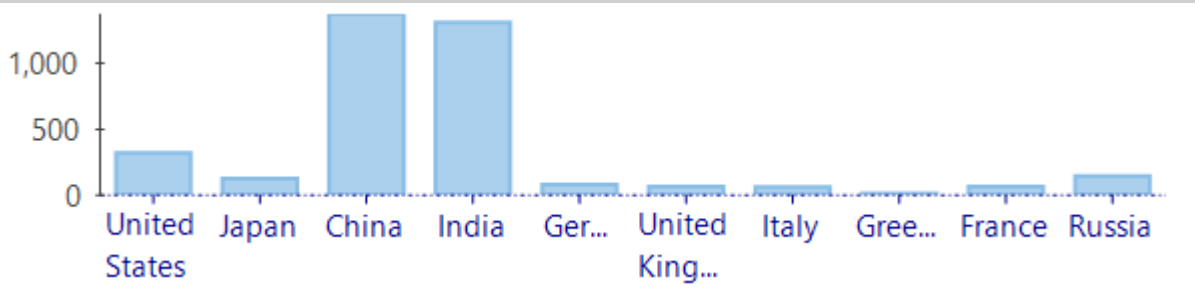


OverlappingLabel.Show



OverlappingLabel.Trim

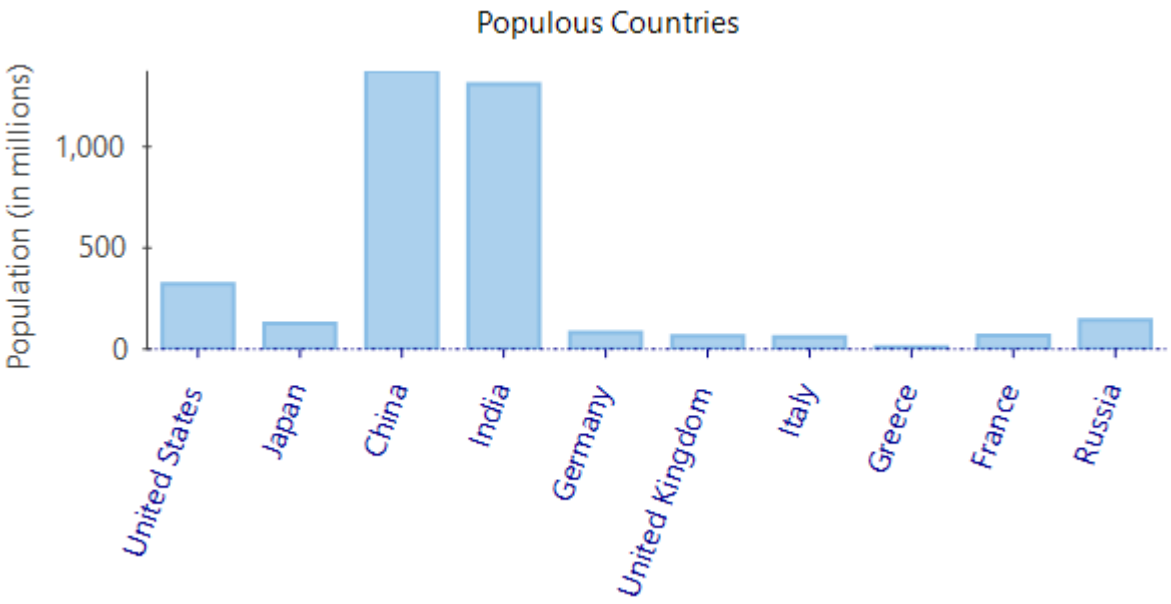
OverlappingLabel.Wrap



CS VB

Rotate axis labels

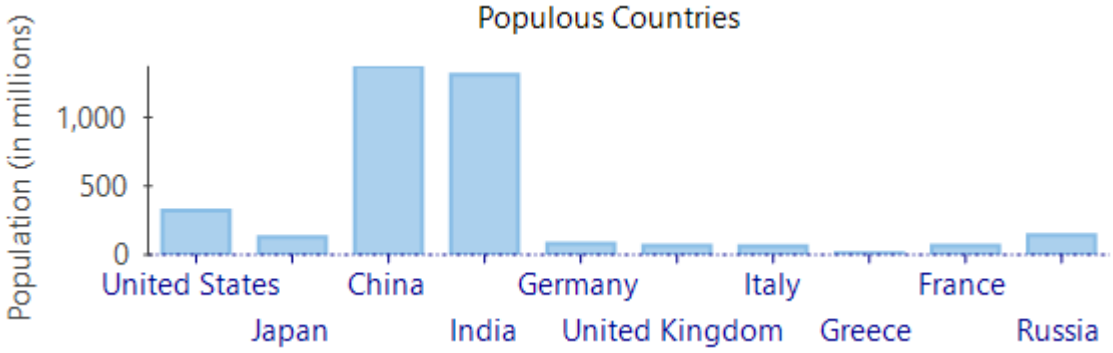
Another option to handle overlapping axis labels could be to rotate them with respect to the axis line by setting the LabelAngle property. This property accepts the numerical values from **-90** to **90** in degrees and rotates the axis labels by the specified angle in anti-clockwise direction, thus giving it a more aesthetic look.



CS VB

Staggered axis labels

Staggering the axis labels is another effective method of managing the overlapping axis labels. This way, you can arrange the axis labels in multiple lines so that they do not overlap and yet be visible. This can be done by setting value of the StaggeredLines property to a value greater than **1**, which is default value of the property.



CS VB

軸グループ

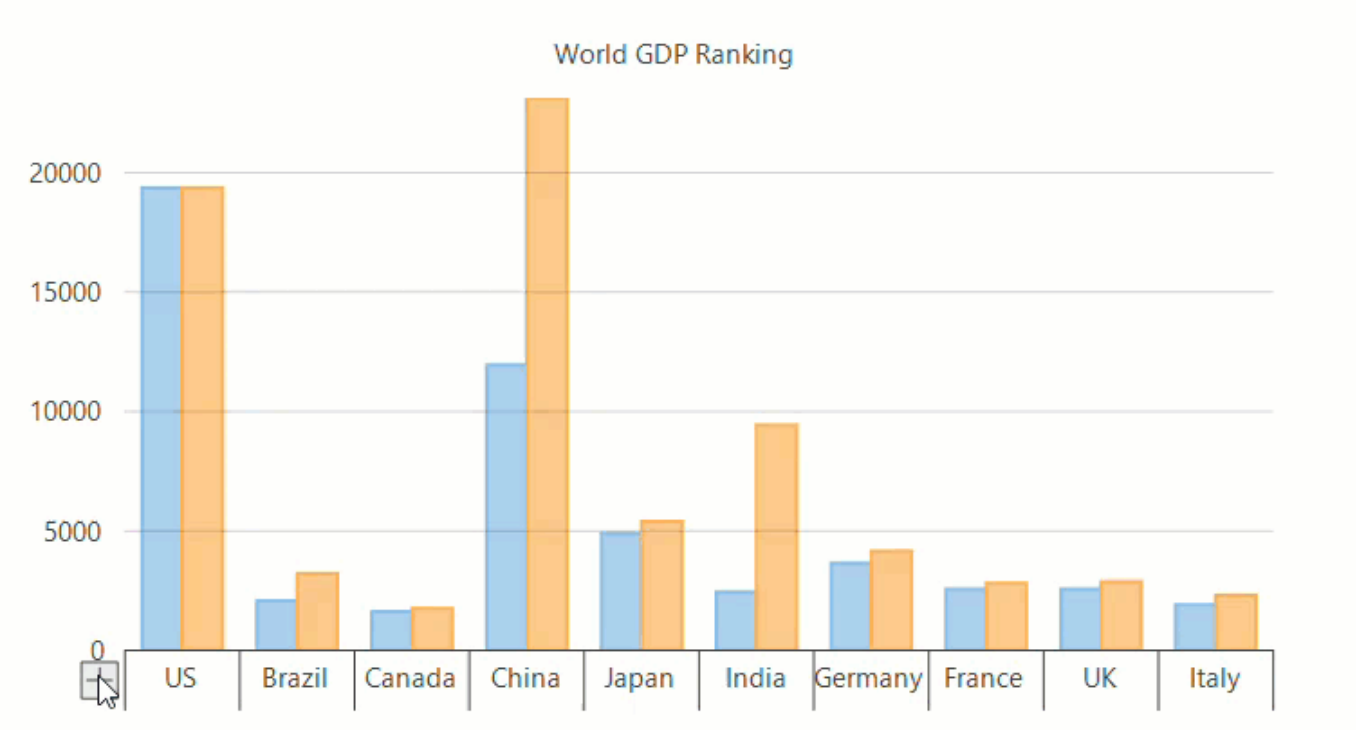
Axis grouping refers to the grouping of axis labels, wherever possible, for better readability and data analysis at different levels. Broadly, there can be three types of data based on which implementation of axis grouping also varies: categorical, numerical and DateTime format.

In **FlexChart**, all these three implementations can be done using the methods discussed in the respective sections below. FlexChart also provides the GroupSeparator property of Axis class that allows you to display the group separator for clear and cleaner division of groups. Moreover, you can allow user to expand or collapse these groups by setting the GroupVisibilityLevel property which accepts an integer value and limits the level of visible collapsible groups. You can also style the groups using the GroupStyle property.

Categorical Axis Grouping

As the name suggests, this type of grouping is done when data is categorical in nature, either flat or hierarchical. For instance, while displaying a country-wise data, you can also group the countries as per the continent and analyze the data of each continent. Another good example could be to analyze the month-wise data which can be grouped into quarters to facilitate the quarterly analysis.

FlexChart provides the GroupNames property to implement grouping on a categorical axis. In case of a hierarchical data, you need to specify the GroupItemsPath property as well to establish the parent-child relationship.

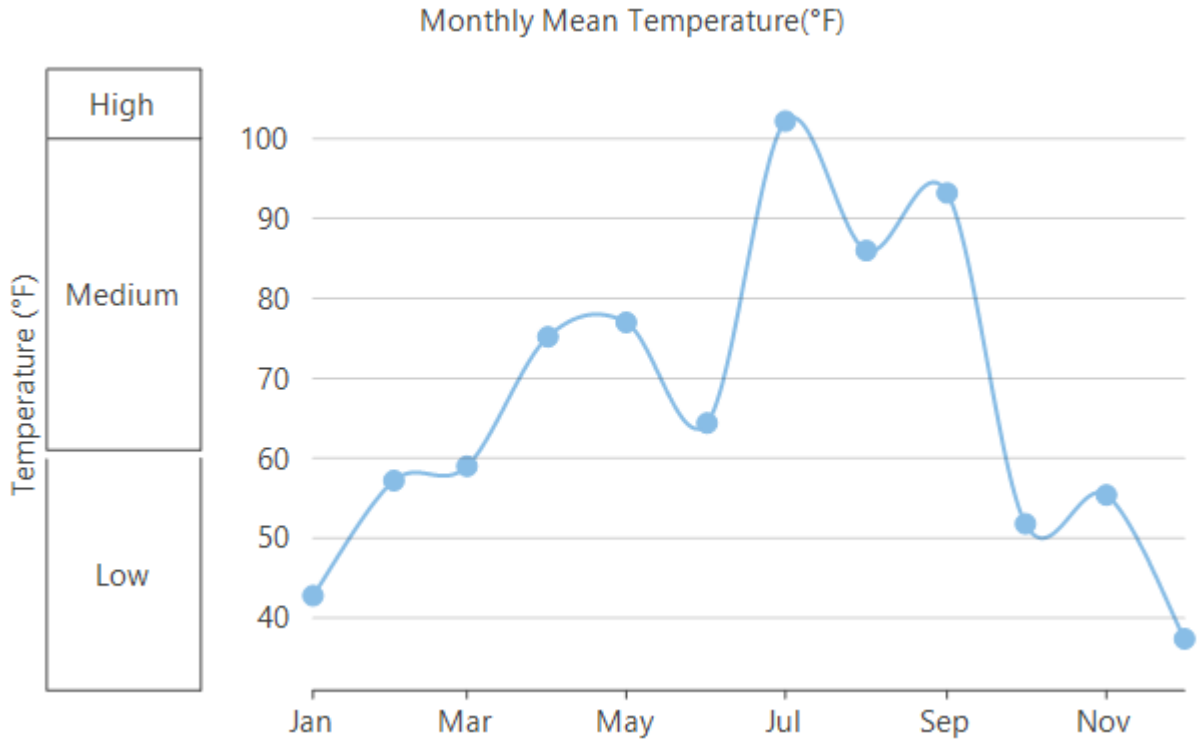


CS VB

Numerical Axis Grouping

Numerical axis grouping is done to group the numerical data into categories or ranges to which user can easily relate. For example, it is more useful to group the temperature data plotted on Y-axis into low, medium and high ranges. To implement grouping on a numeric type axis in FlexChart, you need to create an instance of the IAxisGroupProvider

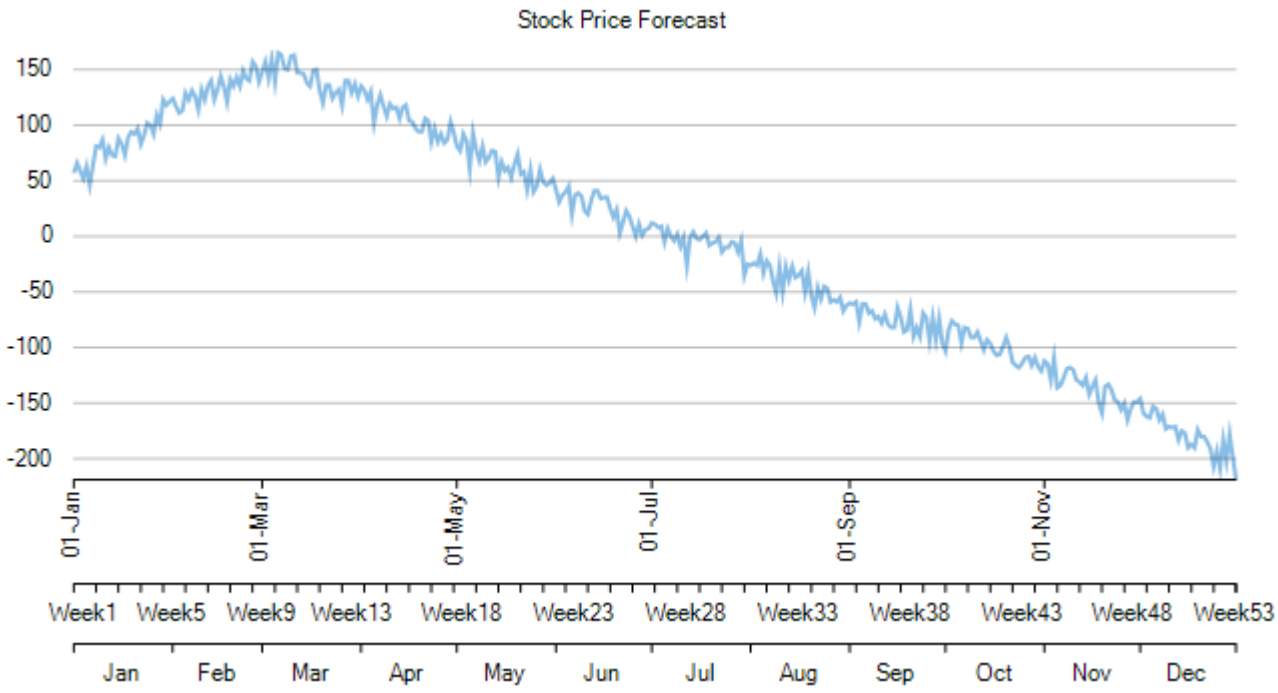
interface and assign it to the GroupProvider property. The interface also provides the GetLevels and GetRanges method which return the number of group levels and a list of range values for a given level respectively.



CS VB

DateTime Axis Grouping

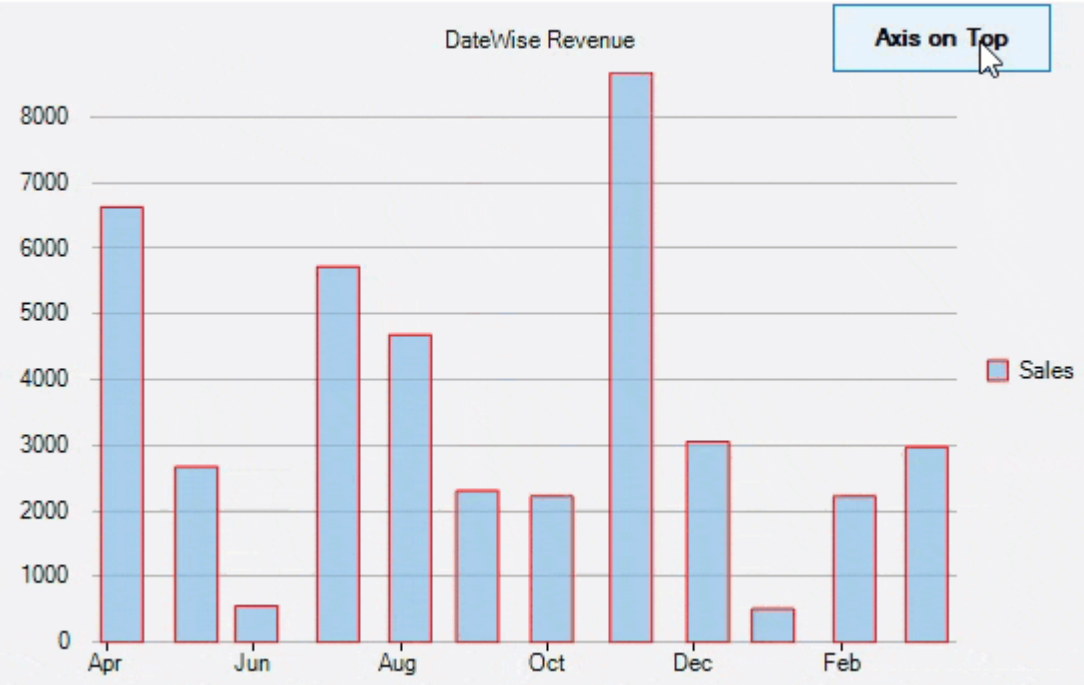
DateTime type data can be grouped into days, weeks, months, quarters or years and hence a DateTime type axis can also be grouped in all these ways. For instance, the example below categorizes the date-wise data plotted on X-axis into weeks and then further into months. For implementing grouping on a DateTime axis in FlexChart, you need to set the **GroupProvider** property to an object of DateTimeGroupProvider class. You also need to specify the type of groups (in the terms of TimeUnits) that you want to add by using the GroupTypes property of this class. The **DateTimeGroupProvider** class also provides the GetLevels and GetRanges method which return the number of group levels and a list of range values for a given level respectively.



CS VB

グリッド線の描画

FlexChart provides you the option to draw gridlines in front of or behind the chart data. For this, the control provides AxesOnTop property of FlexChart class to control the appearance of gridlines.

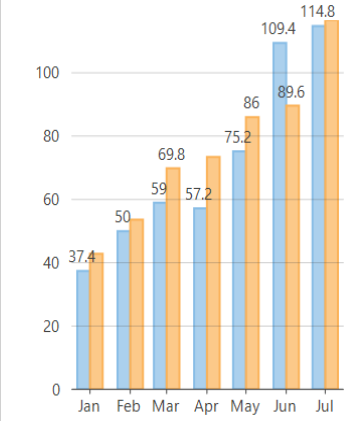
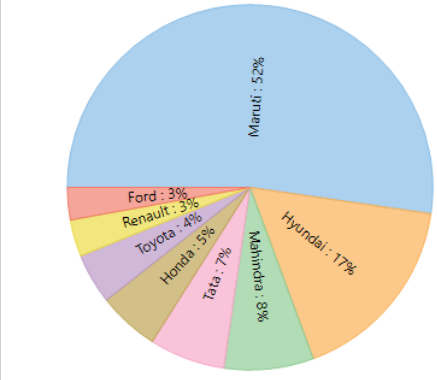
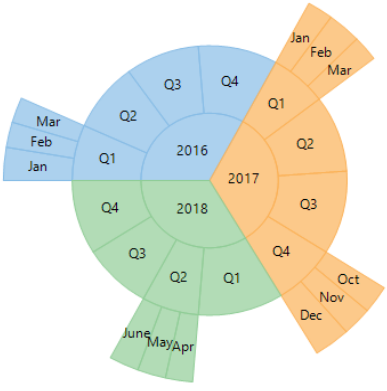


Here, in the above use-case, we have used a button event to draw gridlines on chart at runtime.

CS VB

データラベル

Data labels are the labels associated with the data points, generally, displaying the value of those data points. They are especially useful in the charts where it is difficult to know the exact value of data points by simply looking at them.

Data Labels in Cartesian Charts	Data Labels in Pie Chart	Data Labels in Sunburst Chart
		

In FlexChart, data labels can be displayed by setting the DataLabel property which is of type DataLabel class. You can set the Content property of the **DataLabel** class to specify what to display in the data labels.

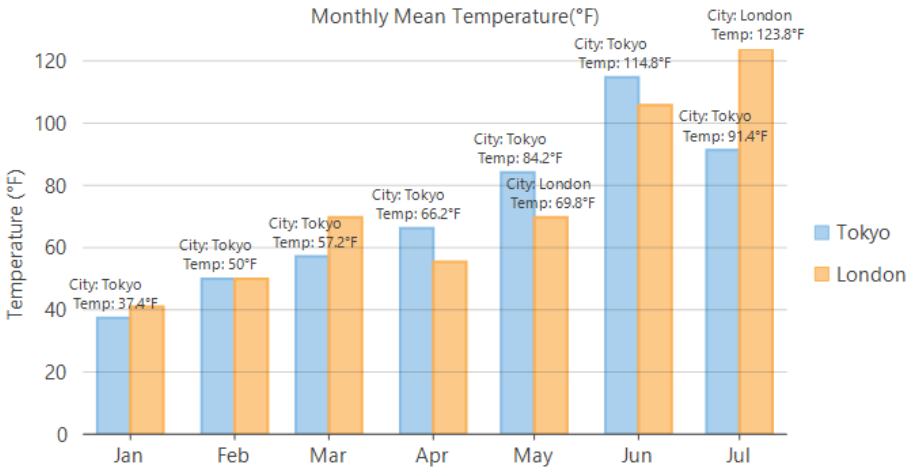
CS VB

Create Custom DataLabels

FlexChart allows you to create custom data labels by assigning a template string to the **Content** property. You can use the following pre-defined parameters to create the template strings.

Parameter	Description
x	Refers to the X value of the data point.
y	Refers to the Y value of the data point.
value	Refers to the Y value of the data point.
name	Refers to the X value of the data point.
seriesName	Refers to the name of the series.
pointIndex	Refers to the index of the data point.
P	Refers to the percentage share with respect to the parent slice in Sunburst.
p	Refers to the percentage share with respect to the whole chart in Sunburst.

FlexChart also allows you to format the values to be displayed in the data label. For instance, you can format the sales or expenses figures by using the required currency, separators or even number of decimal places. You can also use the various date formats, long and short, to display the values of time series.



CS VB

Position Data Labels

FlexChart allows you to specify where to display the data labels with respect to the data point using the **Position** property. This property accepts the values from C1.Chart.LabelPosition enumeration which allows you to position the data labels towards the top, bottom, center, left or right of the data point.

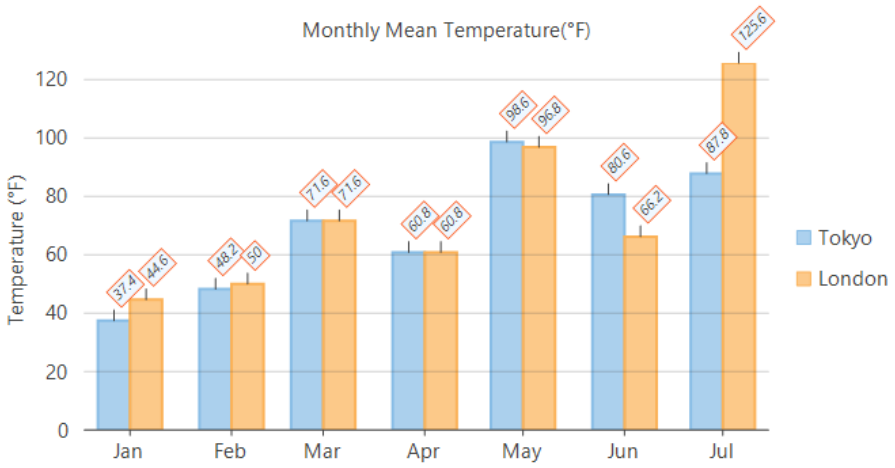
In case of pie chart and sunburst chart, positioning of data labels can be done through the Position property of PieDataLabel class which accepts values from PieLabelPosition enumeration. This property lets you position the data labels towards inside, outside or in the center of the chart. You can also display the data labels in circular or radial direction inside the charts using the same property.

The default value of both of the abovementioned properties is **Auto**, which, automatically positions the data labels according to the available space on the chart. You also have option to set this value to **None** to hide the data labels.

CS VB

Style Data Labels

FlexChart provides various other properties related to data labels which can contribute to the overall appearance and clear understanding of the chart. For instance, to make it clear which data label belongs to a particular data point in a data packed chart, FlexChart allows you to display a connecting line by setting the ConnectingLine property to **true**. You can also set an offset to define how far a data label should appear from the data point. Moreover, FlexChart also lets you display the data labels with borders by setting the Border property to **true**. You can even style the border using the BorderStyle property so that the data labels stand out against the chart background. Another property which helps improve the visibility of the data labels is the Angle property which can be set to a value from **0** to **90** degrees. This also helps in avoiding the overlapping data labels. For more information about managing overlapping data labels, see Manage Overlapping DataLabels.



CS VB

重なったデータラベルの管理

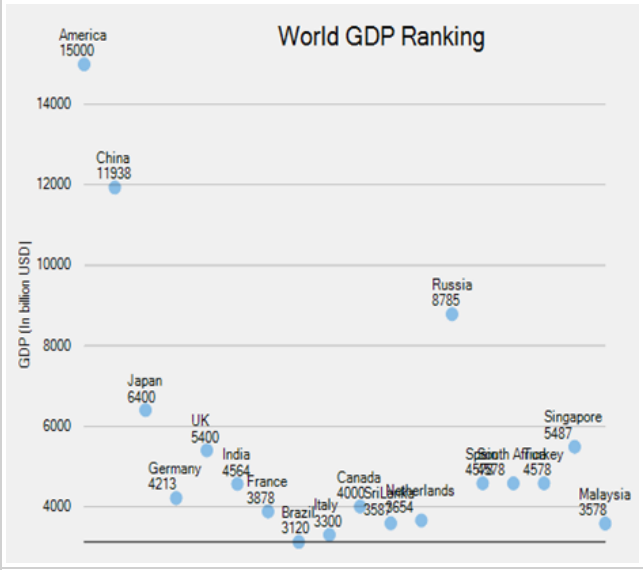
Overlapping of data labels is one of the most comment issues pertaining to charts. Overlapping, generally, occurs due to long data label text or a large number of data points plotted on a chart. With FlexChart, you get many options to manage your data labels. You can choose any of them according to the chart data and your requirement.

- **Auto-arrange data labels**
- **Hide overlapped data labels**
- **Control appearance of overlapped data labels**
- **Rotate data labels**
- **Trim or wrap data label text**

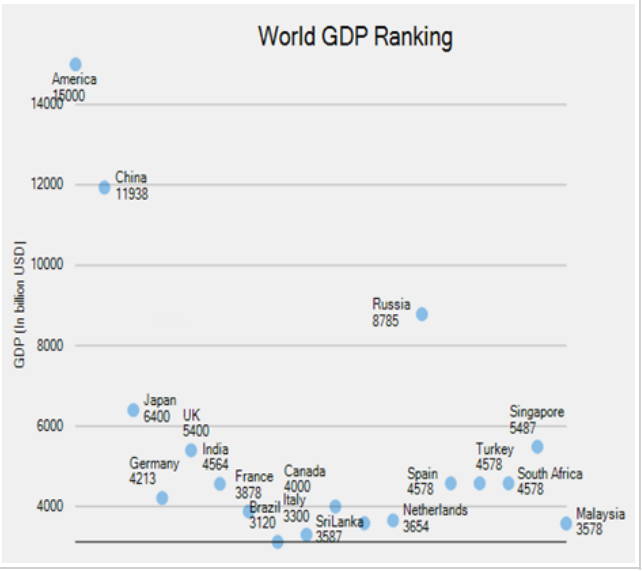
Auto-arrange Data Labels

Allowing FlexChart to arrange the data labels automatically according to the available real estate is the most convenient way of handling overlapping data labels. This can be done by setting the Position property of **DataLabel** class to **Auto**. Not just this, FlexChart also provides **MaxAutoLabels** property so that you can set maximum number of data labels that FlexChart can position automatically. The default value of MaxAutoLabels property is **100** which means FlexChart creates maximum 100 data labels in a chart.

LabelPosition.Top



LabelPosition.Auto

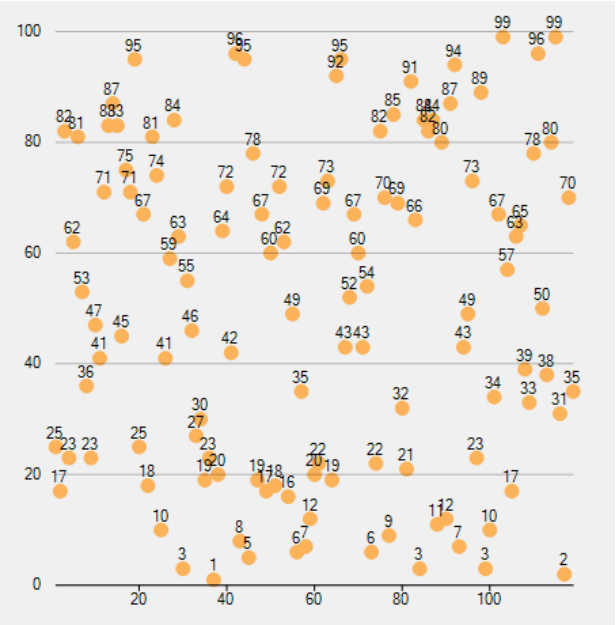
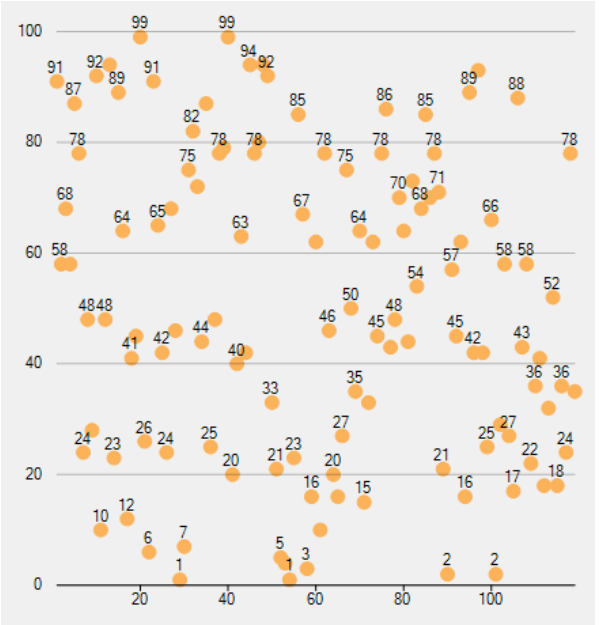


Note that increasing the value of **MaxAutoLabels** property to a value more than 100 may result in slow rendering of FlexChart as the label positioning algorithm becomes expensive in terms of performance when number of data labels is large. Hence, auto-arrangement of data labels may not be an optimal solution when the number of data labels is too large to fit in the available space. In such case, it is recommended to reduce the number of data labels by hiding them at individual series level.

CS VB

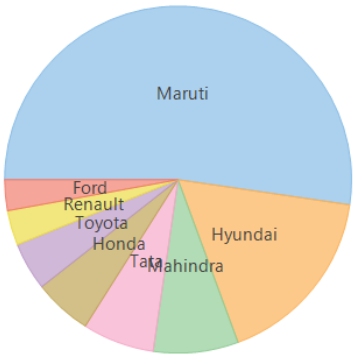
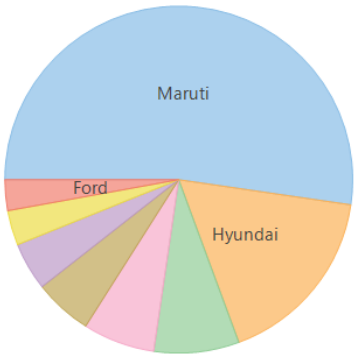
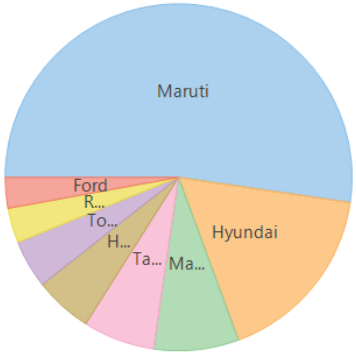
Hide Overlapped Data Labels

FlexChart provides an option to show or hide the overlapped data labels by setting the Overlapping property of **DataLabel** class. This property accepts the values from LabelOverlapping enumeration.

LabelOverlapping.Show	LabelOverlapping.Hide
	

CS VB

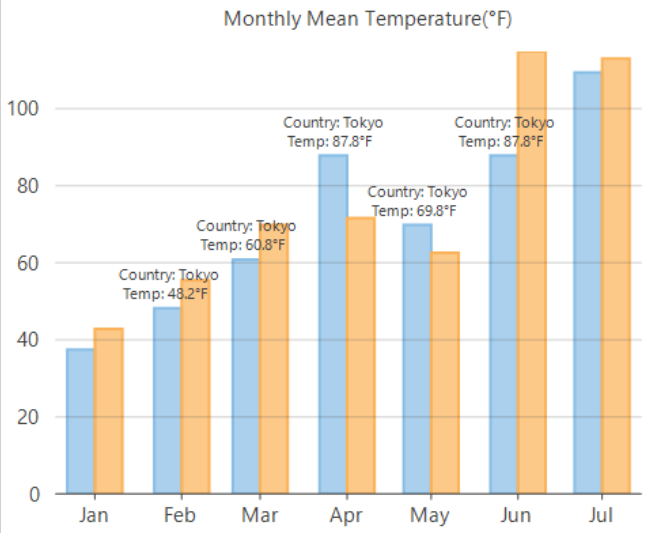
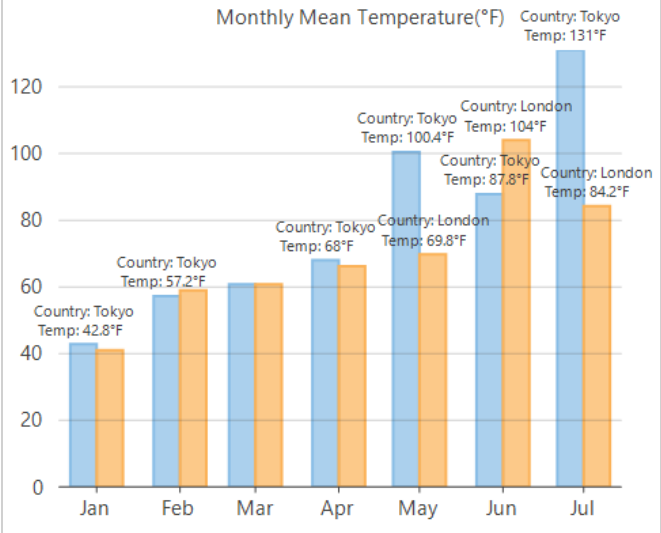
In case of pie chart and sunburst chart, you can use the Overlapping property of the **PieDataLabel** class which accepts the values from **PieLabelOverlapping** enumeration. This property lets you choose to show the data labels, by default, but hides or trims the data labels when label text is larger than the corresponding pie segment.

PieLabelOverlapping.Default	PieLabelOverlapping.Hide	PieLabelOverlapping.Trim
		

CS VB

Control Appearance of Overlapped Data Labels

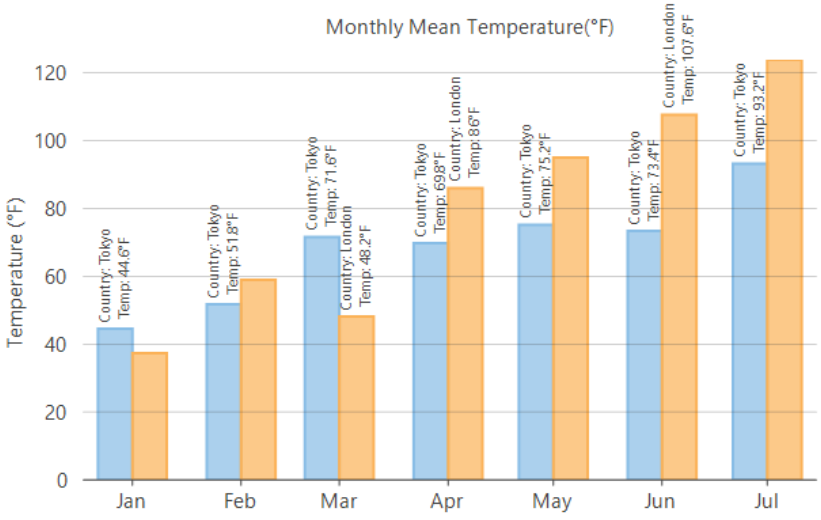
Apart from the options discussed above, you can further refine the settings to manage the overlapping data labels more effectively. You can do this by setting the **OverlappingOptions** property which accepts the values from **LabelOverlappingOptions** enumeration. The property gives you options to avoid overlapping completely, allow data labels to overlap with data points or allow them to display outside plot area.

LabelOverlappingOptions.None	LabelOverlappingOptions.OutsidePlotArea
	

CS VB

Rotate Data Labels

FlexChart allows you to rotate the data labels using the **Angle** property of **DataLabel** class which reduces the chances of overlapping data labels. This property accepts a value between **0** to **90** in degrees.

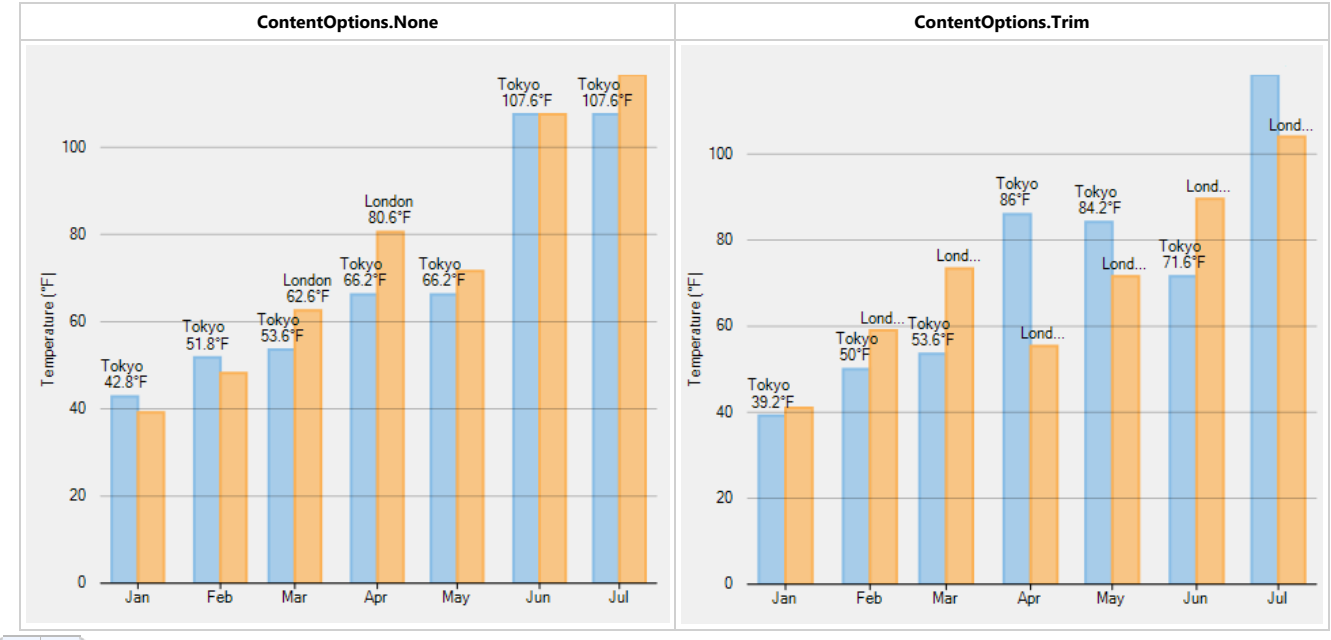


CS VB

Trim Or Wrap Data Labels Text

Another option that FlexChart provides to manage the overlapping of data labels is trimming or wrapping of the data label text if it exceeds a specified character limit or line limit. You can specify whether to trim or wrap the text by setting the ContentOptions property. You can specify the maximum width of data label text after which FlexChart should wrap or trim the text by setting the MaxWidth property. In case of wrapping the text, you can limit the wrapped text to grow vertically by specifying the **maximum number of lines** until which the data labels should be wrapped. To indicate that the further text is trimmed, FlexChart displays an ellipsis at the end of the data label text if it

exceeds the limit defined by the **MaxWidth** and **MaxLines** property.

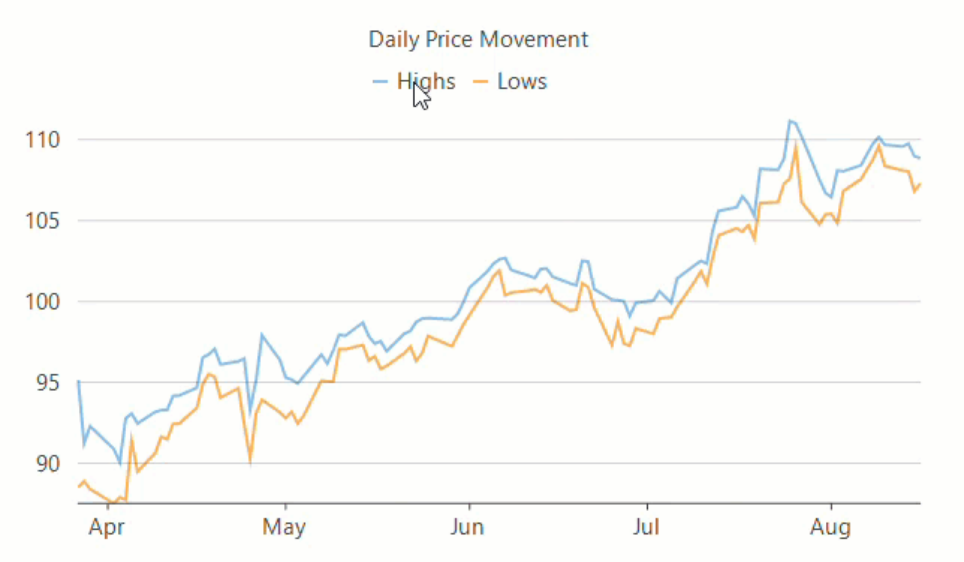


CS VB

凡例

Legend is a chart element which displays a list of colors, symbols and text corresponding to each data series drawn on that chart. It helps in understanding and analysis of the plotted data in the case of multiple series.

In **FlexChart**, a legend is automatically displayed if the Name property of the series is set. In other words, the **Name** property of series is required to generate a legend entry corresponding to the same. Also, FlexChart automatically places the legend according to the space available on the chart area. However, you can also set it to display in top, bottom, left or right with respect to the plot area by setting the Position property. To use the available space in the most optimized manner, FlexChart displays the legend horizontally when placed in top or bottom and vertically when placed in left or right of the plot area. At the same time, FlexChart also allows you to set the Orientation as per requirement.



CS VB

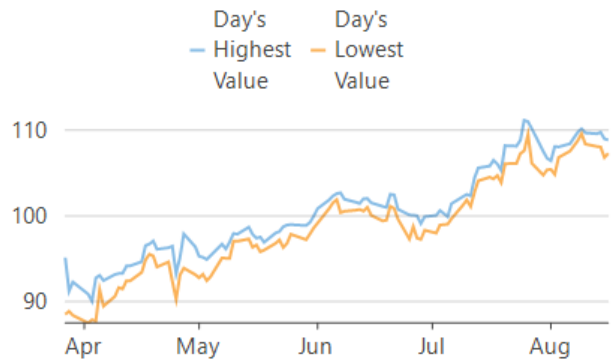
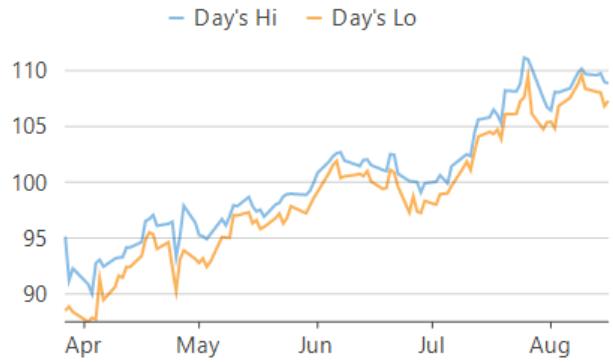
Toggle Series

With FlexChart, user can even toggle the visibility of a data series by clicking on corresponding legend entry at run-time, if the LegendToggle property of FlexChart class is set to **True**.

CS VB

Manage Long Legend Text

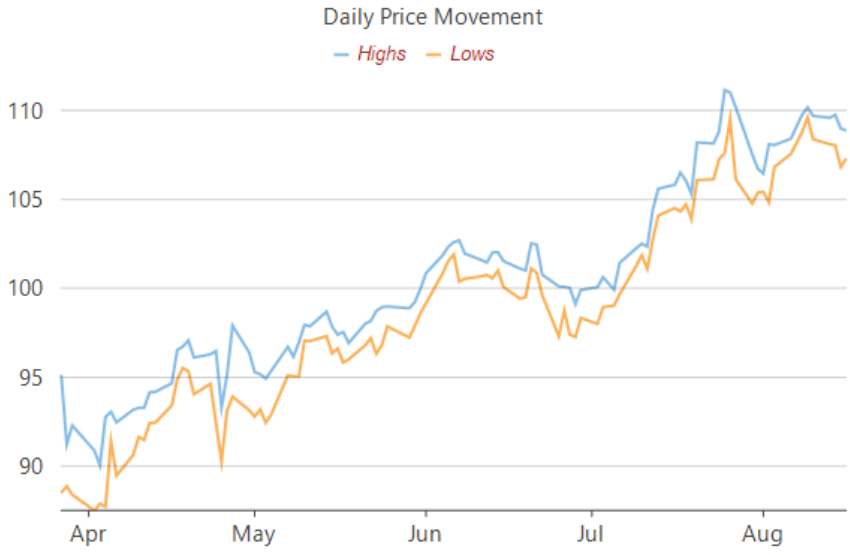
There are times when there is not enough space to display the complete text of legend entries in the chart area. FlexChart provides the TextWrapping property of Legend class that allows you to wrap or trim the text when width of text exceeds the value specified in the ItemMaxWidth property. The **TextWrapping** property accepts values from TextWrapping enumeration.

TextWrapping = TextWrapping.Wrap	TextWrapping = TextWrapping.Truncate
	

CS VB

Style Legend

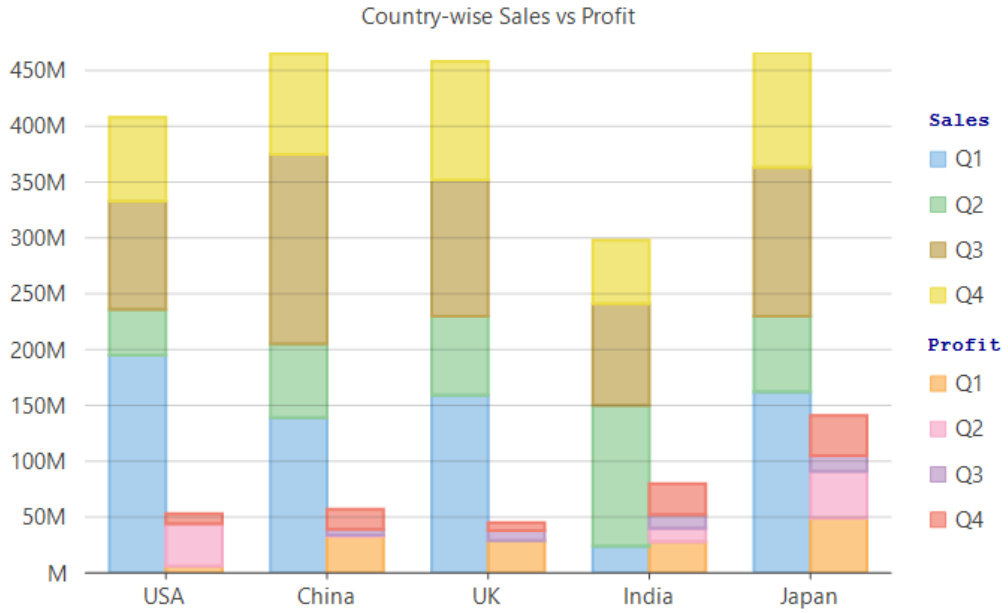
FlexChart also allows you to style the legend and legend entries using the Style property of Legend class. This property lets you specify the fill, fill color, stroke width, stroke color etc. of the legend.



CS VB

Legend Grouping

Legend grouping refers to categorization of legend entries according to the data presented by them. This feature is usually helpful in identifying the categories of data series in the case of multiple stacked series. For instance, while plotting sales and profit of multiple quarters in the same chart area, legend groups prove helpful in identifying which series are the sales series and which ones are the profit series.

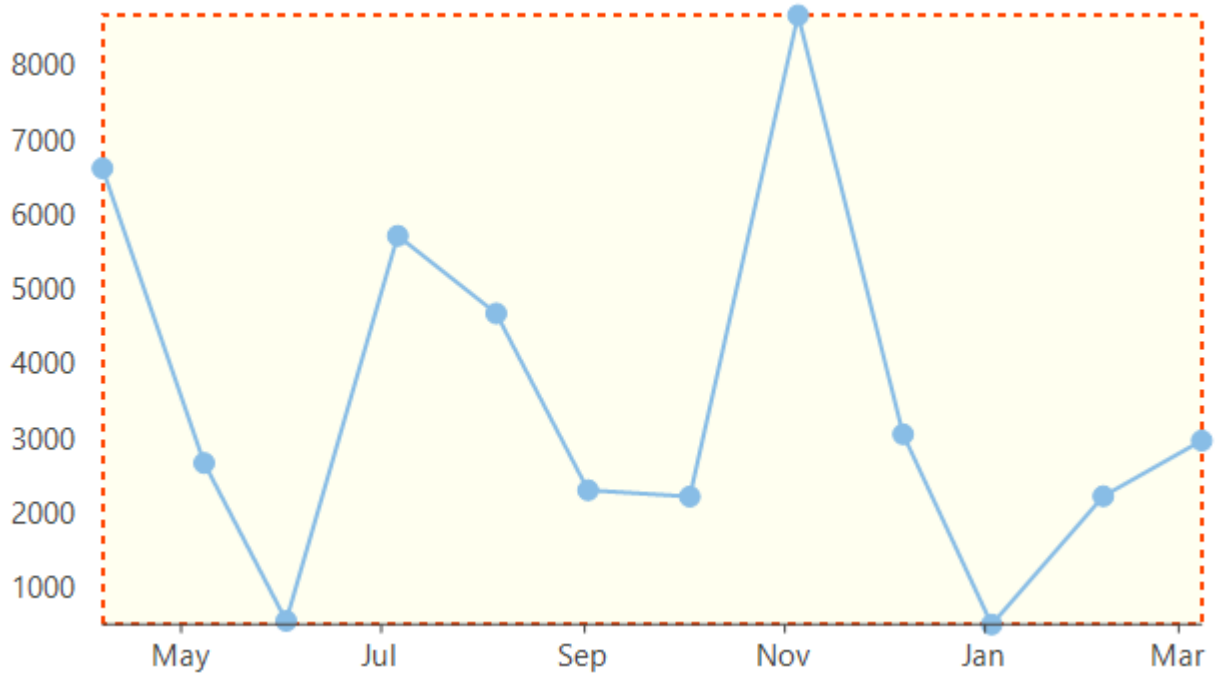


To create legend groups using FlexChart, the control provides the LegendGroup property which accepts a string value and groups the series with same value together. Value of this property not only acts as the group name but the group title as well which gets displayed on the top of the corresponding legend group. The series for which the **LegendGroup** property is not specified is treated as a part of 0th group and is displayed without any group title. FlexChart also lets you customize the legend group titles by using the GroupHeaderStyle property of Legend class.

CS VB

プロット領域

Plot area is the area of chart where data points are plotted. In a chart having X and Y axis, plot area also refers to the area covered by the axes. In a chart like pie chart or sunburst chart, plot area refers to the circular area actually covered by the chart itself.

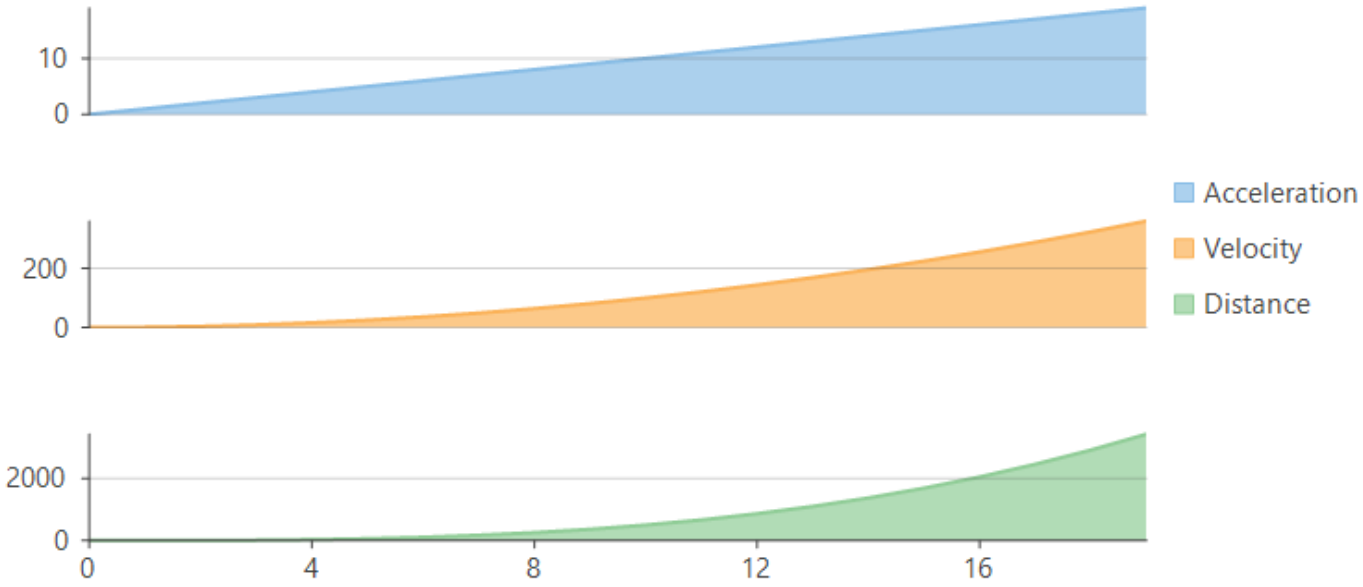


In **FlexChart**, plot area is rendered with a basic white background color. However, you can customize the appearance of plot area to match your requirements using PlotStyle property of ChartStyle type. The **ChartStyle** class provides properties to set fill, fill color, line pattern, stroke, stroke width, stroke dash pattern etc.

CS VB

Create Multiple Plot Areas

Multiple plot areas are advantageous over multiple overlapped series as they increase the readability of data and hence facilitate better analysis. Drawing multiple series, one in each plot area while sharing some of the chart resources like axes, legend etc. is helpful in scenarios like the one shown in the below chart which demonstrates change in acceleration, velocity and distance against time series. In FlexChart, multiple plot areas can be implemented by adding the plot areas to the PlotAreas collection and while defining the series, specify the plot name in which the same has to be rendered. FlexChart also provides properties to set the height, width, row index and column index of each plot area.



CS VB

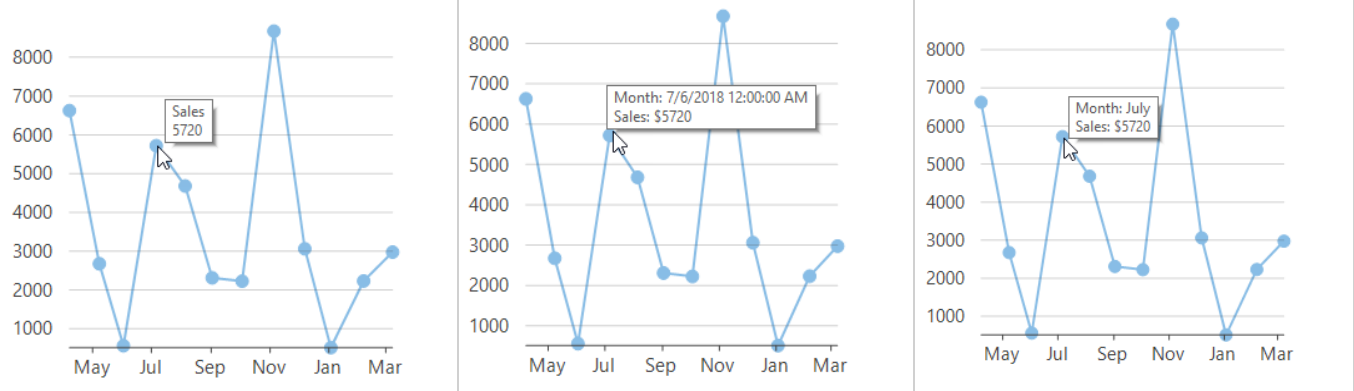
Note that the above sample code uses a custom method named CreateDataPoints to generate random data. You can set up the data source as per your requirements.

CS VB

ツールチップ

Tooltips in a chart are displayed when a user hovers over data points in the chart, thus making the chart more explanatory without creating any visual clutter. By default, FlexChart displays a tooltip showing the series name and value of the data point. However, you can also customize and format these tooltips easily to make them even more informative and clear.

Default tooltip	Custom tooltip	Formatted Tooltip



Custom Tooltip

FlexChart allows you to customize the tooltip by assigning the template strings in the **Content** property. You can use the following predefined parameters to create a template string. For instance, you can display a tooltip with the x-value as well as y-value along with their series name.

Parameter	Description
x	Refers to the X value of data point.
y	Refer to the Y value of data point.
value	Refers to the value of data point.
name	Refers to the name of data point.
seriesName	Refers to the name of series that contains the data point.
pointIndex	Refers to the index of data point.
propertyName	Refers to property of data object represented by the point.

CS VB

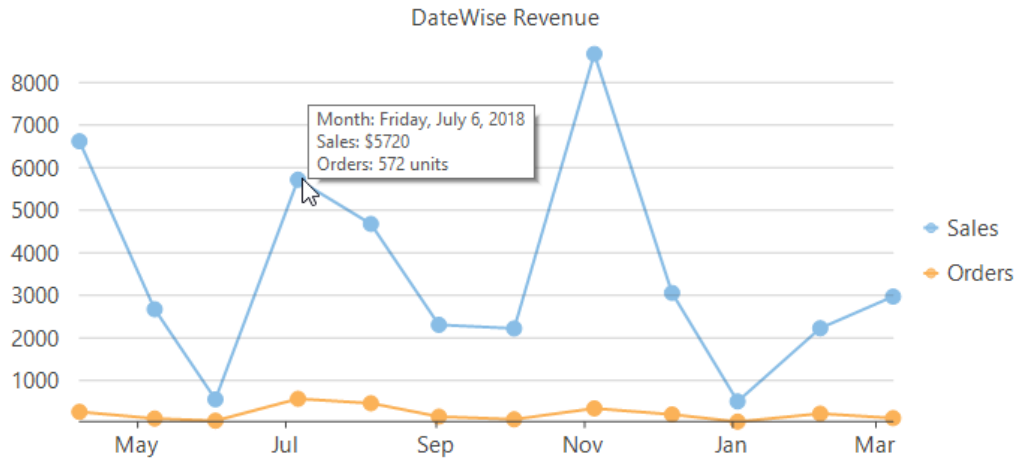
Formatted Tooltip

FlexChart allows you to format the data that is being displayed by specifying the formats in the template string itself which is assigned to the **Content** property. For instance, you can format the sales or expenses figures by using the required currency, separators or even number of decimal places. You can also use the various date formats, long and short, to display the values of time series.

CS VB

Shared Tooltip

Shared tooltip, that is, a single tooltip for multiple series can be used to display the data values of all series against a particular x-value. FlexChart allows you to customize the **Content** property to display the data values of multiple series using the parameters discussed in the **Custom Tooltip** section.



CS VB

ラインマーカー

Line markers are horizontal or vertical lines on the chart plot and are bound to some value on an axis. These are attached with a label to show the exact data values and are generally used for showing a trend or marking an important value or threshold on the chart. Line markers can also be used in case of huge number of data points plotted on a chart or when user wants to display a label with precise data values of multiple series plotted on a chart. For instance, line markers are so useful in a chart that plots price fluctuations of a stock on daily basis across the year.

FlexChart provides line markers through LineMarker class of the C1.Win.Chart.Interaction namespace. You can set the Lines property of this class to specify whether you want to display a horizontal, vertical, both or none(default) of

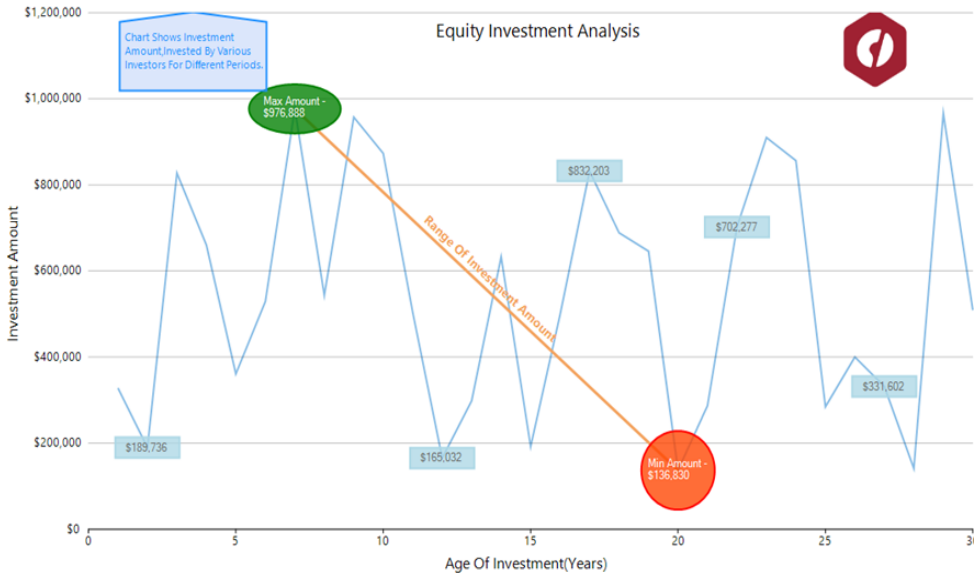
the line markers. This property accepts the value from LineMarkerLines enumeration. **LineMarker** class also provides the Content property to customize the content of line marker label and Alignment property to set the position of the label against the data values.

In FlexChart, by default, the line markers move along with the pointer and facilitate the end user to know the exact data values at the position of the pointer. However, you can change this behavior by setting the Interaction property to **None** or **Drag**. While the value **None** means line marker remains static and does not allow the end-user to interact with the line markers, the value **Drag** lets the end-user drag the line marker to the desired position on the plot area. In the later case, you can specify whether you want to allow dragging of the content by setting the DragContent property. Similarly, DragLines property defines whether the vertical and horizontal lines are linked to each other when one of them is dragged. Moreover, FlexChart also allows you to specify the default position of the line markers when chart is first loaded using the VerticalPosition and HorizontalPosition properties. These properties accept the double type value ranging between **0** to **1**.

CS VB

注釈

Annotations are the user defined objects such as text, images or shapes that can be added to the charts to make them more informative along with enhancing it aesthetically. The extra information in annotations is generally added with the aim to provide better understanding of the chart to the reader.



FlexChart provides following eight types of annotations through C1.Win.Chart.Annotation namespace to make your charts more appealing and informative. The namespace provides a class corresponding to each annotation which provide properties to set its dimensions etc. You can also position or style the annotations using the properties provided by these annotation classes. Moreover, additional information can be added to these annotations in the form of a tooltip by using the TooltipText property of the class.

Annotation Type	Description
Line	Refers to a straight line whose endpoint can be specified using the **Start** and **End** properties. It also provides the **Content** property that renders the text above the line annotation.
Circle	Renders a circular shape in the plot area with its radius specified through the Radius property. Content property provided by the class lets you draw the text inside this annotation.
Ellipse	Adds an ellipse to the chart with its dimensions defined through the **Height** and **Width** property. Similar to other closed shapes, **Content** property lets you render the text inside this annotation.
Rectangle	Creates a rectangular shape in the chart plot area with specified Height and Width. Also, **Content** property lets you add the text inside this annotation.
Square	Draws a square on the plot area with its side length defined by the **Length** property. Similar to other closed shapes, **Content** property lets you render

	the text inside this annotation.
Polygon	Adds a polygon whose vertices can be added through the **Points** property which gets the collection of the vertex points. This class also provides a **ContentCenter** property which specifies the center of the text that is added using the **Content** property.
Image	Renders an image on the plot area for unmatched visual impact. The source of image can be specified through the **SourceImage** property while image size can be adjusted using the **Height** and **Width** property. An image annotation can be made more informative with the help of a tooltip.
Text	Draws the additional information in the form of text that is specified through the **Content** property.

Add an Annotation

To add annotations to your chart, create an annotation layer using the **AnnotationLayer** class and add an annotation to the Annotations collection of this annotation layer.

CS VB

Position an Annotation

There are two aspects that determine the position of an annotation on the chart, one is the position of annotation with respect to the data points and other is how an annotation is attached to the chart, that is, its position with respect to the chart. You can display an annotation in the left, right, top or bottom of the data point by setting the Position property which accepts the values from **AnnotationPosition** enumeration.

To position annotations relative to a chart, FlexChart provides the Attachment property. This property accepts following values from AnnotationAttachment enumeration.

Value of Attachment property	Description
Absolute	The annotation remains fixed irrespective of the resizing of application. In this case, position of annotation is specified in pixels.
DataCoordinate	The annotation is attached to a specific data point. In this case, position of annotation is defined by specifying the data coordinates in **Location** property.
DataIndex	The annotation is attached to the series as per the value of **SeriesIndex** property and to the point as per the value specified by the PointIndex property.
Relative	The annotation retains its location and dimensions relative to the chart. In this case, position of annotation can be defined using the **Location** property. Here, (0,0) coordinates represent the top left corner while (1,1) is for the bottom right corner of the chart.

CS VB

Style an Annotation

FlexChart provides various properties that let you customize the annotations so that they stand out against the chart background. Apart from properties related to dimensions that are provided by the corresponding annotation classes, these class also provide Style and ContentStyle properties to customize the annotation and its content respectively.

CS VB

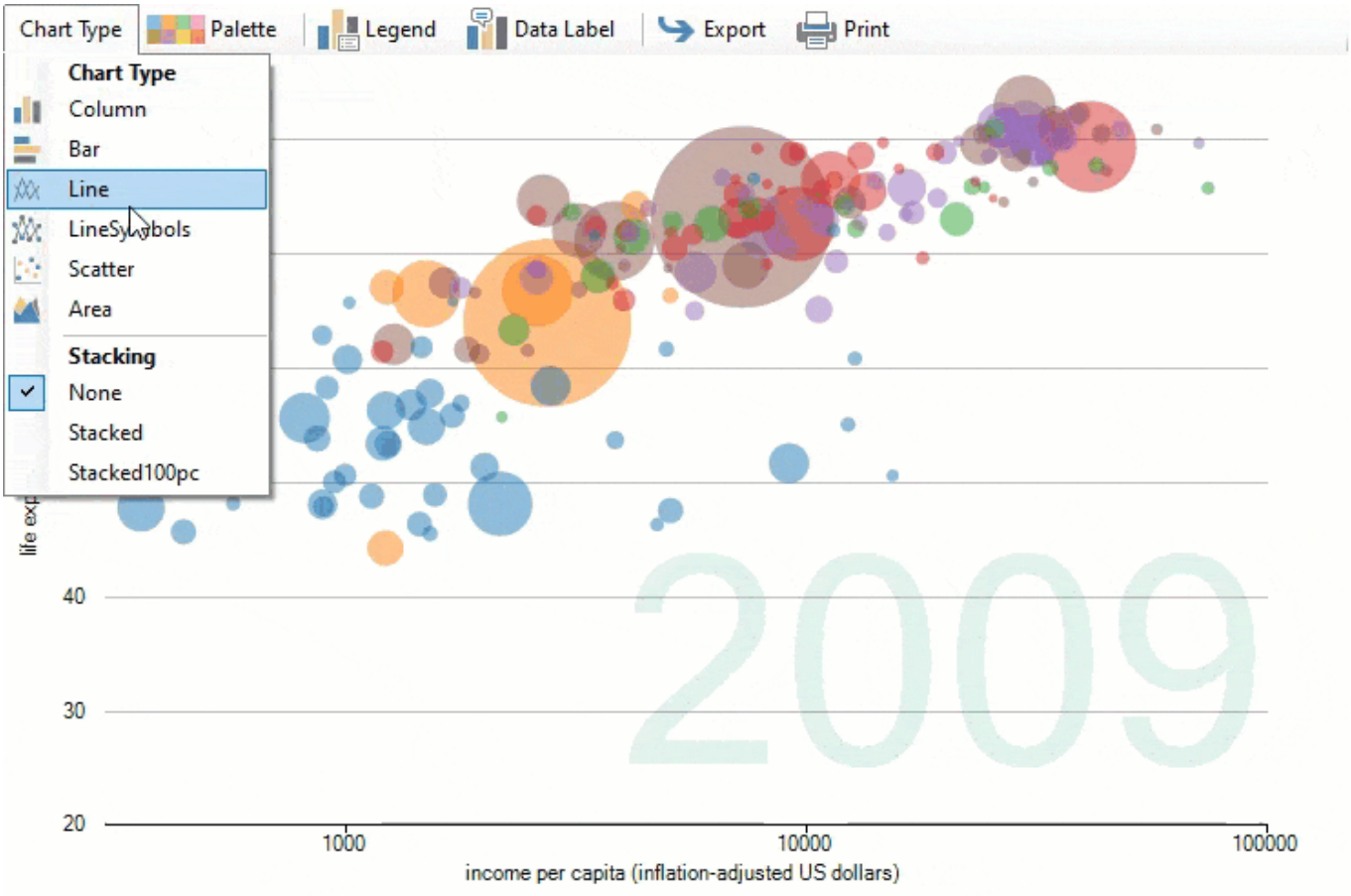
エンドユーザー操作

This section discusses about the FlexChart features related to end-user interaction.

Topic	Snapshot	Content
FlexChart Designer		Discusses about how to invoke and use the built-in chart designer. • Invoke the FlexChart Designer
Drill-down		Discusses about the drill-down feature in sunburst and treemap charts. • Drill-down in sunburst • Drill-down in treemap
Selection		Discusses about the selection and related properties in various charts. • Selection in Cartesian Charts • Selection in Pie and Sunburst • Selection in treemap
Hit Test		Discusses about using the hit test to fetch information of chart objects.
Scroll		Discusses about attaching scroll bar to the axis for detailed analysis.
Range Selector		Discusses about implementing a range selector chart for detailed analysis.
Zoom		Discusses about how to implement zoom in FlexChart.

ツールバー

FlexChart Toolbar is a runtime option you can add to perform basic charting tasks, for example, change chart type/color palette, alter current legends/data label settings, export, print charts etc. So, the toolbar helps your end-user customize the chart appearance. The FlexChartToolbar is inherited from the standard WinForms ToolStrip control and can be used or customized in the same way.



In FlexChart, the FlexChart Toolbar is represented by the **FlexChartToolbar** class provided by **C1.Win.Chart.Toolbar** namespace of the **C1.Win.FlexChart.Toolbar** assembly.

Toolbar Items

By default, the toolbar at runtime comprises the following menu buttons. Each button when clicked provides a dropdown of items. The following bulleted list provides a brief overview about the different options on the FlexChart Toolbar.

ChartType: The ChartType dropdown list-view provides different chart and stacking types.

Palette: The Palette dropdown list-view provides different color options for the user to choose from.

Legend: The Legend dropdown list-view provides different legend position options.

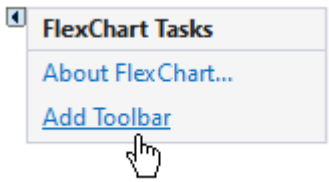
DataLabel: The DataLabel dropdown list-view provides different legend position options.

Export: The Export dropdown list provides different exporting options png, jpg, svg etc.

Print: The Print option lets you print the chart. It also provides the print preview.

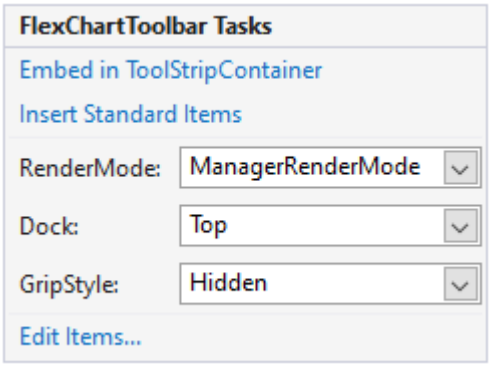
Add the Toolbar

Click the FlexChart Control's smart-tag and select the **Add Toolbar** option.



A toolbar gets added in the FlexChart control on the form.

Click the toolbar control's smart tag **FlexChartToolbar Tasks**.



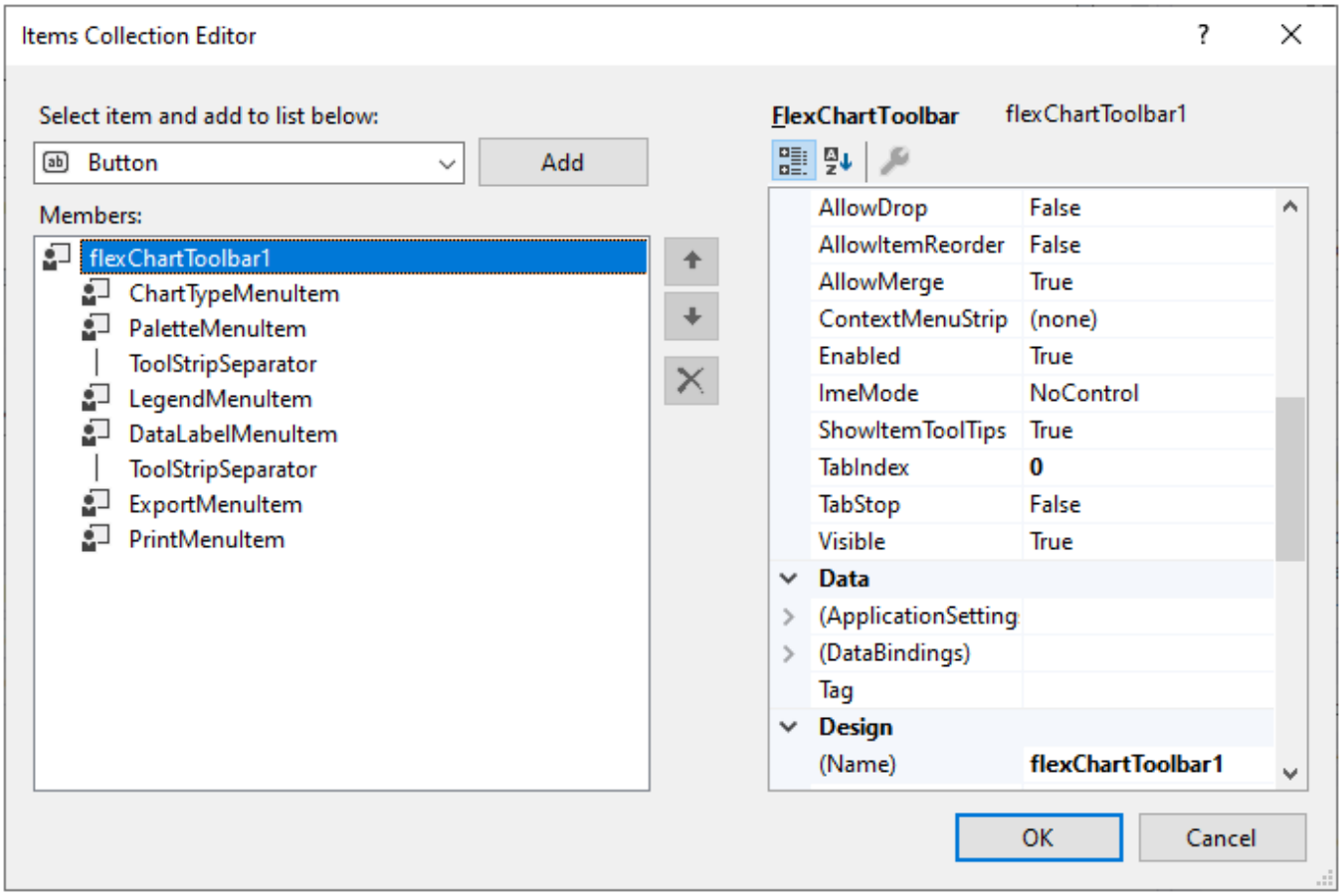
The table below lists the different options available in the smart tag:

Smart tag options	**Description**
Embed in ToolStripContainer	Allows you to add tool items into a toolstrip container.
Insert standard Items	Allows you to insert standard menu items like New, Open, Save, Cut, Copy, Paste, Help etc.
RenderMode	It indicates which visual styles will be applied to the ToolStrip. The default value is ManagerRenderMode.
Dock	Allows you to dock the control on the form.
GripStyle	Specifies the style of the sizing grip.
Edit Items	Opens the Items Collection Editor.

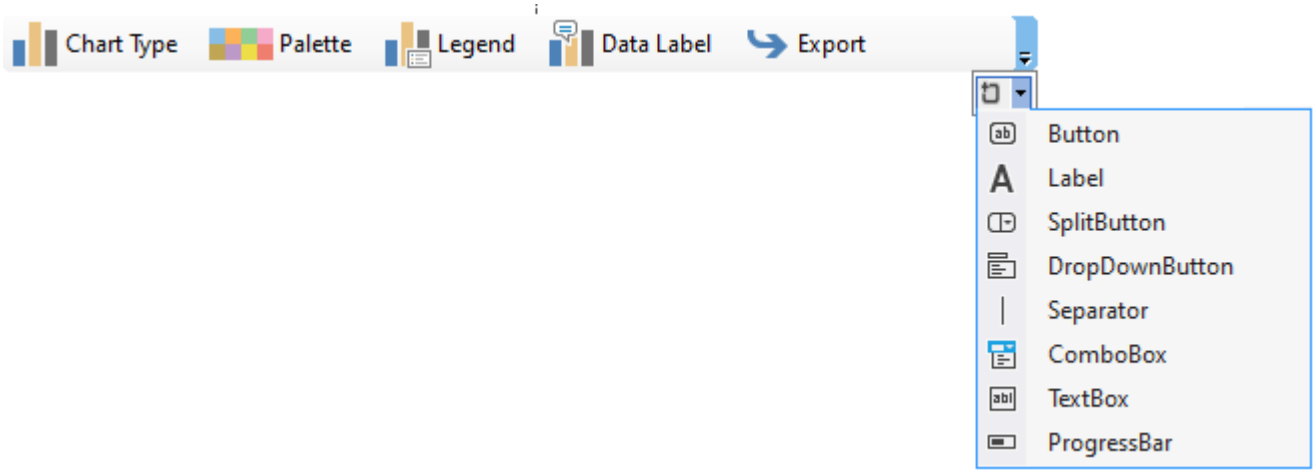
Add, Remove or Move UI Items in Toolbar

Users can add more items to the toolbar using the **Insert Standard Items** option from the smart tag. The toolbar at runtime then will appear as shown below:

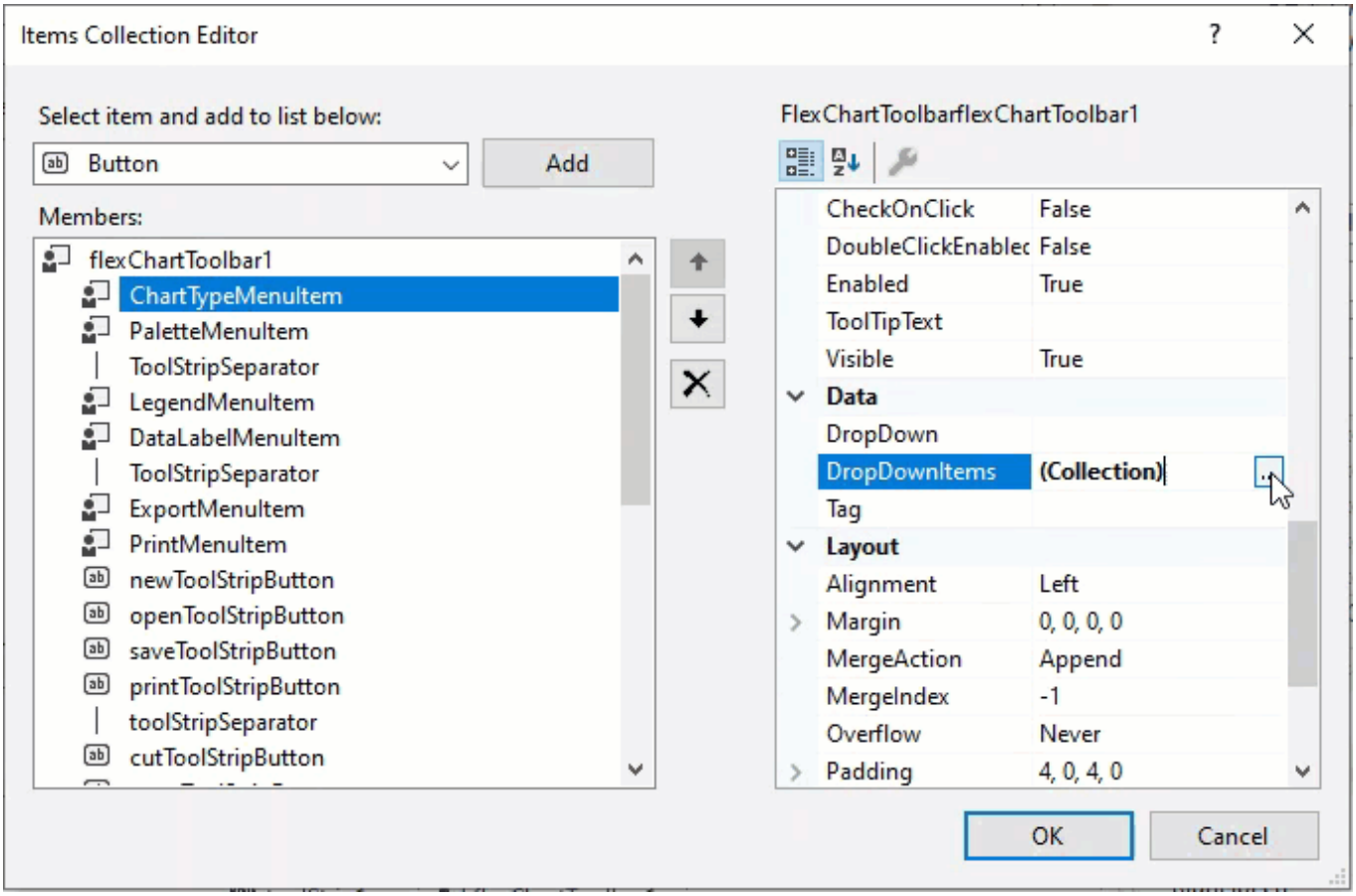
You can also add, remove, or move items, change font/layoutstyle, add back color, alter FlexChart customization options etc. using the **Items Collection Editor,** which can be accessed by clicking the **Edit Items** option in the **FlexChartToolbar Tasks** smart tag menu.



Also, add new items such as button, split button, separator, dropdown, combo box etc. to the FlexChart toolbar using the Add More items button, which opens a dropdown of items for the user to add to the toolbar.



Customize the items in the dropdown list of the Toolbar menu items using the **Items Collection Editor** as shown in the GIF below:

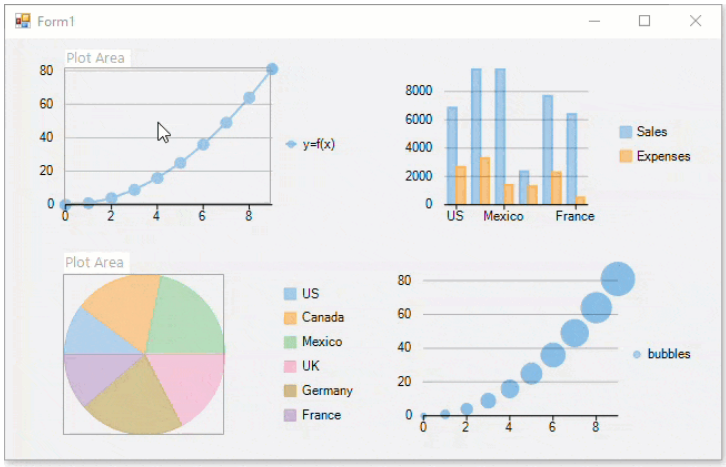


FlexChartデザイナー

Note: This feature is not yet a part of FlexChart for WinForms .NET 5 Core, and might be added in the future releases.

Chart Designer refers to a run-time designer that empowers the end-user to customize the chart by changing the chart properties while visualizing the same. Such designers come really handy in applications such as dashboard designers which require end-user to customize the chart at run-time as per requirement.

FlexChart Designer is a compact and responsive designer which appears on clicking the chart elements at run-time and provides a user interface to change various settings related to **Chart**, **Data**, **Header**, **Footer**, **Legend**, **X-Axis**, **Y-Axis** and **Data Labels**.



The chart elements such as plot area, x-axis, y-axis, legend, header and footer get highlighted as the user moves the mouse cursor over the chart. The

ChartDesigner appears when any of these elements is clicked for the first time and displays the property list of the corresponding chart element. This property list in the designer keeps on changing depending on the element selected in the chart area. You can also select the desired chart element from the drop-down menu available on the top-left corner of the designer. The table below describes each of these options provided by the FlexChart Designer drop-down menu.

Drop-down Options	Descriptions
Chart	Displays the properties related to appearance of the chart such as **BackColor**, **ForeColor**, **PlotColor**, **Font** and **Stacking**. It also provides the **ChartType** property that lets you change the chart type to any of the basic charts at run-time. This property is not displayed in the case of charts such as pie chart, treemap or sunburst chart.
Data	Displays all the data fields of the chart. It lets you bind the data simply by dragging and dropping data fields on to X-axis and Y-axis sections on the right-hand-side of the window. You can easily create multiple series or multiple pie charts by using this option.
Header	Lets you customize the chart header. Using this option, you can add, remove, edit or style the header content through properties such as **Content**, **Color**, **Font**, **Border**, **BorderFill** etc.
Footer	Lets you customize the chart footer. Using this option, you can add, remove, edit or style the footer content through properties such as **Content**, **Color**, **Font**, **Border**, **BorderFill** etc.
Legend	Allows you to specify the legend related settings of the chart. You can choose to hide, display or style the chart legend by using the properties such as **Title**, **Position**, **ForeColor**, **Font** etc.
X-Axis	Displays the properties related to the X-Axis of chart like **Title**, **Labels**, **LabelAngle**, **Min**, **Max**, **MajorTicks** etc. This option does not appear in the drop-down menu list in the case of charts such as pie chart, treemap or a sunburst chart.
Y-Axis	Displays the properties related to the Y-Axis of chart like **Title**, **Labels**, **LabelAngle**, **Min**, **Max**, **MajorTicks** etc. This option does not appear in the drop-down menu list in the case of charts such as pie chart, treemap or a sunburst chart.
Data Label	Lets you display, hide, position or style the data point values on charts. Some of the data label related properties provided by the designer are **Content**, **Color**, **Font**, **Connecting Line**, **Position** etc.

Invoke the FlexChart Designer

In FlexChart, the **FlexChart Designer** is represented by the **ChartDesigner** class provided by **C1.Win.Chart.Designer** namespace of the **C1.Win.FlexChart.Designer** assembly. You can easily invoke the designer for a particular FlexChart by creating an instance of the **ChartDesigner** class and passing the FlexChart control as a parameter to this ChartDesigner constructor.

CS VB

Note that source code of the FlexChart Designer is available in the form of product sample named **ChartDesigner** so that you can customize it as per your application requirements. The ChartDesigner sample is located at \Documents\ComponentOne Samples\WinForms\v4.5.2\C1FlexChart\CS\ChartDesigner on your system, if you have installed the samples while installing WinForms Edition using **ComponentOneC1ControlPanel.exe**.

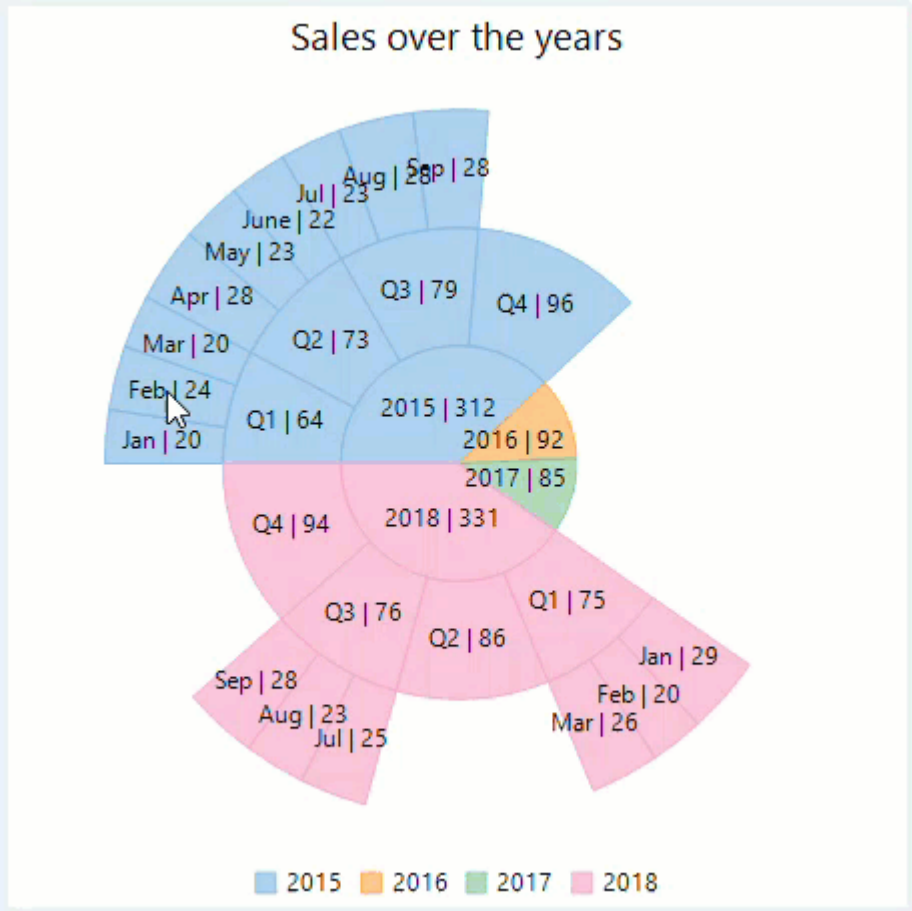
ドリルダウン

Drilling down in charts refer to the feature of displaying hierarchical data and giving the end-user ability to navigate through the various levels by simply clicking on the chart. This lets the user analyze the data at different levels that too within the same chart area.

FlexChart provides built-in drill down feature in treemap and sunburst chart type. End-user can even drill up the chart by right clicking on deeper levels of these charts.

Drill-down in Sunburst

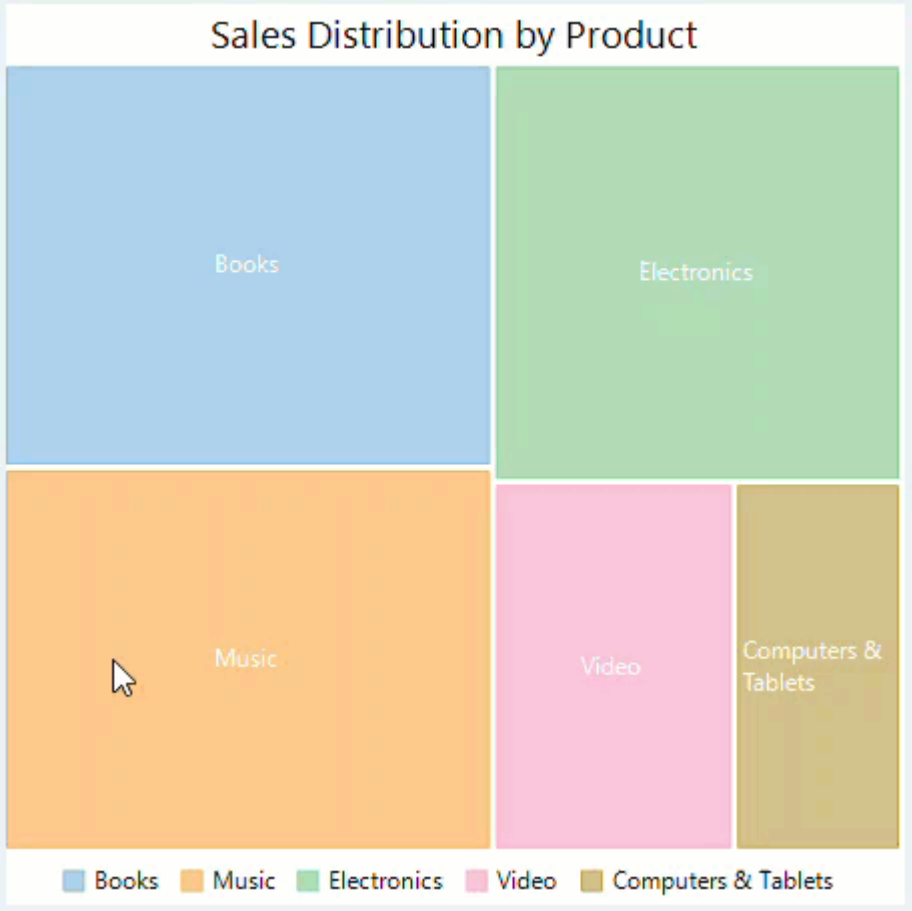
FlexChart provides the Drilldown property of Sunburst class, which when set to **True**, enables drilldown in sunburst chart. Note that as selection and drill down feature, both work on mouse-click, you need to disable the selection by setting the SelectionMode property to **None**.



CS | VB

Drill-down in TreeMap

FlexChart enables drill-down feature in treemap by setting the MaxDepth property of TreeMap class to an integer value greater than **0**. Note that as selection and drill down feature, both work on mouse-click, it is recommended to disable the selection by setting the SelectionMode property to **None** while using the drill-down feature.

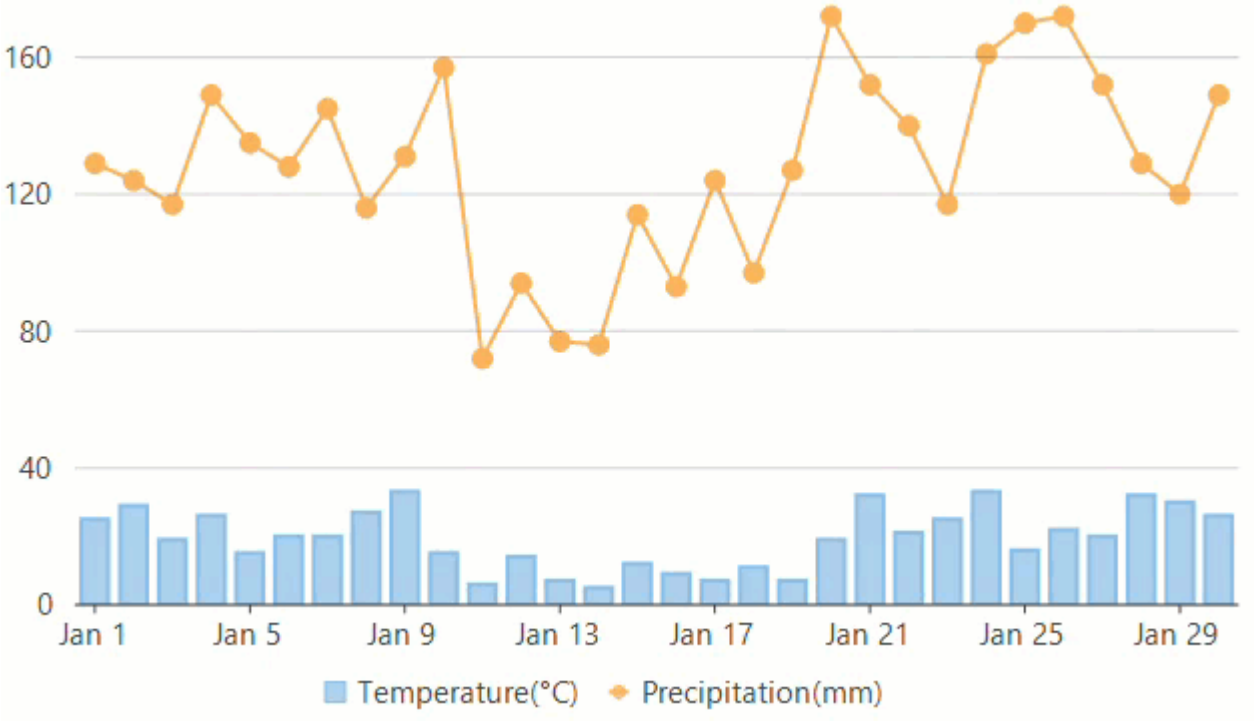


CS VB

選択

Selection in charts helps the end-user to select the required data point or the whole series at run-time during analysis, etc.

In **FlexChart**, selection is disabled by default. However, you can enable the same by setting the SelectionMode property which accepts value from ChartSelectionMode enumeration. You can choose to enable the point or series selection by setting value of this property to **Point** or **Series** respectively. Note that there are some exceptions and some of the charts like pie chart, sunburst chart and treemap do not have any effect when this property is set to **Series**. Similarly, in case of simple line charts and financial charts, setting the value to **Point** does not reflect any change in the run-time functionality.



FlexChart also lets you get or set the index of the selected item using the SelectedIndex property. Moreover, it provides a SelectionChanged event which notifies when the selected point or series is changed so that you can customize the selection process according to the requirements.

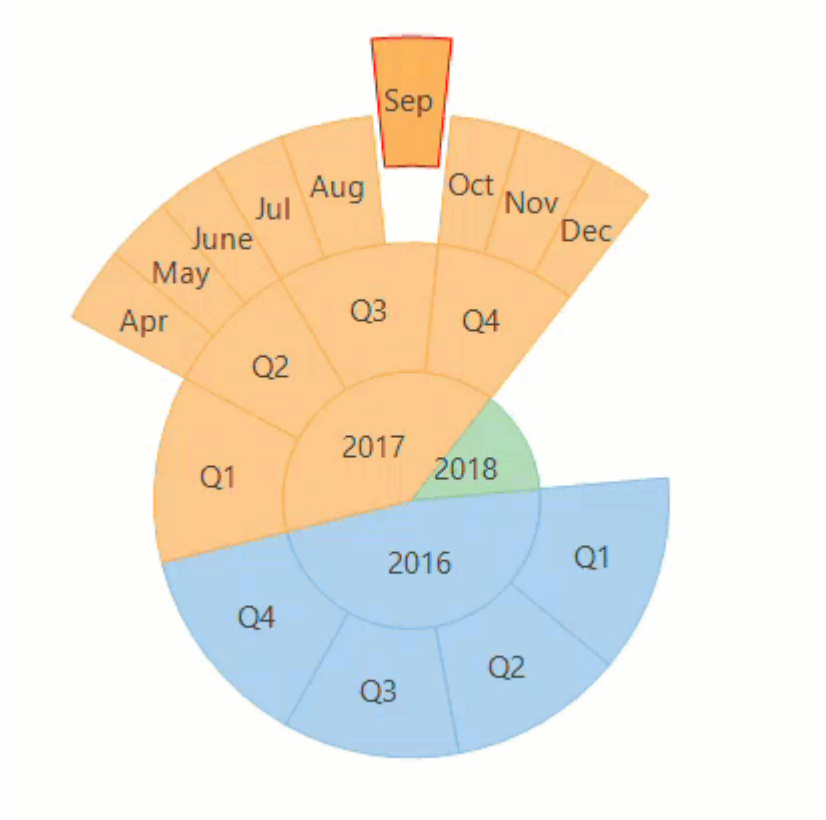
By default, FlexChart highlights the selection through a red colored solid line. However, you can also customize how a selected item should appear by using the SelectionStyle property. The property is of ChartStyle type and lets you change the fill, fill color, stroke, stroke color, line pattern etc.

CS VB

Apart from the abovementioned common settings related to selection, some chart types like pie chart, sunburst chart and treemap have some chart specific properties as well which are discussed in the sections below.

Selection in Pie and Sunburst Chart

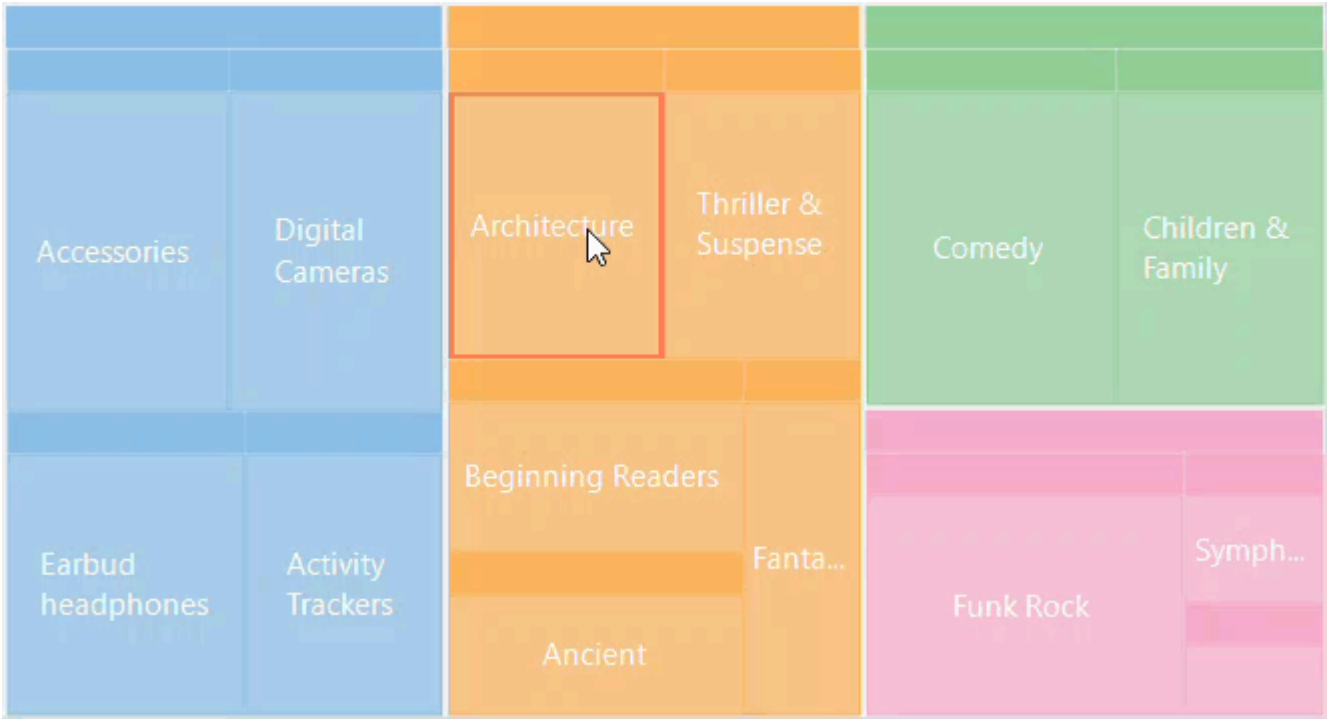
Pie and sunburst chart, being different from usual charts have some properties that are specific to them and help in enhancing the end user experience. When SelectionMode property is set to **Point**, by default, the charts highlight the selected slice with a red colored line. However, to give it an additional impact, you can even set SelectedItemOffset so that the selected slice displays away from rest of the chart. The FlexPie class also provides a SelectedItemPosition property which lets you choose to display the position of selected slice. That is, when you select a particular slice, the charts rotate to always display the selected slice in your chosen position, on top, bottom, right, left etc.



CS VB

Selection in TreeMap

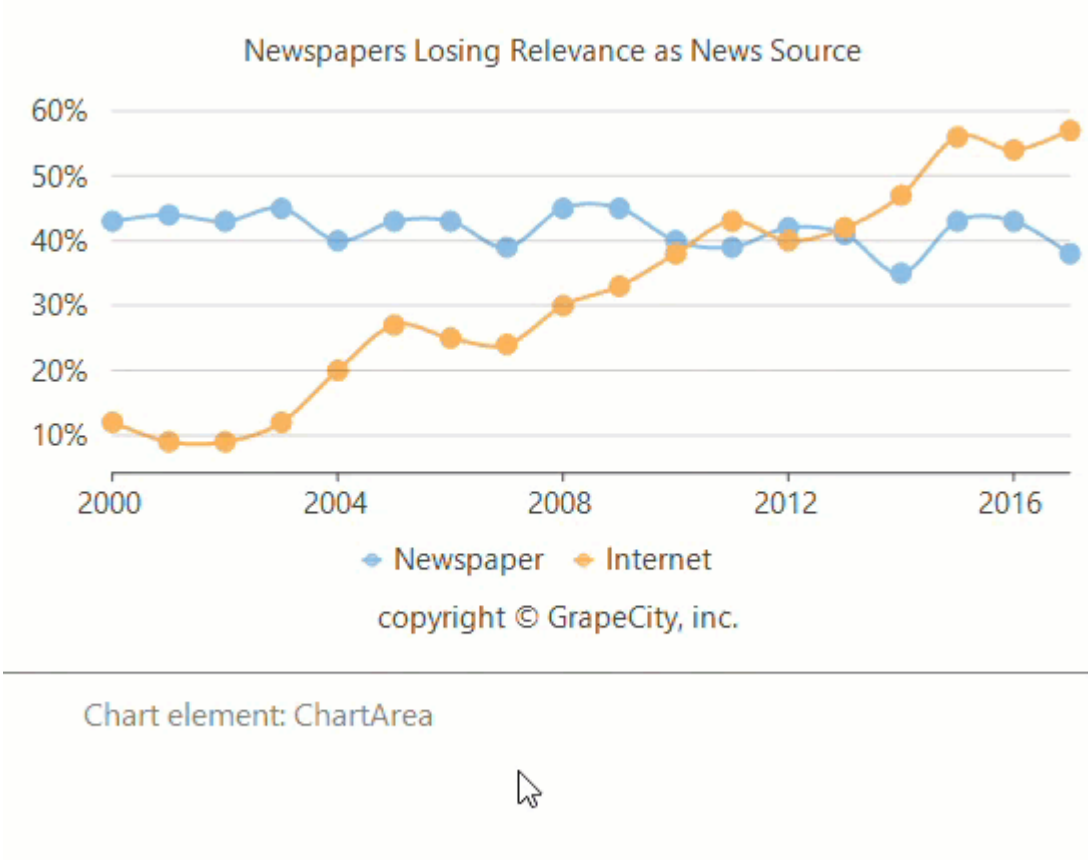
TreeMap lets you select the individual data items or groups by simply clicking on them. Other than the common properties mentioned above, the TreeMap class also provides the SelectedItem property which lets you get or set the selected data item.



CS VB

ヒットテスト

Hit test in charts refers to the ability to fetch information about the chart object under the mouse and lets you create interactive applications. For instance, this feature can be used for displaying a special custom tooltip when mouse hovers over a particular data point. Similarly, you can even use the hit test information for drilling down the chart data, to set alerts and enable other user interaction functionalities.



FlexChart provides HitTest method through FlexChart class to fetch information about the underlying chart objects. This method has following two overloads and returns an object of the HitTestInfo class which provides information such as chart element under pointer, distance from the closest data point, index of the nearest data point etc.

- **HitTest (Point)**
- **HitTest(Point,MeasureOption,Int32)**

To use this method to fetch information, you need to subscribe to a mouse event on which you want to fetch the information and then invoke the **HitTest** method in its event handler. You can then use this information in the required manner. In the example shown in this topic, we have fetched the information of chart objects on mouse movement and have displayed it in the information panel below the chart.

CS VB

スクロール

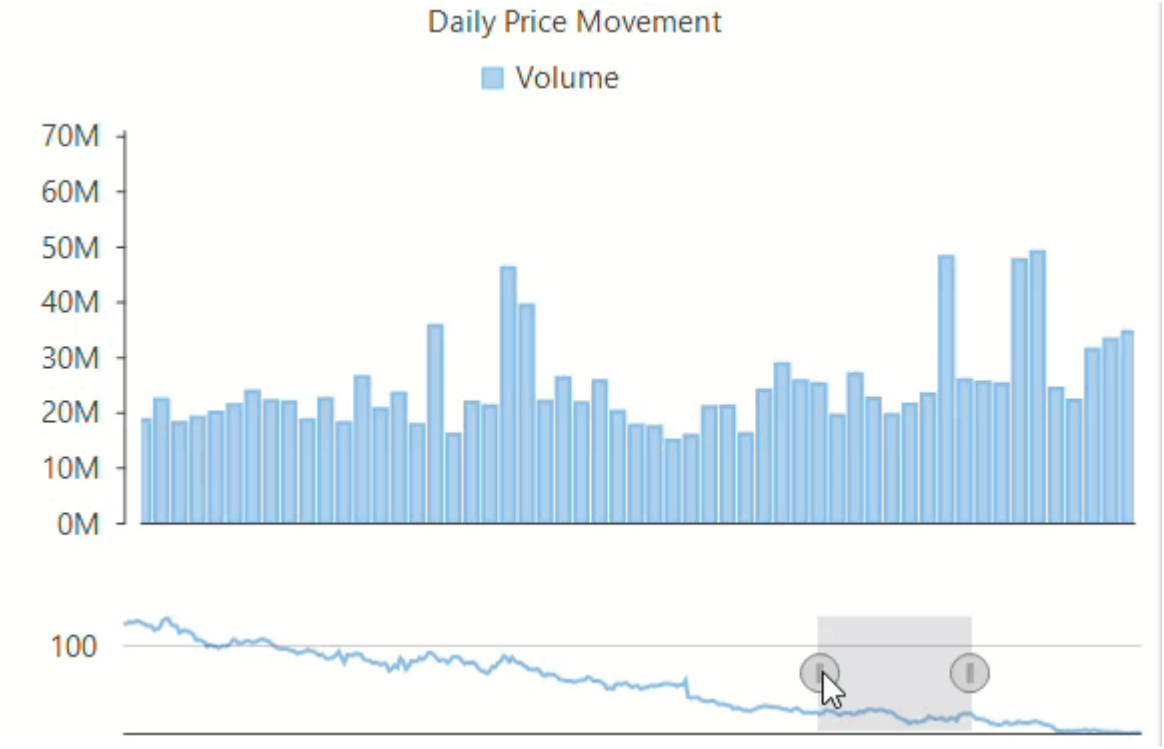
Scrolling in charts comes into picture when huge amount of data needs to be plotted in a limited space but requires detailed analysis. This feature provides end-user with an ability to focus on analysis of the selected range of values instead of the whole data. For instance, while presenting the daily price movement of a stock across an year, axis scroll bars can enable end-user to focus on movement of data in a month or even a week.

FlexChart allows you to add scroll bars to X-axis as well as Y-axis by using AxisScrollBar class of C1.Win.Chart.Interaction namespace. To attach a scroll bar to an axis in FlexChart, you need to create an instance of the **AxisScrollBar** class and pass an Axis object as a parameter to it. By default, the scroll bar gets displayed with two thumbs that define the range of current selection using the UpperValue and LowerValue properties. The upper and lower value changes when user drags these thumbs at run-time and a ValueChanged event is fired. A scroll bar also consists of two scroll buttons on extreme ends which when clicked helps the end-user to scroll the selected range. You can choose to hide these buttons by setting the ScrollButtonsVisible property to **False**.

CS VB

範囲セレクタ

Range selector is a modern approach of scrolling the charts with huge data. In this case, instead of usual scroll bars, another broad view chart is displayed so that end-user can select the desired range more precisely and effectively. Just like axis scroll bars, range selector also acts as a tool for end-user for analyzing the selected range of data in detail. Analysis of stock charts is one of the good use case example of a range selector.



FlexChart provides the RangeSelector class of C1.Win.Chart.Interaction namespace to implement a range selector. To add range selector to a chart, you need to create two instances of the FlexChart class, one for selecting the range and other for displaying the selected range. Set up the two charts and finally, pass the instance of the range selector FlexChart to the **RangeSelector** class. By default, the range selector gets displayed with two thumbs that define the range of current selection using the UpperValue and LowerValue properties. The upper and lower value changes when user drags these thumbs at run-time and a ValueChanged event is fired.

CS VB

In this example, set up of the two charts has been done in the following way:

CS VB

ズーム

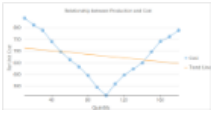
Zooming in charts allows users to select the area they want to enlarge by clicking and dragging the mouse, enabling them to analyze the data packed charts at granular level.

In FlexChart, zooming can be implemented by tracking the region selected by the user and then setting the maximum and minimum value of axes based on the selected region. Bounds of the selected region can be fetched by storing the start position of the mouse in the **MouseDown** event, updating the selection region in the **MouseMove** event and then, finalizing the zoom region when user releases the mouse button, that is, in the **MouseUp** event. The bounds determined in this way are then used to set Min and Max properties of each axis and thereby displaying the zoomed chart.

CS VB

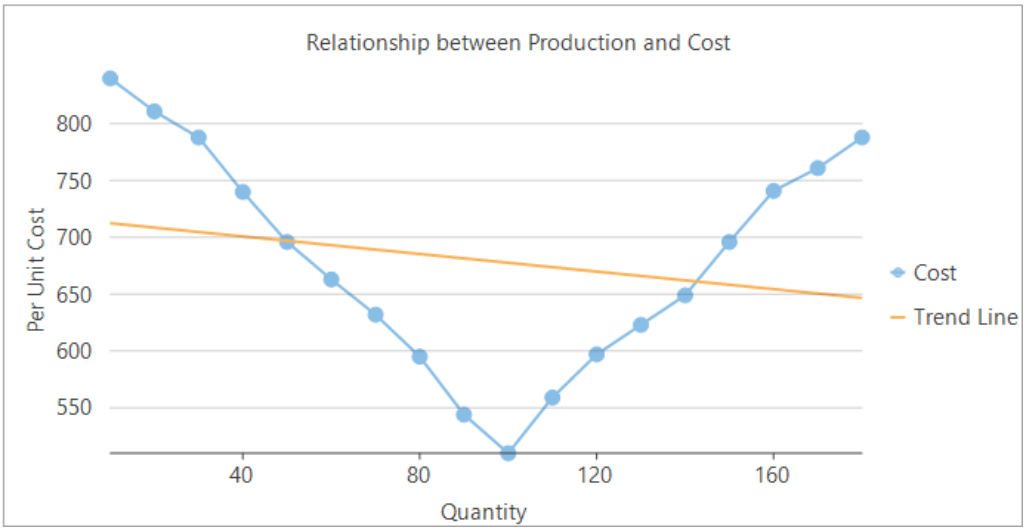
分析

This section discusses various tools to analyze the charts.

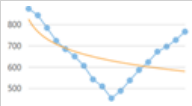
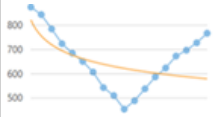
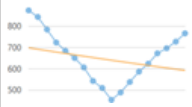
Topic	Snapshot	Content
Trend Line		Discusses various types of trend lines and how to add a trend line.

近似曲線

Trend line refers to a straight or a curved line that is superimposed on the chart to inform user about direction of the data or the trend, and hence helps in predicting the future values. Due to ability to depict future prices, trend lines are often used in trade analysis to understand the price movement and forecast the value of securities.



In FlexChart, trend lines can be implemented by creating an instance of the TrendLine class. Then, you need to bind the trend line to a data source, set other relevant properties and add it to the Series collection. FlexChart supports regression as well as non-regression trend lines and the fit type and order of these lines can be specified using the FitType property and Order property of this class respectively. Following is the list of various fit types supported in the FlexChart control:

TrendLine.FitType	Snapshot	Description
Linear		A linear trend line is the straight line that most closely approximates the data in the chart. The data is linear, if the data pattern resembles a line. Equation - Y(x) = C0 + C1*x
Polynomial		Polynomial trend lines are curved lines that are used with fluctuating data. They are useful for analyzing gains or losses over a large data set. When using a polynomial trend line, it is important to also set the Order of the line, which can be determined by the number of fluctuations in the data. Equation - Y(x) = C0 + C1*x + C2*x2 + : + Cn-1*xn-1
Logarithmic		Logarithmic trend line 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 trend line can use negative and/or positive values. Equation - Y(x) = C0 * ln(C1*x)
Power		Power trend line is a curved line that is best used with data sets that compare measurements that increase at a specific rate — for example, the acceleration of a race car at one-second intervals. You cannot create a power trend line if your data contains zero or negative values. Equation - Y(x) = C0 * pow(x, C1)
Exponent		Exponential trend line is a curved line that is most useful when data values rise or fall at increasingly higher rates. You cannot create an exponential trend line if your data contains zero or negative values. Equation - Y(x) = C0 * exp(C1*x)
Fourier		Fourier trend line identifies patterns or cycles in a series data set. It removes the effects of trends or other complicating factors from the data set, thereby providing a good estimate of the direction that the data under analysis will take in the future. Equation - Y(x) = C0 + C1 * cos(x) + C2 * sin(x) + C3 * cos(2*x) + C4 * sin(2*x) + ...

MinX		The minimum X-value on the chart.
MinY		The minimum Y-value on the chart.
MaxX		The maximum X-value on the chart.
MaxY		The maximum Y-value on the chart.
AverageX		The average X-value on the chart.
AverageY		The average Y-value on the chart.

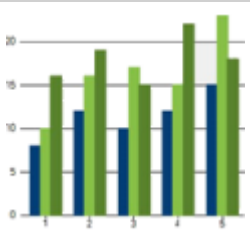
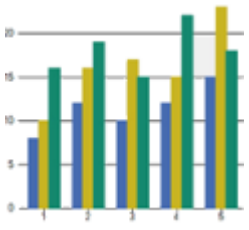
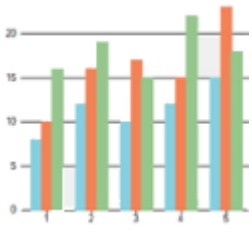
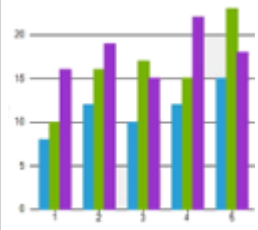
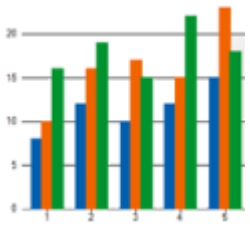
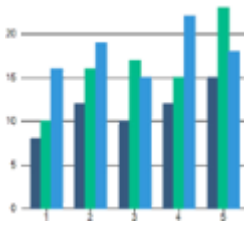
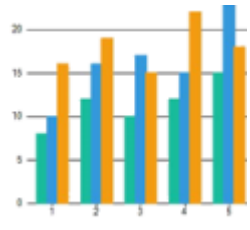
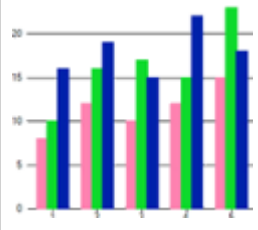
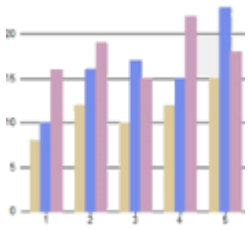
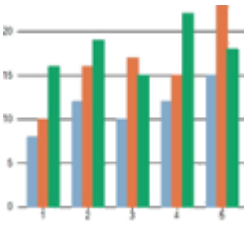
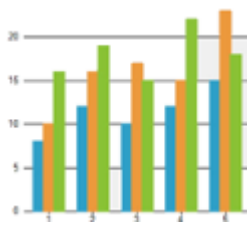
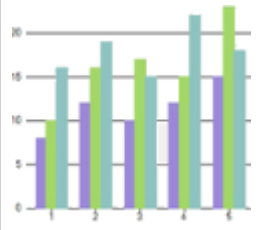
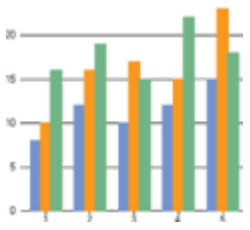
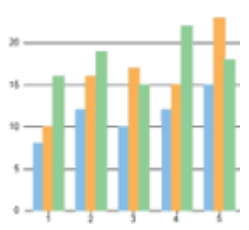
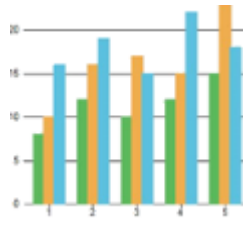
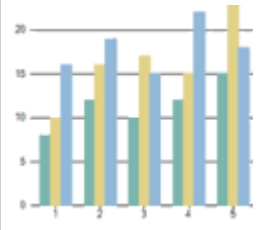
CS VB

外観とスタイル

FlexChart provides various options to customize the appearance of chart and chart elements individually, so that you can generate charts as per your requirement and look and feel of the application you are creating. this topic discusses all the appearance related options available with the FlexChart.

Chart Palettes

FlexChart provides 16 pre-defined color palettes, so that you can generate the charts with desired appearance with a single line of code. You can apply these palettes to FlexChart using the Palette property which accepts the values from Palette enumeration of C1.Chart namespace. By default, this property is set to **Standard**. You can also create a custom palette using the existing color palettes.

Cerulan	Cocoa	Coral	Cyborg
			
Dark	**Darkly**	**Flatly**	**HighContrast**
			
Light	**Midnight**	**Modern**	**Organic**
			
Slate	**Standard**	**Superhero**	**Zen**
			

CS VB

Styling of Chart Elements

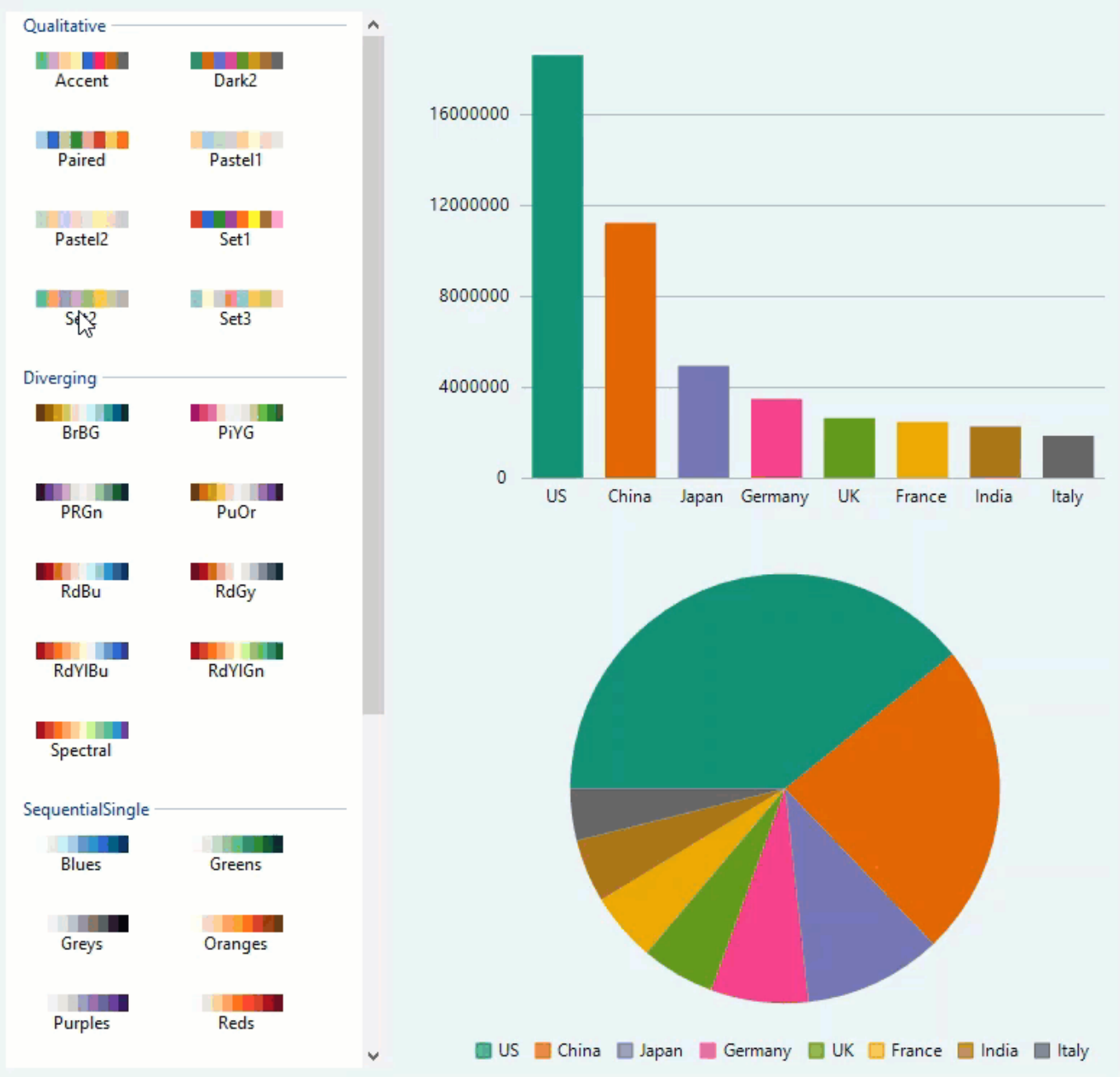
Apart from the color palettes, FlexChart also lets you style the chart titles, axis titles, axis, data labels, legend and plot area. For styling of specific elements, see the styling section of corresponding topics.

Style chart titles	Style axis titles	Style a series	Style an axis
Style data labels	Style legend	Style plot area	Style an annotation

Extended palettes

FlexChart now provides more palette options. These palettes were inspired by ColorBrewer 2 and are grouped in the following categories:

- Qualitative
- Diverging
- SequentialSingle
- SequentialMulti



The Extended Palettes feature is provided by the **C1.Win.Chart.Palettes** namespace. You can apply these palettes to FlexChart using the **CustomPalette** property. Each category of palettes also exposes the brushes dictionary, which the user can enumerate or use as a data source.

C#	Copy Code

```
chart.Palette = Palette.Custom;
chart.CustomPalette = C1.Win.Chart.Palettes.Qualitative.Accent;
```

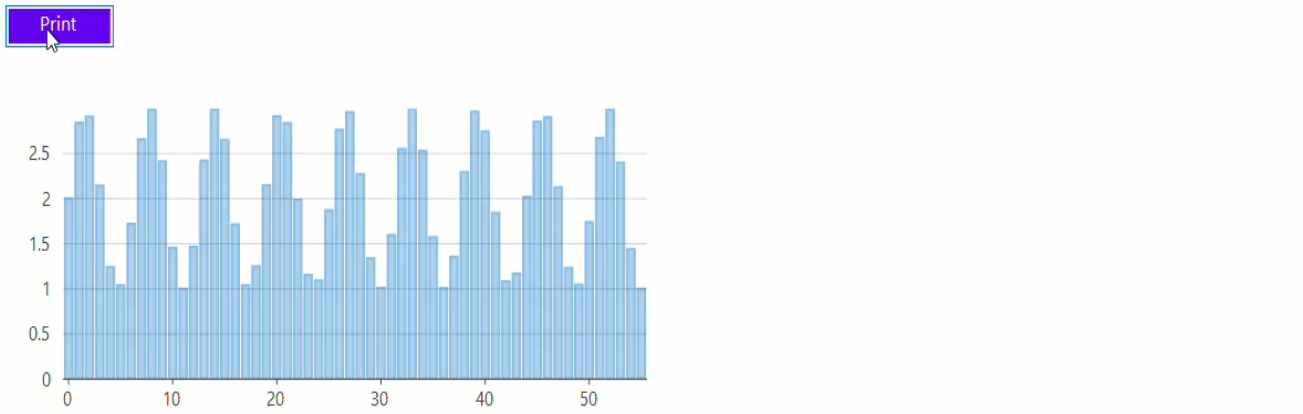
The different categories of these extended palettes are useful in varying scenarios as listed below:

Palette Type	**UseCase**
Qualitative Palette (Represented by the **Qualitative** class)	To show nominal or categorical data.

Multi (Represented by the **Diverging** class)	To give equal emphasis to mid-range critical values and extremes at both ends– light colors for mid-range and dark colors for extreme low and high values.
Sequential Single/SequentialMulti (Represented by the **SequentialSingle** or **SequentialMulti** class)	To show ordered data that progress from low to highlight colors for low-data values to dark colors for high-data values.

印刷

Printing a readable chart could be a tricky task, especially, while printing charts with large data. Depending on the use case and requirement, there can be multiple ways in which a chart can be printed. For instance, printing a chart on a single page is a good idea when chart has less data or you want to understand the overall pattern of data without knowing the detailed data points. On the other hand, when you want to study chart details, you can display one chart on each page or even a single chart divided into sub-charts or strip charts on multiple pages.



In FlexChart, you can carry out printing by using the **C1.Chart.FlexChart.Printing** assembly that can be obtained by building the product sample named C1.Chart.FlexChart.Printing and accessing obj\Debug folder inside the project. This assembly provides the **ChartPrinter** class to achieve printing in FlexChart. The sections below discuss how this class can be used in various printing scenarios.

> **Note: C1.Chart.FlexChart.Printing** sample is located at \Documents\ComponentOne Samples\WinForms\v4.5.2\C1FlexChart\CS\FlexChartPrint on your system, if you have installed the samples while installing WinForms Edition using **ComponentOneC1ControlPanel.exe**.

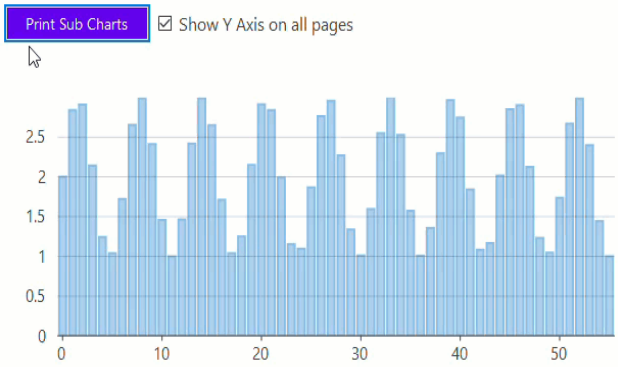
Single Page Printing

In FlexChart, you can print a chart on a single page by creating an instance of the above-mentioned ChartPrinter class. The constructor takes two parameters, one is the target FlexChart instance and other one is a Boolean value that indicates whether to display the print dialog or not. You, then need to specify whether you need to scale the chart for printing by using the **PrintScale** enumeration which lets you set the chart scaling to none, fit, stretch and zoom according to the available space. You can also opt to set the custom page and printer settings, grayscale etc. as per the requirement and then call the **PrintCtrl** method which accepts PrintScale and some of the settings as its parameters.

CS VB

Multi-page Printing

There can be various scenarios in which a chart or charts are required to be printed on multiple pages. To achieve the multi-page printing, we need to use the above-mentioned **ChartPrinter** class in conjunction with the PrintDocument class. The **PrintDocument** class, along with the settings related to the print, provides the **PrintPage** event which is required to implement the main printing job. You can use **PrinterCtrlToPage** method of the ChartPrinter class, **DrawChart** method of the **FlexChart** class or **DrawImage** method of the **Graphics** class as per the layout requirements. In this example, we have demonstrated how to print a single chart on multiple pages in the form of sub-charts.



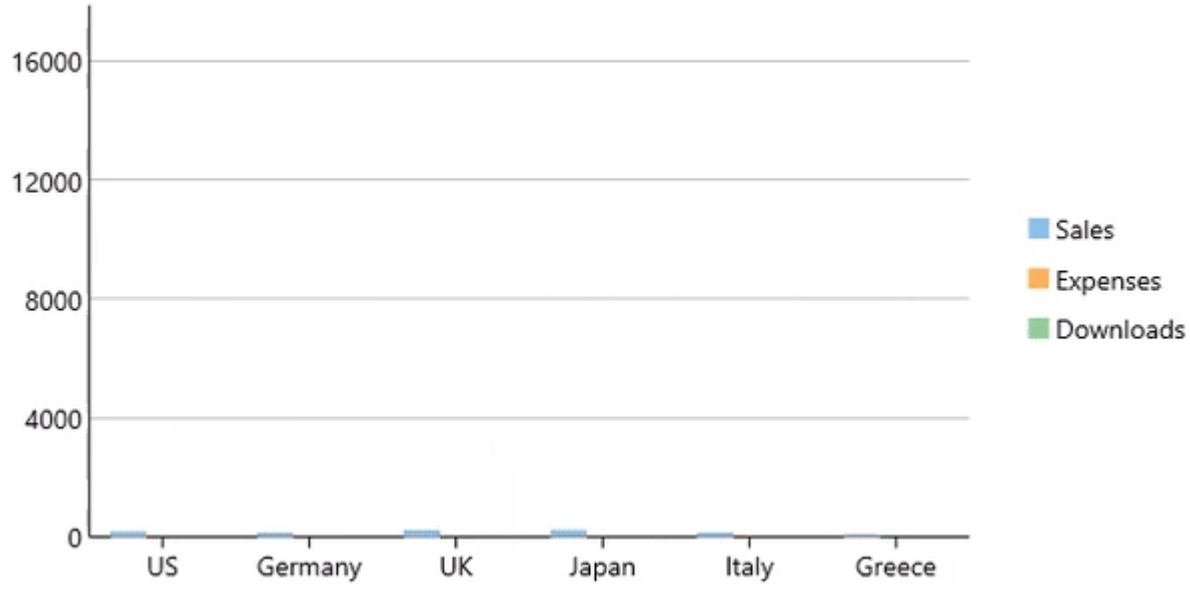
CS VB

For more printing scenarios and detailed implementation, see **FlexChartExplorer** sample which is shipped with the control. To see this feature in action, you can also download the **FlexChartExplorer** demo from our website.

アニメーション

Animation gives an edge to any application and enhances the user experience by bringing life to its otherwise static elements.

FlexChart provides out of the box features to animate the charts. The AnimationSettings property of FlexChart class allows you to choose whether and where to show the animation. this property accepts the values from **AnimationSettings** enumeration and lets you set the animation to appear in case of chart load or update, axes load or update or in all situations. Variety of animation options help in making your charts dynamic and can be set using the AnimationLoad and AnimationUpdate properties. These properties are of AnimationLoadOptions and AnimationOptions type respectively and gives you access to the properties to set these options such as easing effects, duration, direction, type etc.



CS VB

Animation in Pie Chart and Sunburst Chart

In the case of a pie chart or sunburst chart, AnimationSettings property of the FlexPie class lets you set the animation on chart load, update or both. Just like **FlexChart** class, the **FlexPie** class also lets you set the **AnimationLoad** and **AnimationUpdate** properties to specify various animation options such as easing, duration, as well as the slice attributes through PieAnimationOptions class.

CS VB

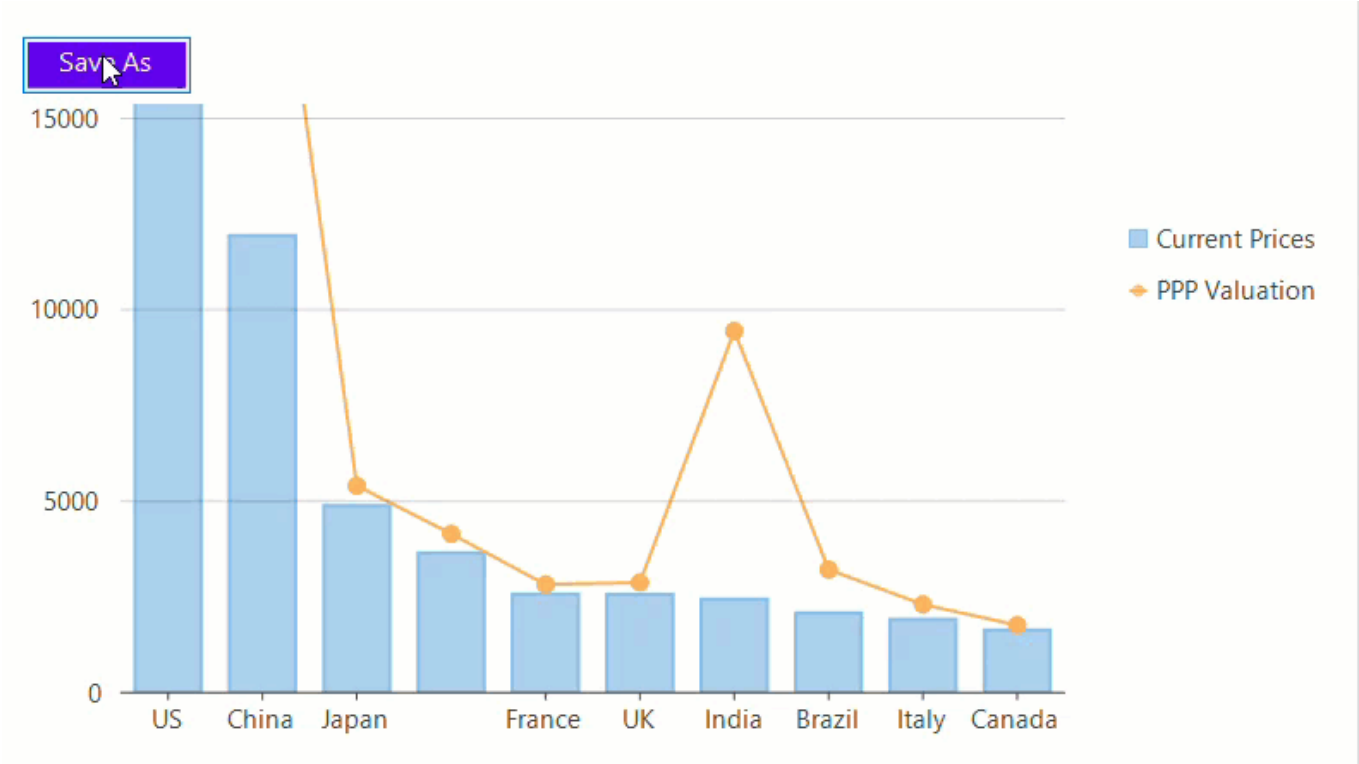
エクスポート

Exporting a chart refers to saving it in some other file format to enable the end-user to store it for later use. For instance, a chart saved as an image can be used in a presentation or any other application.

Save as Image

FlexChart provides the **SaveImage** method of FlexChartBase class that lets you save a chart as an image to the specified stream. This method takes four parameters, the stream, the image format, width and height of the image. The formats currently supported by the FlexChart control are **.png**, **.jpeg**, and **.svg**.

Also, the **FlexChartBase** class also provides the **SaveImage (int w, int h)** method to save the chart as image to the clipboard, with the method taking two parameters, the image width and height.



CS VB

Serialization

Serialization refers to the conversion of chart object in to a sequence of bytes or a file, that can be stored and transmitted. This concept is generally used when data related to objects have to be transferred from one application to another to replicate the same in another application for further use.

In FlexChart, you can serialize chart into various file formats using the **C1.Win.Chart.Serialization** assembly. You can obtain this assembly by building the product sample named **C1.Win.Chart.Serialization** and accessing obj\Debug folder inside the project. This assembly provides the **Serializer** class which provides methods to serialize a chart to the xml, json, binary and base64 formats. In this example, we are using **SerializeChartToFile** method that lets you serialize

a chart to any of these formats. This method accepts three parameters, file name to which FlexChart object properties are to be stored, the FlexChart instance to be serialized and the file format to which FlexChart instance is to be saved. Similarly, you can use other methods such as **SerializeChartToXml** for serializing chart to a specific format such as XML in this case.

> **Note: C1.Win.Chart.Serialization** sample is located at \Documents\ComponentOne Samples\WinForms\v4.5.2\C1FlexChart\CS\FlexChartSerializer on your system, if you have installed the samples while installing WinForms Edition using **ComponentOneC1ControlPanel.exe**.

CS VB

De-serialization

De-serialization refers to the process of reading the state of object stored in a byte stream to import the original object. In FlexChart, just like serialization, you can de-serialize these settings saved in a particular file format to re-construct the chart by using various de-serialization methods exposed by the **Serializer** class of C1.Win.Chart.Serialization assembly. In this example, we are using **DeserializeChartFromFile** method which accepts three parameters and can re-create a chart from any file format. The three parameters are name of the file that contains the FlexChart object properties, FlexChart instance to be recovered and format of the file from which FlexChart instance has to be recovered.

CS VB

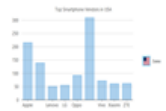
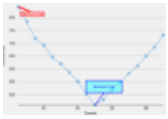
サンプル

All the ComponentOne Editions are accompanied with a wide range of product samples to facilitate a better understanding of controls' features and their implementation. To install the samples, you need to select the "**Install Samples**" check box while **installing the edition** using C1ControlPanel. By default, these samples are downloaded and stored in a control specific folder at following location: **Documents\ComponentOne Samples\WinForms**. For FlexChart, following CS samples are installed in the folder named **C1FlexChart**:

Sample	Description
AnimationDemo	Demonstrates the advanced animation options available in FlexChart and FlexPie through an interactive demo.
AnnotationExplorer	Demonstrates how to use the annotation layer to render various built-in annotations such as circle, ellipse, image, line, polygon, rectangle and square. The sample also shows how to create callouts using the polygon type annotation.
AxisScrollBar	Demonstrates how to implement AxisScrollBar to accomplish detailed chart analysis using the FlexChart control.
CurrencyComparison	Demonstrates how to use FlexChart to compare the exchange rates or percentage change of different currencies.
DataManipulation	Demonstrates how to perform aggregation or sorting operation on data getting displayed in FlexChart.
DrawingTools	Demonstrates the implementation of DrawingLayer attached to FlexChart to add and edit the chart annotations interactively.
DrillDown	Demonstrates how to implement drill-down functionality with data grouping in FlexChart, FlexPie and Sunburst controls.
ExtendedFeatures	Demonstrates features included in the C1.Win.FlexChart.Extended.dll library such as heatmap, color axis etc.
FlexChart101	Demonstrates how to get started with the FlexChart control and also shows implementation of some basic features.
FlexChartCustomization	Demonstrates how to use the custom symbols, lines, legend items etc. in FlexChart.
FlexChartEditableAnnotations	Demonstrates the advanced implementation related to annotations. The sample also shows how to add, edit, move or remove an annotation at run-time.
FlexChartExplorer	Demonstrates all the chart types and features of FlexChart in an easy to explore manner.
FlexChartPrint	Demonstrates how to print FlexChart in different ways such as single chart on multiple pages or a multiple charts on a single page. This sample creates an assembly which encapsulates most of the printing logic and can be re-used in other applications to implement printing scenarios.
FlexChartSerializer	Demonstrates how to serialize/deserialize FlexChart properties to or from XML, JSON and binary formats. The sample also provides library to implement the serialization and de-serialization logic in other applications.
FlexRadarIntro	Demonstrates how to implement the key features of the FlexRadar control.
FloatingBarChart	Demonstrates how to create a Floating bar chart and Gantt chart by customizing the series symbols.
LineMarkerExplorer	Demonstrates how to implement and use the line markers in FlexChart.
SunburstIntro	Demonstrates how to implement the key features of the sunburst chart.
Tooltips	Demonstrates how to customize the chart tooltips to display the plain as well as HTML content in the FlexChart and FlexPie controls.
WealthHealth	Demonstrates a dynamic chart based on Gapminder's Wealth and Health of Nations created using FlexChart.
WeatherChart	Demonstrates how to use range slider in FlexChart.

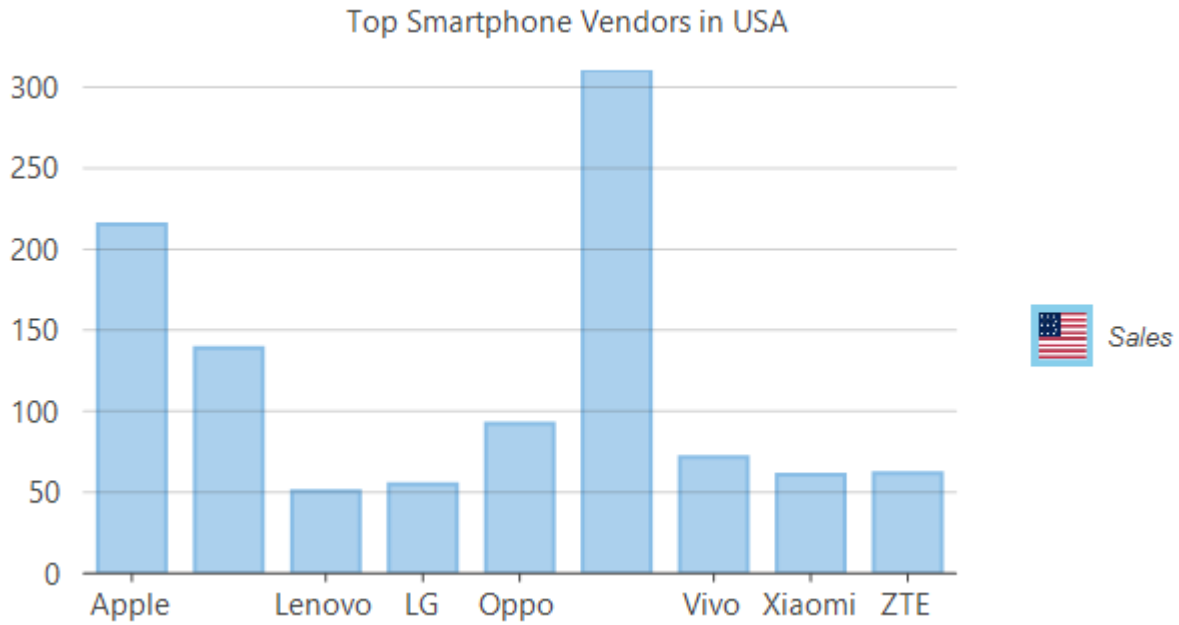
チュートリアル

This section discusses how to implement some real-world scenarios using FlexChart:

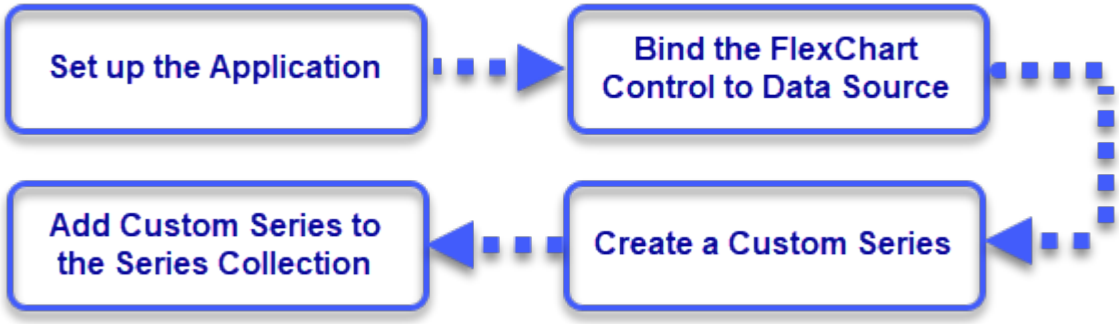
Topic	Snapshot	Content
Create Custom Legend Icon		Provides step-by-step guide on how to create a custom legend icon.
Create Chart Callouts		Provides step-by-step guide on how to create chart callouts.

カスタム凡例アイコンの作成

By default, the FlexChart control assigns color to each series and creates a legend icon based on the color assigned to the corresponding series. However, you can also create a custom legend icon. For instance, it would make it easier for user to read the data if country flags are displayed as legend icons while showing data for different countries.



To customize the legend icon once you have created a chart, you must create a custom series that inherits the Series class and **ISeries** interface. This interface provides the **GetLegendItemImageSource** method that lets you access the source of legend icon and hence customize the same by setting the flag image inside the legend icon box. This walkthrough takes you through the following detailed steps to create a custom legend icon.



Set up the Application

1. Create a new Windows Forms app.
2. Drag and drop the FlexChart control from the toolbox onto the form.
   Observe: A column type chart is drawn with a default data.

Bind the FlexChart Control to a Data Source

1. Create a data source.

CS | VB

```
```

2. Bind the FlexChart to this data source by setting the **DataSource** property.
3. Configure the X and Y axes by setting the **BindingX** and **Binding** property.
4. Configure the chart by setting the **ChartType** and other required properties.

CS | VB

```
```

Create a Custom Series

1. Create a series of custom type inherited from the **Series** class and **ISeries** interface. In this example, we have created a class named **SeriesWithPointLegendItems**.
2. Implement the **ISeries.GetLegendItemImageSource** method to access the size of the legend and to draw the icon using the Graphics.DrawImage method for the legend point.
3. You can also set the legend icon size to adjust according to the window size.

CS | VB

```
```

Add the Custom Series to Series Collection

1. Clear the default series getting displayed in the chart.
2. Create an object of the custom series class, **SeriesWithPointLegendItems** in this case.
3. Add this object to the chart **Series** collection using the Add method .

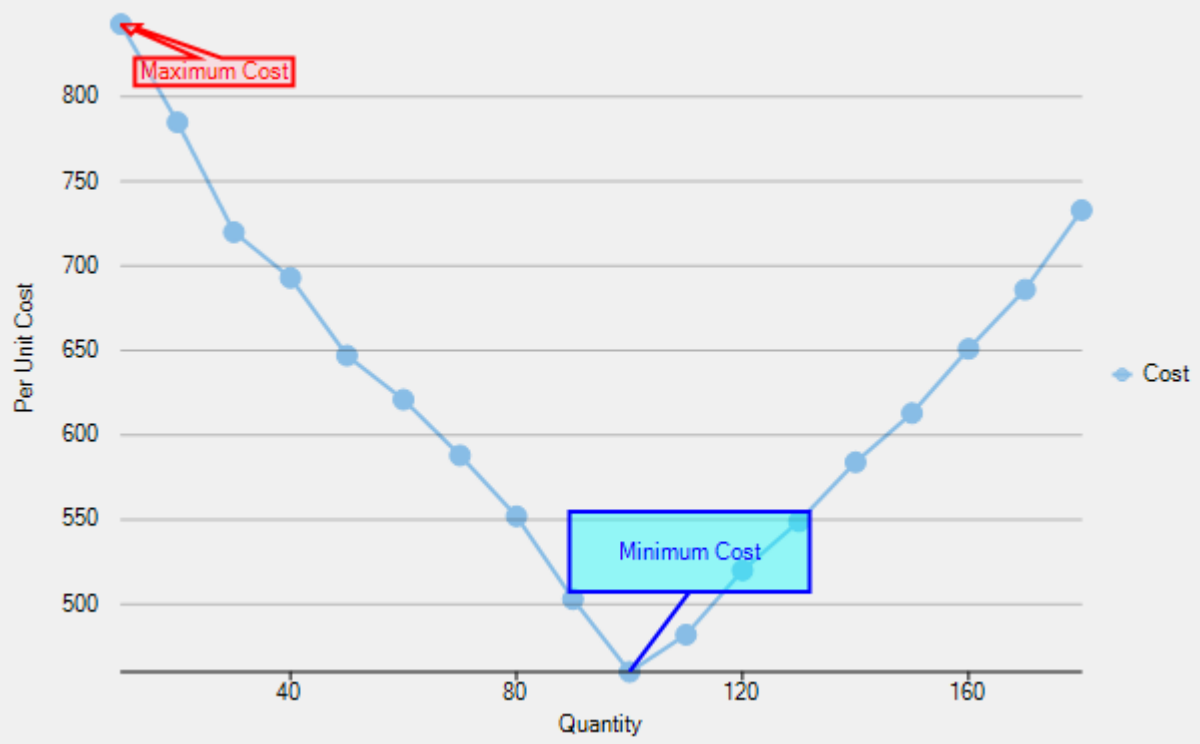
CS | VB

```
```

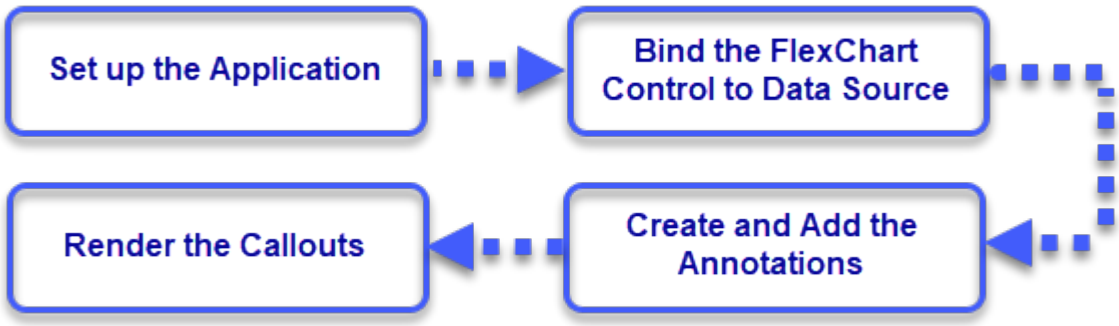
Run the application to observe that a country flag surrounded by border of series color appears as a custom legend icon in the chart. Similarly, multiple legend icons can be customized in the case of multiple series and charts. For detailed implementation of custom legend icons, see **FlexChartExplorer** sample which is shipped with the control. To see this feature in action, you can also download the **FlexChartExplorer** demo from our website.

チャート吹き出しの作成

Chart callouts are visual tools that help in emphasizing a specific series or data point on a chart, through a line or arrow from the point to be highlighted to a box with information text. These callouts not only supplement charts with additional information but also help in easy comprehension as they are directly connected to the point of emphasis. For instance, chart callouts can make it easy to indicate the values corresponding to maximum and minimum cost as shown in the example below.



In FlexChart, you can create chart callouts using the **Polygon** type annotations by carrying out the following simple steps. In this example, we have demonstrated two type of callouts, one with simple line connector and other one with the arrow connector.



Set up the Application

1. Create a new Windows Forms app.
2. Drag and drop the FlexChart control from the toolbox onto the form.
   Observe: A column type chart is drawn with a default data.

Bind the FlexChart Control to a Data Source

1. Create a data source.

   CS | VB

   ```
   ```

2. Bind the FlexChart to this data source by setting the DataSource property.
3. Clear the default series getting displayed in the chart and add a new series using the **Add** method.
4. Configure the X and Y axes by setting the BindingX and Binding property.
5. Configure the chart by setting the **ChartType** and other required properties.
6. Initialize the **Rendering** event of FlexChart to call the custom method **SetupAnnotations** which is implemented in the following steps to create the callouts.

   CS | VB

   ```
   ```

Create and Add the Annotations

1. Create an annotation layer by creating an instance of the **AnnotationLayer** class.
2. Create a line callout by creating an instance of the **Polygon** class.
3. Specify the points to create the line callout and set the related properties to attach and style the annotation.
4. Create a custom method, **GetArrowCalloutPoints** in this case, to measure the size of annotation text and calculate the coordinates for arrow callout annotation accordingly.
5. Create an arrow callout by creating another instance of the **Polygon** class.
6. Call the **GetArrowCalloutPoints** method and specify other related properties to attach and style the annotation.
7. Add the two annotations in the annotation layer using the **Add** method.

   CS | VB

   ```
   ```

Note that the custom method **GetArrowCalloutPoints** used in the step above to get the size of the annotation text and to calculate the polygon coordinates based on that can be implemented as follows.

Render the Callouts

1. Invoke the **SetupAnnotations** method in the **Rendering** event of the **FlexChart** class.

   CS | VB

2. Run the sample to render the chart with callouts.

Observe that a chart displaying a simple line callout and an arrow callout is displayed to indicate the data points related to minimum and maximum cost. Similarly, you can create callouts in the form of other polygons by measuring the size of text to be used and calculating the coordinates of the Polygon annotations accordingly. For detailed implementation, see **FlexChartExplorer** sample which is shipped with the control. To see this feature in action, you can also download the **FlexChartExplorer** demo from our website.