
ComponentOne

BarChart for ASP.NET Web Forms

ComponentOne, a division of GrapeCity

201 South Highland Avenue, Third Floor
Pittsburgh, PA 15206 USA

Website: <http://www.componentone.com>

Sales: sales@componentone.com

Telephone: 1.800.858.2739 or 1.412.681.4343 (Pittsburgh, PA USA Office)

Trademarks

The ComponentOne product name is a trademark and ComponentOne is a registered trademark of GrapeCity, Inc. All other trademarks used herein are the properties of their respective owners.

Warranty

ComponentOne warrants that the media on which the software is delivered is free from defects in material and workmanship, assuming normal use, for a period of 90 days from the date of purchase. If a defect occurs during this time, you may return the defective media to ComponentOne, along with a dated proof of purchase, and ComponentOne will replace it at no charge. After 90 days, you can obtain a replacement for the defective media by sending it and a check for \$2 5 (to cover postage and handling) to ComponentOne.

Except for the express warranty of the original media on which the software is delivered is set forth here, ComponentOne makes no other warranties, express or implied. Every attempt has been made to ensure that the information contained in this manual is correct as of the time it was written. ComponentOne is not responsible for any errors or omissions. ComponentOne's liability is limited to the amount you paid for the product. ComponentOne is not liable for any special, consequential, or other damages for any reason.

Copying and Distribution

While you are welcome to make backup copies of the software for your own use and protection, you are not permitted to make copies for the use of anyone else. We put a lot of time and effort into creating this product, and we appreciate your support in seeing that it is used by licensed users only.

Table of Contents

Cover Page	1
Cover Page Info	2
Overview	6
Help with ASP.NET Web Forms Edition	6
Key Features	7
Feature List	8
Function of bar graph	8
General function of chart	8
Function of the data type	8
Function of the data connection	8
Functionoftheshaft	8
Function of the axis scale	8
Function of axis labels	8
Function of the value labels	8
Function of the header footer	8
Function of legend	8
Function Tips	8
Function of animation	8
Function of style	8
Design-time features	8
Quick Start	9
Step 1 of 4: Creating an Application	9
Step 2 of 4: Adding a SeriesList to the Bar Chart	9-10
Step 3 of 4: Customizing the BarChart	10
Step 4 of 4: Running the Project	10-11
Design-Time Support	12
Smart Tag	12-13
Collection Editors	13
BarChartSeries Collection Editor	13-14
ChartStyle Collection Editor	14
BarChart Fundamentals	15
Inverted Bar Charts	15
Stacking Bar Charts	15-16
Special Bar Chart Properties	16-17

Bar Chart Elements	18
Axes	18-19
Axis Position	19
Axis Appearance	19
Axis Title and Rotation	19
Axis Tick Marks	19-20
Axis Grid Lines	20
Axis Bounds	20-21
Axes Annotation	21
Values Annotation	21
Chart Labels	21
Header and Footer	21-22
Legend	22
Series	22-23
Annotations	23-24
Data Binding	25
Export Service	26-29
BarChart Animation	30
Transition Effects	30
Animation Effect Duration	30
Samples	31
Task-Based Help	32
Axis Tasks	32
Rotating the Axis Label	32-33
Formatting the Axis Labels	33-34
Using the Origin to Display Positive and Negative Axes Values	34
Trendline	35-39
Binding the BarChart to an Access Data Source	39-40
Creating a Column Chart	40-41
Styling the Series	41-42
Client-Side Tasks	42-43
Creating a Bar Chart from an External Data Source from the Client Side	43-45
Client-Side Tutorials	46
Drilling Down in BarChart Data	46
Step 1 of 4: Set up your application	46-51
Step 2 of 4: Create the C1BarChart control	51-52

Step 3 of 4: Add script	52-56
Step 4 of 4: Run Your Application	56-57
Loading Data Conditionally	57-58
Step 1 of 4: Set Up Your Application	58
Step 2 of 4: Create the Server-Side Controls	58-59
Step 3 of 4: Add Script	59-61
Step 4 of 4: Run Your Application	61-63
Client-Side Reference	64
Using the Wijmo CDN	64-65

Overview

Add bar and column charts to your ASP.NET Web applications with **BarChart for ASP.NET Web Forms**. The Bar Chart draws each series as a bar or column of data and includes rich customization and animation.



Getting Started

To get started, review the following topics:

- [Key Features](#)
- [Quick Start](#)
- [Samples](#)

Help with ASP.NET Web Forms Edition

For information on installing **ComponentOne Studio ASP.NET Web Forms Edition**, licensing, technical support, namespaces and creating a project with the control, please visit [Getting Started with ASP.NET Web Forms Edition](#).

Key Features

BarChart for ASP.NET Web Forms provides the following unique key features:

- **HTML5 Charts**
What makes these charts superior is that they're powered by Scalable Vector Graphics (SVG). SVG is a perfect for creating rich and interactive visualizations in HTML5. Some benefits of charting with SVG include true client-side charting, DOM-friendly charts, hardware accelerated graphics, interactive animations and tooltips, and streaming visualizations. Plus, the charts require no additional plug-ins.
- **Column Charts**
Easily convert the bar chart to a column chart. It's as simple as setting the C1BarChart's Horizontal property to False.
- **Multiple Series**
Display multiple series on the bar chart. This is useful when showing two related series of data.
- **Stacked Bar Charts**
C1BarChart supports a stacked bar. To enable this feature, simply set the Stacked property to True.
- **Data Binding**
C1BarChart supports binding data from an external data source on the server. Enabling DataBinding is as simple as setting the DataSourceID or DataSource and DataBindings properties. Our Chart supports all of the standard ASP.NET DataSource Controls such as SqlDataSource and all standard DataSource types such as DataTable. DataBinding in our charts is very familiar and can be done at design time or in code.
- **Live Data**
All ComponentOne charts are optimized for streaming live data. The charts will animate as the data changes and show transitions to visualize the data in the data set.
- **Animation**
Add special effects to your bar chart. Enable transition settings, where on page load the chart may transition as a bounce, elastic, and so on.
- **Tooltips**
Associate tooltips and links with chart elements, so the chart responds to mouse events. This is done using an HTML image map, which is handled on the client.
- **DateTime**
You can format the data as DateTime format strings, for example. The chart automatically generates numeric annotation based on the data itself.
- **Appearance**
C1BarChart's rich set of properties allow you to change the look and feel of your chart. Modify its appearance with different colors for series and for items in a series, and shadows for the Chart elements. For a better UX, the position, size, and color are also adjustable.
- **Automatic Axis**
C1BarChart automatically creates the chart axes.
- **Axis Origin**
The C1BarChart control supports negative values and can display them nicely using the Origin property for each axis. The Origin property tell the axis where to draw its starting point. This is optimal for displaying positive and negative values on the same chart.
- **Labels**
Increase the readability of the bar chart with labels. Highlight an important data point, or provide information on data or on the chart.
- **Full Cross-browser Compatibility**
Ensure your UI works in every browser and every device without worrying about compatibility issues. Create interactive charts that render just as well in IE6 as they do on an iPad. ComponentOne ASP.NET Charts support these popular browsers: IE6+, Firefox 3+, Safari 3+, and Chrome.

Feature List

Function of bar graph

General function of chart

Function of the data type

Function of the data connection

Function of the shaft

Function of the axis scale

Function of axis labels

Function of the value labels

Function of the header footer

Function of legend

Function Tips

Function of animation

Function of style

Design-time features

Quick Start

The BarChart Quick Start describes how to get started with the ASP.NET control, **BarChart**. In this quick start, you will create an ASP.NET application containing one **C1BarChart** control, add data to C1BarChart, and customize C1BarChart.

Step 1 of 4: Creating an Application

In this topic you will add a **C1BarChart** control.

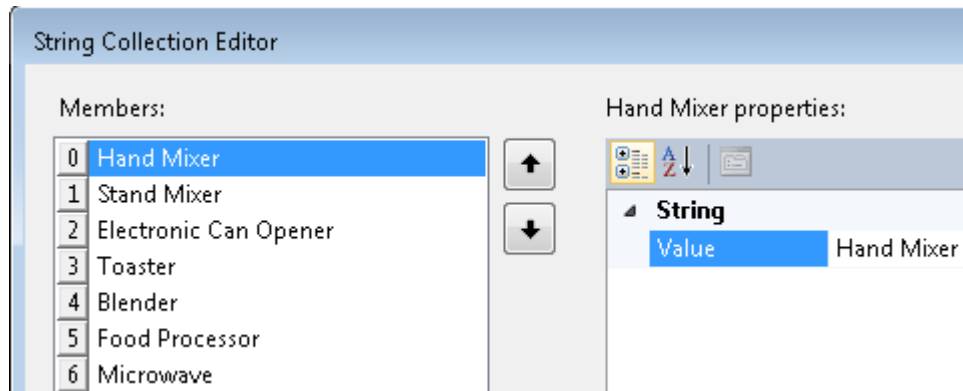
1. Begin by creating an ASP.NET Web application. Note that if using Visual Studio 2008, you must add a **ScriptManager** control to the form. If using Visual Studio 2005, the **ScriptManager** control is automatically added to the form.
2. Add the following references to your project:
 - o C1.Web.Wijmo.Controls.4.dll
 - o C1.Web.Wijmo.Controls.Design.4.dll
 - o C1.C1Report.4.dll
3. Add the controls to the Toolbox.
 1. Open the Visual Studio IDE (Microsoft Development Environment). Make sure the Toolbox is visible (select **Toolbox** in the **View** menu if necessary), and right-click it to open the context menu.
 2. To make the Studio for ASP.NET Web Forms components appear within a tab in the Toolbox, select **Add Tab** from the context menu and type in the tab name (for example, Studio for ASP.NET Web Forms).
 3. Right-click the tab where the component is to appear and select **Choose Items** from the context menu. The **Choose Toolbox Items** dialog box opens.
 4. In the dialog box, select the **.NET Framework Components** tab. Sort the list by Assembly Name (click the **Assembly Name** column header) and check the check boxes for all components corresponding to Assembly Name **C1.Web.Wijmo.Controls.x**.
 5. Click **OK** to close the dialog box. The controls are added to the Visual Studio Toolbox.
4. Add a new Web Form to your project. Right-click the project name in the Solution Explorer and select **Add | New Item**. Within the Add New Item dialog box select **Web Form** from the list of templates. Provide a name to the form.
5. While in the Design view, navigate to the Visual Studio Toolbox and double-click the **C1BarChart** control. This adds BarChart to your form.

Notice in the Source view of your form, markup for C1BarChart gets added within `<div></div>` tags inside the `<body></body>` tags.

Step 2 of 4: Adding a SeriesList to the Bar Chart

In this step, you will date time values to the **X BarChartSeries** and double values to the **Y BarChartSeries**. You will also modify the series symbols and label.

1. Click the **BarChart** smart tag and select **SeriesList** from the **C1BarChart Tasks** menu. The **BarChartSeries Collection Editor** appears.
2. In the **BarChartSeries Collection Editor**, click the **Add** button to add a new **BarChartSeries** to the BarChart.
3. Select **Data -> X** from the properties pane in the **BarChartSeries Collection Editor** and click the ellipsis button next to the **Values** property to open the **ChartXData Collection Editor**.
4. Click **Add** seven times to add seven members and enter the following values for each string: "Hand Mixer", "Stand Mixer", "Electronic Can Opener", "Toaster", "Blender", "Food Processor" and "Microwave".



5. Click **OK** to save and close the **String Collection Editor**.
6. Select **Data -> Y** from the properties pane in the **BarChartSeries Collection Editor** and enter the following data next to the **DoubleValues** property: 50, 300, 40, 60, 130, 250, 500.
7. Set **LegendEntry** to **False** and click **OK** to save and close the **BarChartSeries Collection Editor**.

Step 3 of 4: Customizing the BarChart

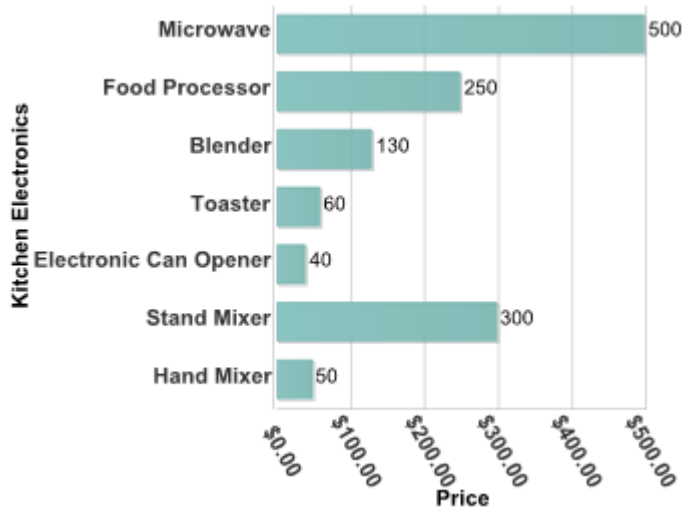
In this topic you will customize the axis annotation.

1. In design view select the Bar Chart and expand the **Axis -> Y -> Labels -> AxisLabelStyle** and set the **Rotation** property to **60**.
2. Set the **AnnoFormatString** property to **c** for the Y-Axis.
This will change the y-axis annotation to currency.
3. Expand **Axis -> X** and set the **Max** property to **500**.
4. In the **C1BarChart** properties window expand **Axis -> X -> Labels -> AxisLabelStyle** and set **FontSize** to **11**.
5. In the **C1BarChart** properties window expand **Axis -> Y -> Labels -> AxisLabelStyle** and set **FontSize** to **11**.
6. In the **C1BarChart** properties window expand **Axis -> Y** and set **Text** to **Price**.
7. In the **C1BarChart** properties window expand **Axis -> Y -> TextStyle -> Fill** and set color to **Black**.
8. In the **C1BarChart** properties window expand **Axis -> Y -> TextStyle** and set **FontSize** to **11**.
9. In the **C1BarChart** properties window expand **Axis -> X** and set **Text** to **Kitchen Electronics**.
10. In the **C1BarChart** properties window expand **Axis -> X -> TextStyle** and set **FontSize** to **11**.
11. In the **C1BarChart** properties window expand **Axis -> X -> TextStyle -> Fill** and set color to **Black**.
12. In the **C1BarChart** properties window expand **ChartLabelStyle** and set **FontSize** to **10**.
13. Set the BarChart **Width** property to **400** and BarChart **Height** to **300**.
14. In the **C1BarChart** properties set the **ClusterWidth** to **70**.

Step 4 of 4: Running the Project

Press **F5** to run the project and view the following:

- The Y-Axis annotation is rotated to a 60 degree angle using the **BarChart.Axis.X.Labels.Style.Rotation** property.
- The default font size was modified for the chart elements.



Users can set this view that causes the issue. To prevent rendering in quirks mode, you can force the page to render with the latest browser. Add the following meta tag to the header of the page:

To write code in Source View

```
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1"/>
```

Design-Time Support

C1BarChart provides customized context menus, smart tags, and a designer that offers rich design-time support and simplifies working with the object model.

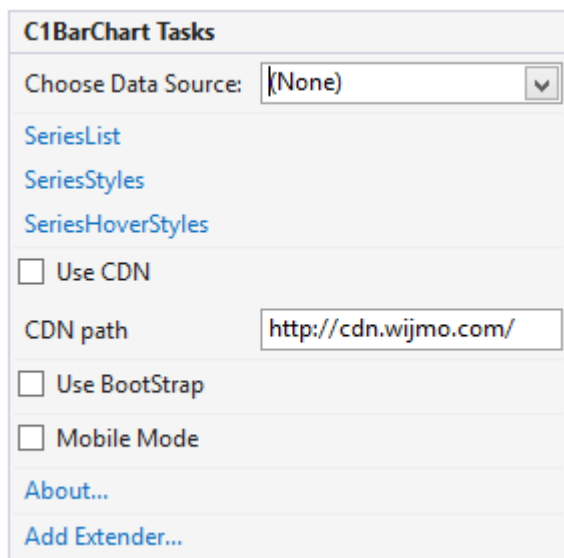
The following sections describe how to use **C1BarCharts** design-time environment to configure the **C1BarChart** control.

Smart Tag

In Visual Studio, the **C1BarChart** control includes a smart tag. A smart tag represents a short-cut tasks menu that provides the most commonly used properties in **C1BarChart**.

The **C1BarChart** control provides quick and easy access to common properties through its smart tag.

To access the **C1BarChart Tasks** menu, click on the smart tag (🔗) in the upper-right corner of the **C1BarChart** control. This will open the **C1BarChart Tasks** menu.



The **C1BarChart Tasks** menu operates as follows:

Choose Data Source:

Clicking on the **Choose Data Source** item opens a drop-down list where you can choose an existing data source or select a new data source to bind to.

SeriesList

When the **SeriesList** item is selected it opens the **BarChartSeries Collection Editor** dialog box where you can add or remove **BarChartSeries** members to the **C1BarChart** control. Once a **BarChartSeries** member is added, you can modify its properties.

SeriesStyles

Clicking the **SeriesStyles** item opens up the **ChartStyle Collection Editor** dialog box where you can add or remove **ChartStyle** members to the **C1BarChart** control. Once a **ChartStyle** member is added you modify its properties.

SeriesHoverStyles

Clicking the **SeriesHoverStyle** item opens the **ChartStyle Collection Editor** dialog box where you can add or remove **ChartStyle** members to the **C1BarChart** control. Once a **ChartStyle** member is added you modify its properties.

Use CDN

When the **Use CDN** checkbox is selected it loads the client resources from CDN. This is not selected by default.

CDN path

Displays the url path of the CDN.

Use Bootstrap

Selecting the **Use Bootstrap** option allows you to apply Bootstrap theming to your control. To find more information on using Bootstrap theming in your application, see [Bootstrap theming](#).

About

Clicking on the **About** item displays a dialog box, which is helpful in finding the version number of **ASP.NET Web Forms Edition** and online resources.

Collection Editors

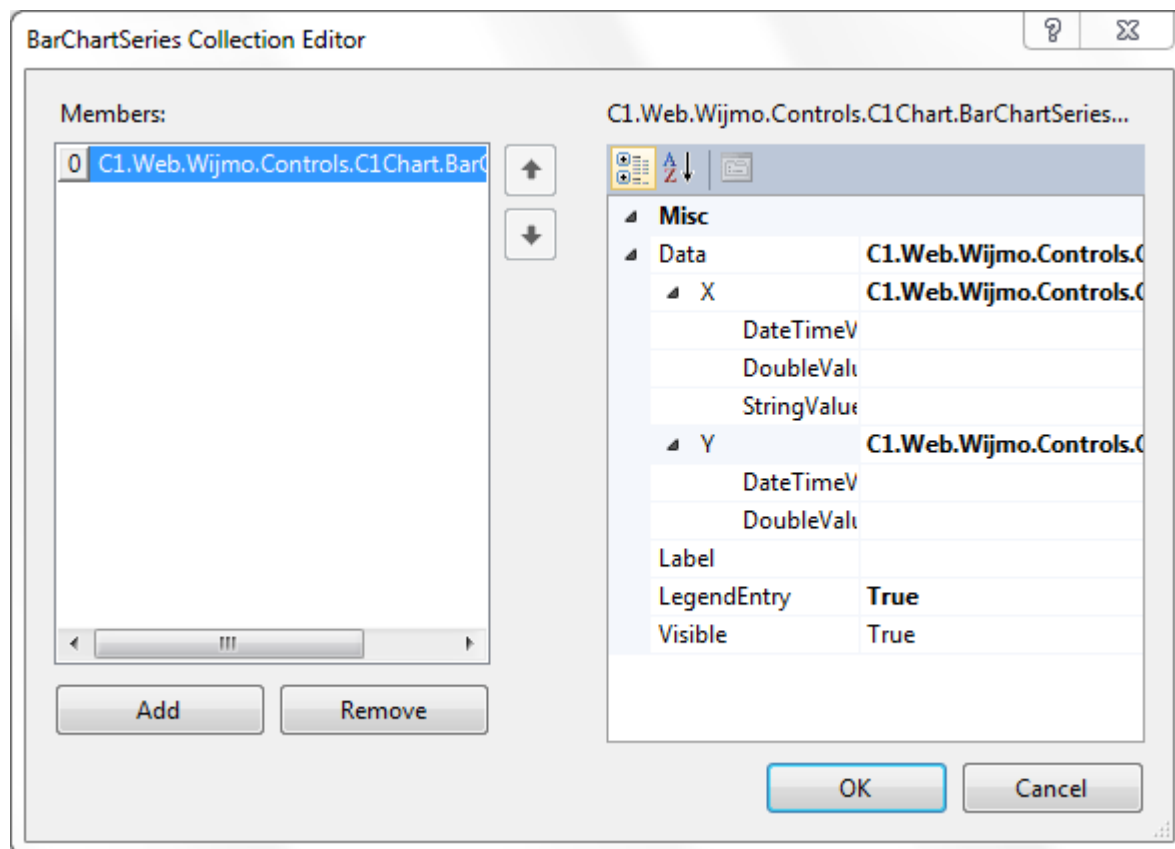
[C1BarChart](#) includes the following collection editor to add/remove selected and disabled dates:

- **BarChartSeries Collection Editor**
- **ChartStyle Collection Editor**

BarChartSeries Collection Editor

The **BarChartSeries Collection Editor** allows the user to add/remove selected add or remove **BarChartSeries** members to the **C1BarChart** control. Once a BarChartSeries member is added you can modify its properties.

The **BarChartSeries Collection Editor** appears like the following once you click the **Add** button to add a new member to the series:

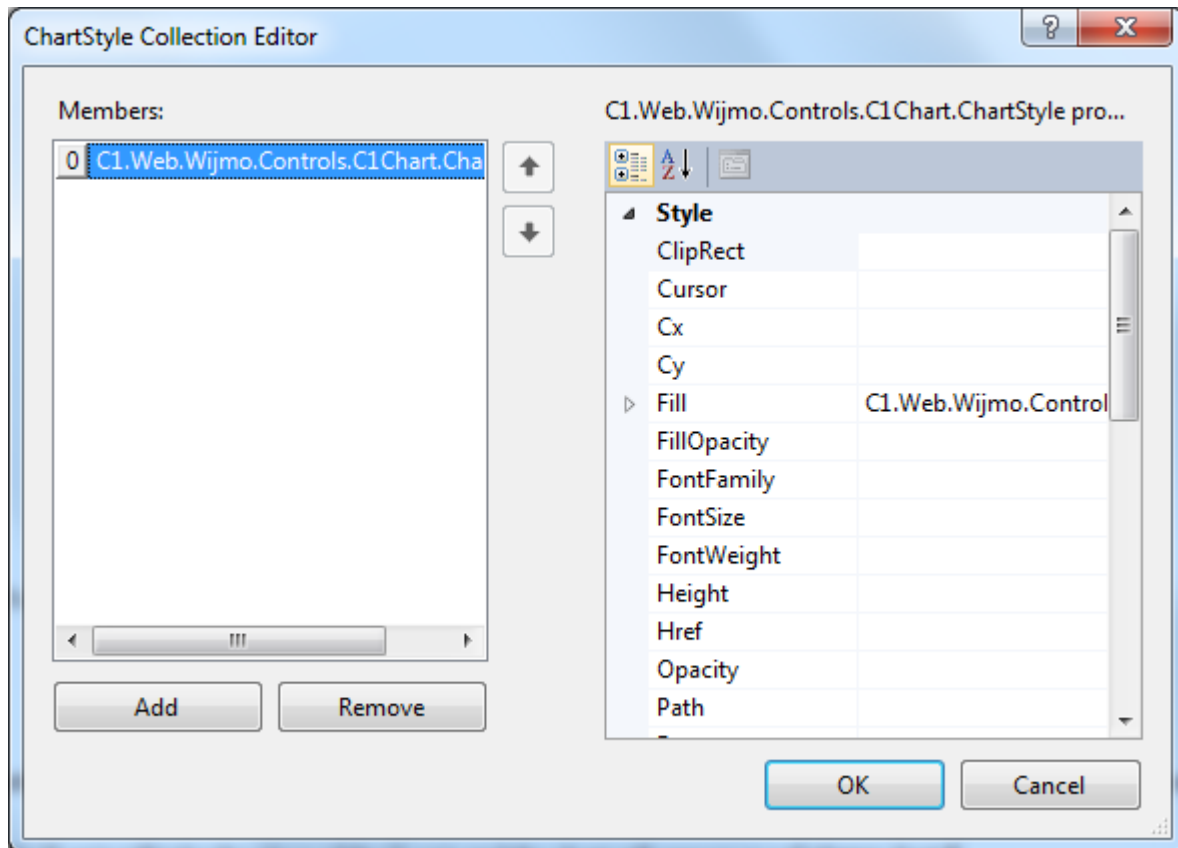


To access the **BarChartSeries Collection Editor**:

1. Select the **C1BarChart** control and click on its smart tag.
2. In the **C1BarChart** tasks menu, select the **SeriesList** item and this will open the **BarChartSeries Collection Editor** dialog box.

ChartStyle Collection Editor

The **ChartStyle Collection Editor** allows the user to add or remove **ChartStyle** members to the **C1BarChart** control. Once a **ChartStyle** member is added you modify its properties.



To access the **ChartStyle Collection Editor**:

1. Select the **C1BarChart** control and click on its smart tag.
2. In the **C1BarChart** tasks menu, select the **SeriesStyle** item and this will open the **ChartStyle Collection Editor** dialog box.

BarChart Fundamentals

A **Bar** chart is an inverted column chart where the category axis is the vertical axis. A Bar/Column chart draws each series as a bar in a cluster. The number of clusters is the number of points in the data. Each cluster displays the nth data point in each series.

Inverted Bar Charts

A Bar chart is an inverted **Column** chart in the sense that the X and Y axes are reversed.

When a data set contains X or Y values which span a large range, sometimes the normal chart setup does not display the information most effectively. Formatting a chart with a vertical Y-axis and axis annotation that begins at the minimum value can sometimes be more visually appealing if the chart could be inverted. Therefore, [C1BarChart](#) provides the [Horizontal](#) property of the C1BarChart.

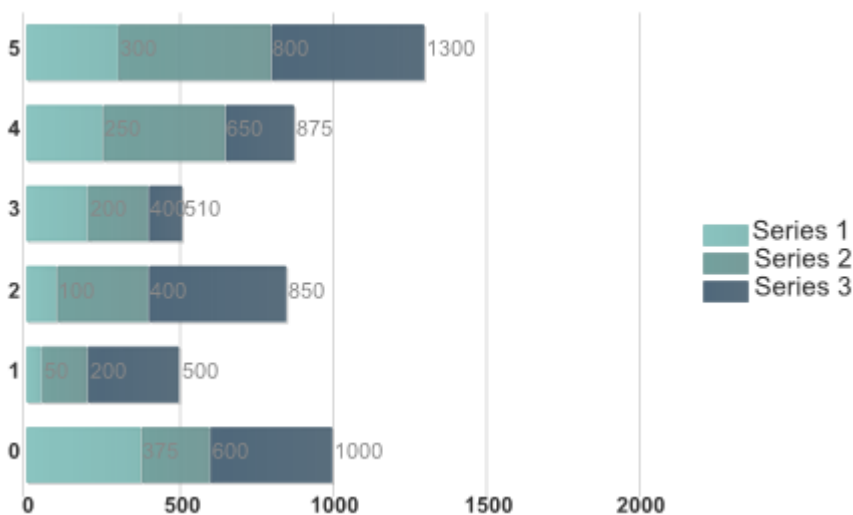
Setting the [Horizontal](#) property of the C1BarChart to **False** will revert the axes. This means that the Y-axis will take the place of the X-axis, and the X-axis will take the place of the Y-axis and the chart will appear as a **Column** chart.

Stacking Bar Charts

A Stacking Bar chart draws each series as a portion of a stacked bar cluster, the number of clusters being the number of points in the data. Each bar displays the nth data point in each series. A Bar chart or Column chart can be stacked by setting the [Stacked](#) property to True.

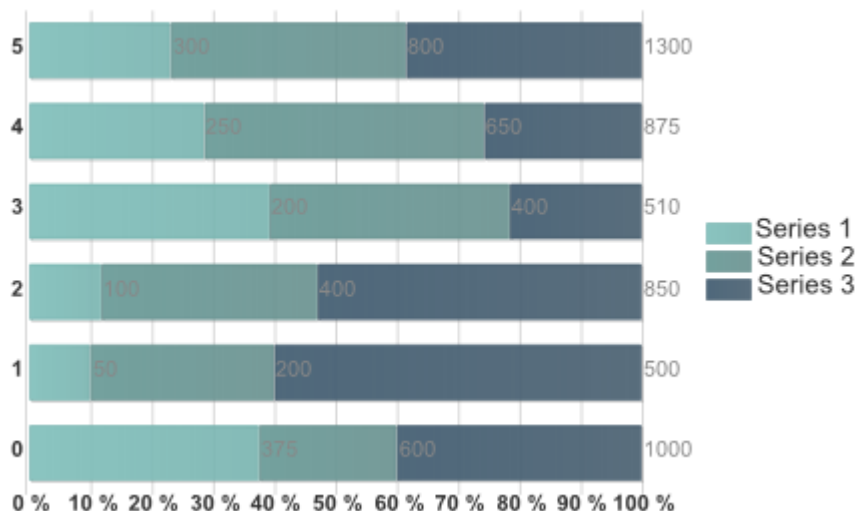
Stacked Bar Chart

To display a Stacked Bar chart, set the [Stacked](#) property to true.



Stacked Percentage Chart

To display a stacked percentage chart, set the [Is100Percent](#) to true and [Stacked](#) to true.

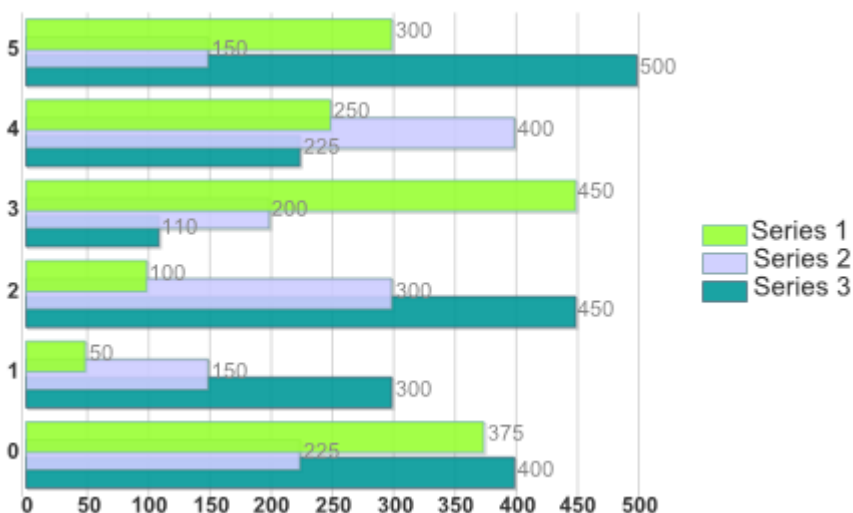


Special Bar Chart Properties

A Bar chart draws each series as a bar in a cluster. The sizing and spacing of the clusters for Bar and Stacking Bar charts can be customized.

Cluster Overlap

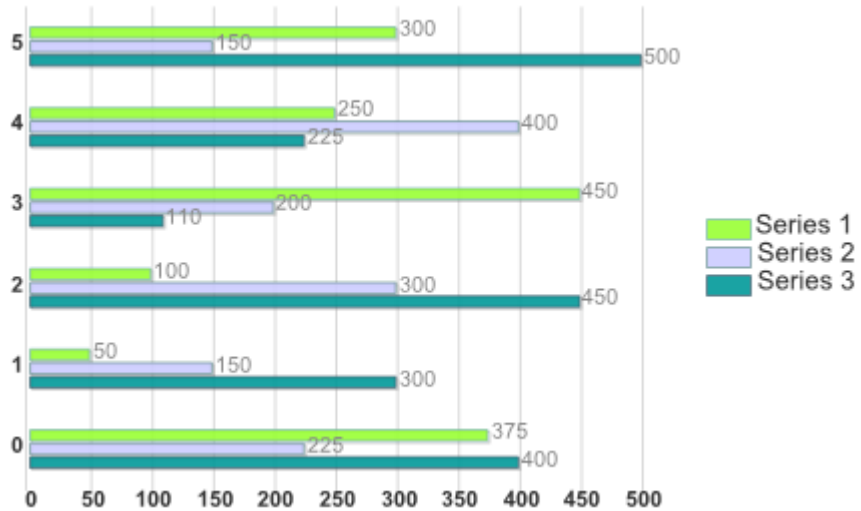
Use the [ClusterOverlap](#) property to set the amount that bars overlap each other in a cluster. The value represents the percentage of bar overlap, with valid values between 0 and 100. The following figure illustrates a bar chart with a [ClusterOverlap](#) of 50 percent:



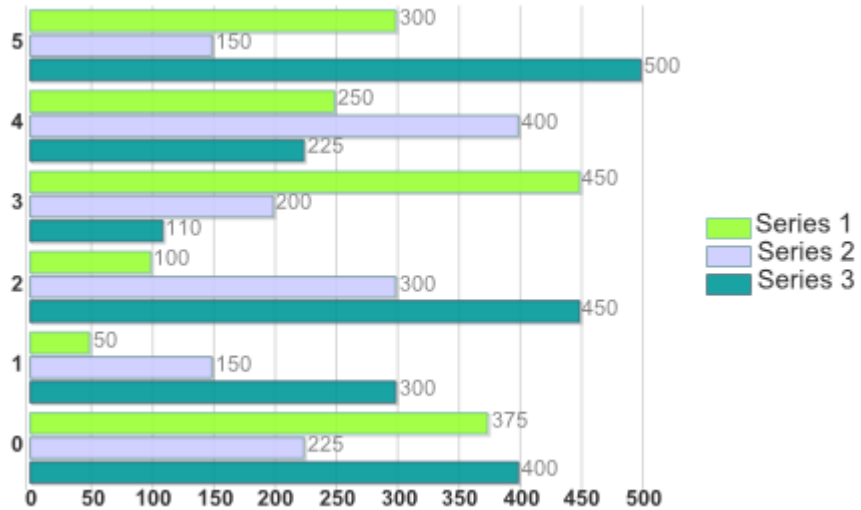
Cluster Width

Use the [ClusterWidth](#) property to set the space used by each bar cluster. The value represents the percentage of available space, with valid values between 0 and 100.

ClusterWidth = 50%

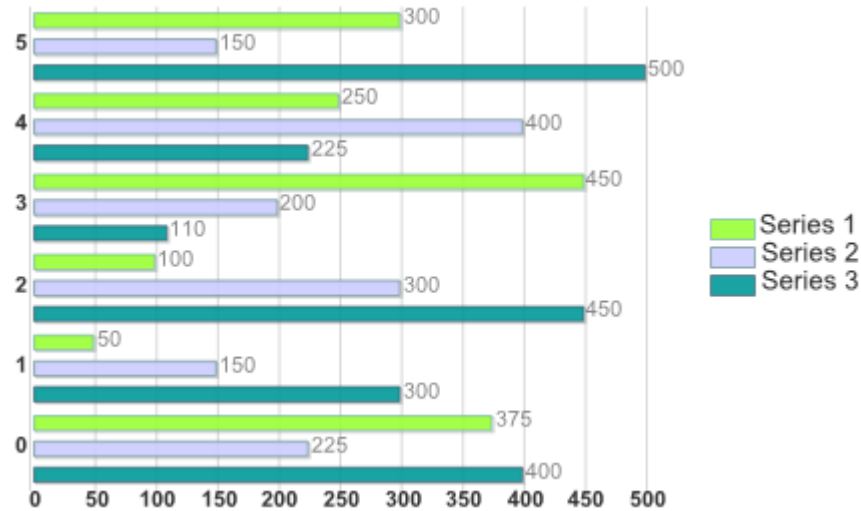


ClusterWidth = 90%



Cluster Spacing

Use the [ClusterSpacing](#) property to indicate the spacing between the adjacent bars. The following Bar chart shows the [ClusterSpacing](#) property set to 10.



Bar Chart Elements

This section provides a visual and descriptive overview of the elements that comprise the **C1BarChart** control. The topics are categorized into distinct elements that represent different aspects of the **C1BarChart** control.

Axes

The X and Y properties of the [ChartAxes](#) object returns [ChartAxis](#) objects that allow you to customize the appearance of the chart axes. The axes are represented by sub-properties of the ChartAxes property: [X](#) and [Y](#). Each of these properties returns a ChartAxis object with the following main properties:

Axis Property	Description
Alignment	A value that indicates the alignment of the axis text. The default value for this property is ChartAxisAlignment.Center .
AnnoFormatString	A value that indicates the format string of annotation.
AnnoMethod	A value that indicates the method of annotation. The default value for this property is ChartAxisAnnoMethod.Values .
AutoMajor	A value that indicates whether the major tick mark values are calculated automatically. The default value for this property is true.
AutoMax	A value that indicates whether the maximum axis value is calculated automatically.
AutoMin	A value that indicates whether the minimum axis value is calculated automatically. The default value for this property is True.
AutoMinor	A value that indicates whether the minor tick mark values are calculated automatically. The default value for this property is true.
Compass	A value that indicates the compass of the axis. The default value for this property is ChartCompass.South .
GridMajor	A value that provides information for the major grid line.
GridMinor	A value that provides information for the minor grid line.
Labels	A value that provides information for the labels.
Max	A value that indicates the maximum value of the axis. The default value for this property is 0.
Min	A value that indicates the minimum value of the axis. The default value for this property is 0.
Origin	A value that indicates the origin value of the axis.
Style	A value that indicates the style of the axis.
Text	A value that indicates the text of the axis.
TextStyle	A value that indicates the style of text of the axis.
TextVisible	A value that indicates the visibility of the axis text. The default value for this property is True.
TickMajor	A value that provides information for the major tick.
TickMinor	A value that provides information for the minor tick.
UnitMajor	A value that indicates the units between major tick marks. The default value for this property is 0.
UnitMinor	A value that indicates the units between minor tick marks. The default value for this property is 0.
ValueLabels	A value that shows a collection of valueLabels for the axis.

Visible

A value that indicates the visibility of the axis. The default value for this property is True.

Axis Position

Axis annotation typically appears beside its axis. This may be a problem on charts with an origin that is not at the axis minimum or maximum. The chart can automatically determine where to place annotation in different situations, depending on the chart type. The [Compass](#) property can also specify annotation placement for an axis. The [Compass](#) value for the X-axis can be set to either North or South, while the value for a Y-axis can be set to either East or West. By default, the X-axis is set to South and the Y-axis is set to West.

Axis Appearance

The [Alignment](#) property can be set to three different settings: **Center**, **Near**, or **Far**. Setting the alignment to center centers the axis title in comparison to the chart area. Setting the alignment to Near places the axis title to the left side of the chart area. Setting the alignment to Far places the axis title to the right side of the chart area.

To modify the X-Axis labels at design time, expand the **Axis->X->Labels->AxisLabelStyle** and set the **FontSize** property.

The **Fill.Color** property changes the color of the axis line, tick marks, label, and title. For example, to modify the X-Axis labels font color at design time, expand the **Axis->X->Labels->AxisLabelStyle->Fill** and then click the ellipsis button next to **Color** and select a color.

The following example shows the source view for the **Color** property after it has been set:

To write code in Source View

```
<Axis>
  <X Max="2010" AutoMin ="false" Min="2005" Text="Year">
    <TextStyle FontSize="16">
      </TextStyle>
    <Labels>
      <Style FontSize="12">
        <Fill Color="#0033CC">
          </Fill>
        </Style>
      </Labels>
      <TickMajor Position="Outside">
        </TickMajor>
      </X>
    </Axis>
```

Axis Title and Rotation

Adding a title to an axis clarifies what is charted along that axis. The title or the annotation along the axis can also be rotated.

Adding an Axis Title

Use the axis **Text** property to add a title to an axis. To remove the title, delete the text from the Text property.

Rotating Axis Title

Use the **Rotation** property to rotate the axis title to 90, 180, or 270 degrees. The 90 and 270-degree rotations are most efficient for vertical axes. To modify the rotation for the X-Axis labels at design time, expand the **Axis->X->Labels->AxisLabelStyle** and set the **Rotation** property.

Axis Tick Marks

Tick marks are the lines that appear perpendicular to the axis to display the units of measurement in the chart. Major and minor tick marks can be shown on the bar chart when the **ChartAxisTick.Position** is set to **Cross** or **Outside**. Customizing the tick spacing or attributes is as easy as manipulating a set of properties.

The [TickMajor](#) and [TickMinor](#) properties set the state of the Axis' tick marks. This property can be set to any of the [ChartAxisTickPosition](#) values.

Tick Mark Position

These values set where and if the tick marks will be displayed:

Value	Description
ChartAxisTickPosition.None	No tick marks along axis.
ChartAxisTickPosition.Cross	Tick marks cross over axis.
ChartAxisTickPosition.Outside	Tick marks located outside chart area on axis.
ChartAxisTickPosition.Inside	Tick marks located inside chart area on axis.


Tick Mark Spacing

The [AutoMajor](#) and [AutoMinor](#) properties set whether the tick marks are to be automatically configured by the chart. When both these properties are set to **True**, the chart uses the current data to logically place the major and minor tick marks. When the [AutoMajor](#) property is true, its not necessary to enable overlap for axis annotations.

The [UnitMajor](#) and [UnitMinor](#) properties set the units by which the ticks will be spaced. When the [UnitMajor](#) property is set, the [UnitMinor](#) property is automatically set by the chart to half the [UnitMajor](#) value. Although the chart automatically sets the [UnitMinor](#) property, it also can be manually changed to a different value.

Tick Mark Length

You can increase the length of the Major and Minor tick marks using the [Factor](#) property. Before using the [Factor](#) property set the [Position](#) to **Outside** or **Cross**. Tick marks are sized based on the thickness of the axis line as well as the tick factor. Doubling the tick factor doubles the length of the axis tick mark. If negative values are used for the X-axis tick marks, the tick marks will appear above the x-axis labels. If negative values are used for the y-axis tick marks, the tick marks will appear to the left of the y-axis labels.

 **Note:** When increasing the value **ChartAxisTick.Factor** property, the appropriate **MarginBottom**, **MarginLeft**, **MarginRight**, or **MarginTop** property should also be increased for adequate space for the increased length of the tick marks.

Axis Grid Lines

Grid lines are lines that appear perpendicular with major/minor tick marks at unit major/minor intervals. The lines that appear perpendicular to an axis at Major intervals are controlled by the [GridMajor](#) property and the lines that appear perpendicular to an axis at Minor intervals are controlled by the [GridMinor](#) property. Grid lines can help improve the readability of the Chart when you are looking for exact values.

Axis Bounds

Normally a graph displays all of the data it contains. However, a specific part of the chart can be displayed by fixing the axis bounds.

The chart determines the extent of each axis by considering the lowest and highest data value and the numbering increment. Setting the [Min](#), [Max](#), [AutoMin](#), and [AutoMax](#) properties allows the customization of this process.

Axis Min and Max

Use the [Min](#) and [Max](#) properties to frame a chart at specific axis values. If the chart has X-axis values ranging from 0 to 100, then setting [Min](#) to 0 and [Max](#) to 10 will only display the values up to 10.

The chart can also calculate the [Min](#) and [Max](#) values automatically. If the [AutoMax](#) and [AutoMin](#) properties are set to **True** then the chart automatically formats the axis numbering to fit the current data set.

Axes Annotation

The annotation along each axis is an important part of any chart. The chart automatically produces the most natural annotation possible, even as chart data changes.

The following properties represent the format and the layout for the annotation of the axes in `C1BarChart`:

Axis Property	Description
AnnoFormatString	A value that indicates the format string of annotation.
AnnoMethod	A value that indicates the method of annotation.
ValueLabels	A value that shows a collection of valueLabels for the axis.

Values Annotation

Values Annotation is an implementation where the chart automatically generates numeric annotation based on the data itself. Values Annotation can be used for any axis, with any chart type, and with any data layout. It is controlled by the following properties of the axis:

Property	Description
AnnoFormatString	A value that indicates the format string of annotation.

Chart Labels

The chart label in a Bar chart represents the label that displays the x, y value for the **C1BarChart** inside or outside each set of data points.

Chart labels are useful when highlighting an important data point, but can also be used generally to provide information on data or on the chart.

Chart labels automatically appear inside the data point when you specify the x, y values. The `ChartLabels` can be hidden by setting the **ShowChartLabels** property to `False`.

Chart Labels Formatting

The chart labels can be formatted using the **ChartLabelFormatString** property.

Chart Labels Appearance

You can customize the appearance of the chart labels using the **ChartLabelStyle** property.

Header and Footer

The header and footer elements are used to display descriptive information about the chart. They are controlled by the **Header** and **Footer** properties.

The chart header and footer properties return a [ChartTitle](#) object that contains the following main properties:

Property	Description
Compass	A value that indicates the compass of the title. This determines the position of the title: North (top of chart), South (bottom of chart), East (right of chart), and West (left of chart).
Style	Contains properties that set the font, orientation, colors, and border of the title.
Text	Determines the position of the title.
TextStyle	A value that indicates the style of the title text.
Visible	Determines whether the title is visible.

C1Chart sizes and positions the titles automatically, based on their contents and how the [Compass](#) property is set.

Customizing header and footer elements

The header and footer elements' text and alignment, position, border, colors, and font can be customized using the **ChartTitle's** properties.

The Header element can be moved to the left (negative value) or right (positive value) using the **X** property and to the top (positive value) or bottom (negative value) using the **Y** property.

Legend

The legend element displays information about each data series of the chart. The chart legend displays the mapping between the physical colors and the data series.

C1BarChart automatically generates a **Legend** whenever data exists in the chart and if the [LegendEntry](#) property is enabled. The name of each series is represented in the legend when the [Label](#) property is specified. If there is no value specified in the [Label](#) property then the series names appear as undefined in the legend.

The legend is controlled by the **Legend** property, which returns a [ChartLegend](#) object with the following main properties:

Property	Description
Text	Contains text displayed in the legend title.
Style	Contains properties that set the font, orientation, colors, and border of the legend.
Compass	Determines the position of the legend.
Visible	Determines whether the legend is visible.
Orientation	Determines whether the legend items should be displayed in the horizontal or vertical direction.

C1Chart sizes and positions the legend automatically, based on its contents and the [Compass](#) and [Orientation](#) properties.

The legend's orientation can be horizontal or vertical through its [Orientation](#) property and its position can be north, south, east, or west through its [Compass](#) property.

Series

The [BarChartSeries](#) object represents the data being plotted for the Bar chart. You can have one or more BarChartSeries on the C1BarChart. Each BarChartSeries is represented in a different color. The BarChartSeries can be added in design view through the **BarChartSeries Collection Editor**, in source view through the BarChartSeries element, or programmatically through the [BarChartSeries](#) object.

Adding a BarChartSeries in Design View:

1. Select the **C1BarChart** control and click on its smart tag.
2. In the **C1BarChart** tasks menu, select the **SeriesList** item and this will open the **BarChartSeries Collection Editor** dialog box.
3. Click **Add** to add a **BarChartSeries** member to the **SeriesList** collection.

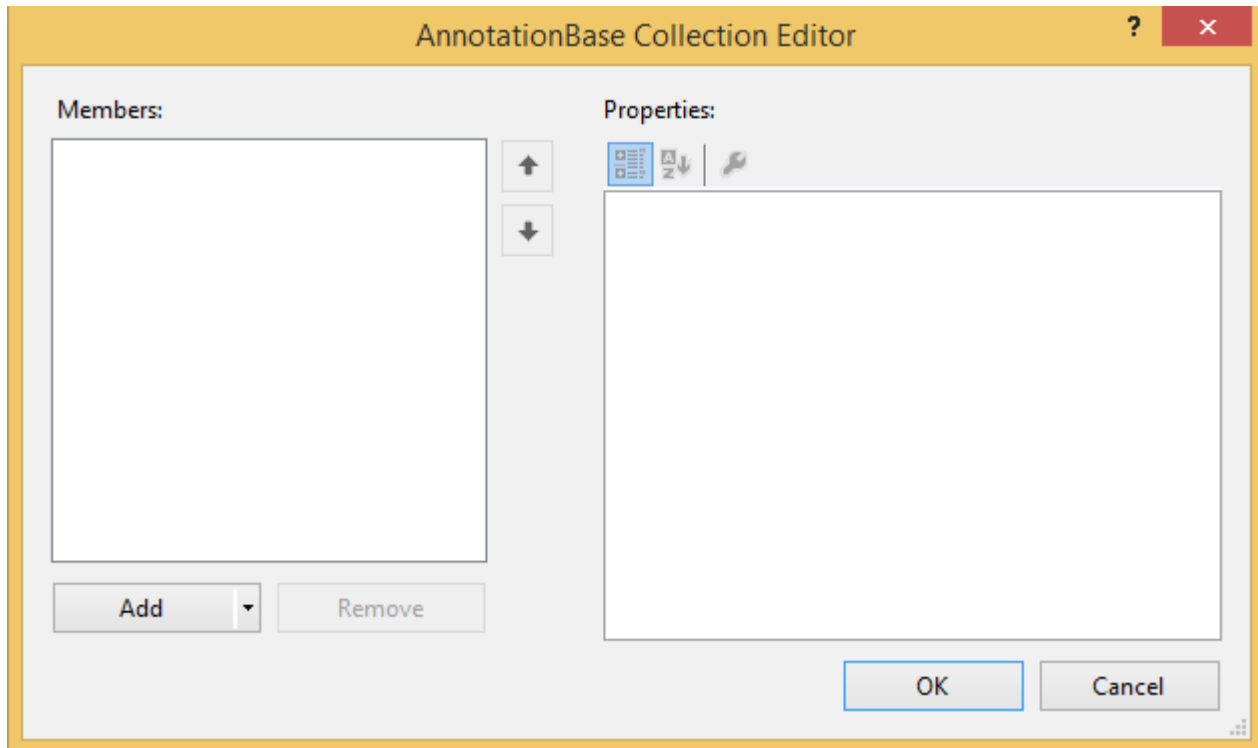
Adding a BarChartSeries in Source View:

To write code in Source View

```
<SeriesList>
<cc1:BarChartSeries Label="West" LegendEntry="true">
  <Data>
    <X>
      <Values>
        <cc1:ChartXDataStringValue="Desktops"/>
        <cc1:ChartXDataStringValue="Notebooks"/>
        <cc1:ChartXData StringValue="AIO"/>
        <cc1:ChartXDataStringValue="Tablets"/>
        <cc1:ChartXDataStringValue="Phones"/>
      </Values>
    </X>
    <Y>
      <Values>
        <cc1:ChartYData DoubleValue="5"/>
        <cc1:ChartYData DoubleValue="3"/>
        <cc1:ChartYData DoubleValue="4"/>
        <cc1:ChartYData DoubleValue="7"/>
        <cc1:ChartYData DoubleValue="2"/>
      </Values>
    </Y>
  </Data>
</cc1:BarChartSeries>
```

Annotations

The [Annotations](#) property enables you to add annotation in your C1BarChart. Use the AnnotationBase Collection Editor to add and customise annotations in your chart.



Data Binding

The **C1BarChart** supports binding data from an external data source on the server. DataBinding is allowed by setting the DataSourceID or the DataSource and DataBindings. The following properties are used to bind X and Y values to the specified data field:

- DataSourceID
- DataBindings
- C1BubbleChartBinding.XField
- C1BubbleChartBinding.XFieldType
- C1BubbleChartBinding.YField
- C1BubbleChartBinding.YFieldType
- C1BubbleChartBinding.Y1Field

DataBindings is a collection that contains the C1BarChartBindings instances. C1BarChartBinding includes the following properties:

- DataMember - This property is used for specifying the name of the list of data if the data source contains more than one list.
- HintField - This property is used for binding Hint content to the specified field name. If the HintField is set, then moving the mouse to a series will show the hint value that has the same index with the series.

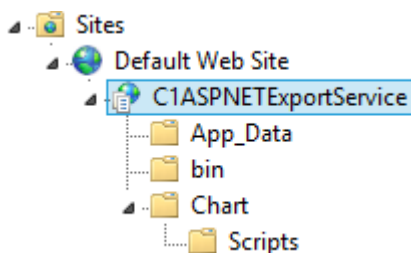
Export Service

Use the C1 ASP.NET Export Service, to export Charts as images or PDF, without creating a complex export application. This service resides on the application server.

The advantages of using C1 ASP.NET Export Service are:

- Export your charts, while preserving their formatting.
- Change exported file settings as per the requirement.

Export Service is a web-application and is deployed on Internet Information Services (IIS). Run the **C1ASPNETExportService** installer, placed in C:\Program Files\ComponentOne\ASP.NET Web Forms Edition folder. It installs the following files in IIS.




You can also find these files at the following location:

C:\ProgramData\ComponentOne\C1ASPNET\C1APNETExportService

System Requirements

Following are the system requirements of the service host:

- Microsoft Windows 7 or above.
- IIS 7.0 or above with ASP.NET 4.0 or above (.NET framework 4.0).
- IE9 or higher on the service host.

 You may update .Net framework 4.0 in **Microsoft Windows 7** or **Microsoft Windows Server 2008 R2**. See <http://support.microsoft.com/kb/2468871>, for more information.

Export Settings for Charts

Following are the settings to export charts.

Export as Image

- **FileFormat:** Exports as a *.jpg*, *.bmp*, *.gif*, *.png* or a *.tiff* image.
- **Author:** Specifies the name of the person or organization responsible for creating the data.
- **Server URL:** Sets the server URL. Enter ServerURL/**exportapi/chart**.
- **File Name:** Sets the file name to be used for the exported image.

Export to PDF

- **Auto Fit Width:** Enables auto-fit.
- **Landscape:** Enables landscape mode.
- **File Content:**
 - Image Quality: Sets the image quality to Low, Medium or High
 - Compression: Sets the compression level to Default, None, Best Speed or Best Compression.
 - Font Type: Sets the Font type to True Type or Embedded.
- **Document Info:**

- Author: Sets the name of the person or organization that created the document.
- Creator: Sets the name of the application that created the original document.
- Subject: Sets the subject of the document.
- Title: Sets document title in the title bar.
- Producer: Sets the name of the application that created the PDF document.
- Keywords: Sets the keywords associated with the PDF document that can be used to locate the document.
- **Document Security:**
 - Encryption Type: Sets the Encryption Type to NotPermit, Standard40, Standard128 or Aes128.
 - Owner Password: Sets the password required to edit permissions for the document.
 - User Password: Sets the password required to open the document.
 - Allow Copy Content: Enables or disables copy content.
 - Allow Edit Annotations: Enables or disables users from editing annotations.
 - Allow Edit Content: Enables or disables users from editing content in the document.
 - Allow Print: Enables or disables printing for the document.
- **Configuration Setting:**
 - Server URL: Sets the server URL. Enter ServerURL/**exportapi/chart**.
 - File Name: Sets the file name to be used for the exported PDF.

Usage

The `exportChart` function will be called to export the chart as an image or a PDF. These steps assume that you have added a button to the form, on whose click event you would call the export function. Add the following code within the `<head></head>` tags, to export chart to an image.

```
<script src="http://code.jquery.com/jquery-1.9.1.min.js" type="text/javascript">
</script>
<asp:Placeholder runat="server">
  <!--Export Methods-->
  <script type="text/javascript">
    $(function () {
      $("#Button1").click(exportImage);
    });
    function getChart() { return $("#<%=C1BarChart1.ClientID%>"); }

    // Export function
    function exportImage() {
      var fileName = "ExportImage";
      var type = "Png";
      var url = "http://demos.componentone.com/ASPNET/ExportService" +
"/exportapi/chart";
      var chart = getChart();
      $("#<%=C1BarChart1.ClientID%>").clbarchart("exportChart", fileName, type,
url);
    }
  </script>
</asp:Placeholder>
```

Add the following code within the `<head></head>` tags, to export chart to PDF:

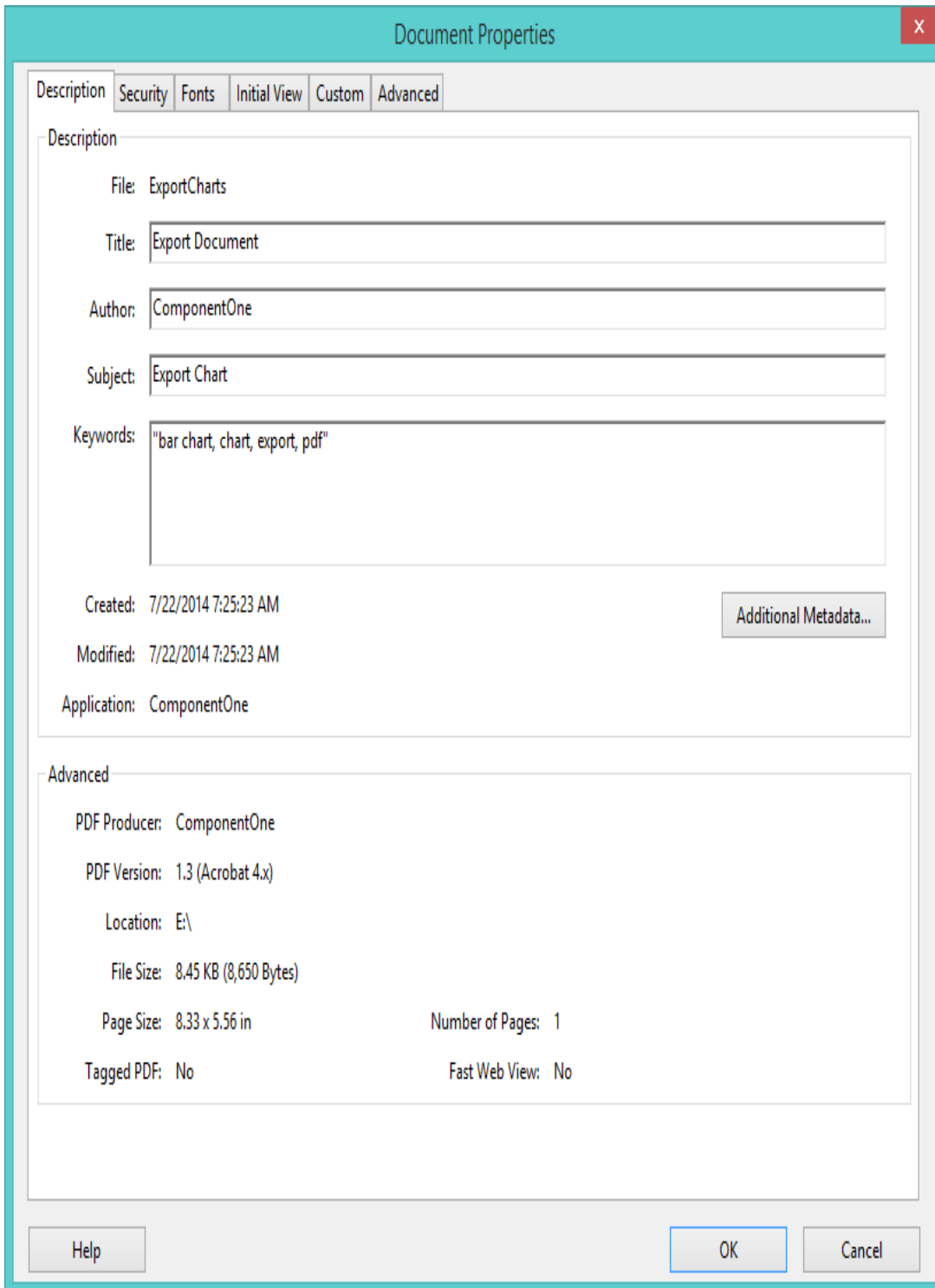
```
<script src="http://code.jquery.com/jquery-1.9.1.min.js" type="text/javascript">
</script>
```

```
<asp:Placeholder runat="server">
<script type="text/javascript">
    $(function () {
        $("#Button1").click(exportPdf);
    });

    // Export function
    function exportPdf() {
        var fileName = "ExportCharts";
        var url = "http://demos.componentone.com/ASPNET/ExportService" +
"/exportapi/chart";
        var pdfSetting = {
            imageQuality: 'Low',
            compression: 'BestCompression',
            fontType: 'TrueType',
            author: 'ComponentOne',
            creator: 'ComponentOne',
            subject: 'Export Chart',
            keywords: 'bar chart, chart, export, pdf',
            allowCopyContent: true,
            allowEditAnnotations: true,
            allowEditContent: true,
            allowPrint: true
        }

        $("#<%=C1BarChart1.ClientID%>").c1barchart("exportChart", fileName, "pdf",
pdfSetting, url);
    }
</script>
</asp:Placeholder>
```

The following image displays the properties of the PDF generated:



In case the file does not download on Internet Explorer, turn off Internet Explorer protected mode to export file or run Internet Explorer as administrator. To turn off the protected mode:

- Open Internet Explorer **Settings** and select **Internet Options**.
- In the **Security** tab select **Internet** and uncheck "**Enable Protected Mode**".

BarChart Animation

C1BarChart's series can be animated using the [Duration](#) and [Easing](#) properties.

Transition Effects

When the [Enabled](#) property is true you can apply animation effects to the Bar chart series. You can enhance your Bar chart's appeal by adding transition effects between the sliding or fading animated states to create a seamless flow between these states. Instead of the Bar chart series moving along smoothly from left to right while loading, the chart could bounce in as the series slides in and bounce out as the series slides out. By default the [Easing](#) property is set to **Easear** and when you reload the bar chart each series loads with a smooth ar transition effect.

The following transitions effects are available for you to animate the transition between states so it looks smooth to help keep the user oriented while you loading the chart series:

Transition Name	Transition Description
EaseInBack	Back easing in. Begins slowly and then accelerates.
EaseInCubic	Cubic easing in. Begins at zero velocity and then accelerates.
EaseInOutCubic	Cubic easing in and out. Begins at zero velocity, accelerates until halfway, and then decelerates to zero velocity again.
EaseOutBack	Back easing out. Begins quickly and then decelerates.
EaseOutBounce	Bouncing easing out. Begins quickly and then decelerates. The number of bounces is related to the duration: longer durations produce more bounces.
EaseOutCubic	Cubic easing in and out. Begins at full velocity and then decelerates to zero.
EaseOutElastic	Quintic easing out. Begins at full velocity and then decelerates to zero.

Animation Effect Duration

You can set the length of **C1BarChart**'s animation effect takes using the [Duration](#) property. The unit of time used for specifying animation effect duration is in milliseconds, and the default setting for the **Duration** property is **500** milliseconds (or half a second). Increase this value for longer animation effect, and decrease this number for a shorter animation effect.

Samples

Please be advised that this ComponentOne software tool is accompanied by various sample projects and/or demos, which may make use of other ComponentOne development tools included with ComponentOne Studio Enterprise.

C# Samples

The following pages within the **ControlExplorer** sample installed with **ASP.NET Web Forms Edition** detail the [C1BarChart](#) control's functionality:

Sample	Description
Overview	Demonstrates some of the basic features of the C1BarChart. The source in this sample will show you how to set the text of the X axis and Y axis; how to add a header to the chart; how to add label text to the legend; and how to populate the chart with data.
Series transition	Demonstrates how to set the seriesTransition of your barcharts. This example uses the seriesTransition options of the bar chart.
Simple bar chart	This sample illustrates a simple bar chart that compares the sales between the Nintendo Wii, the Xbox 360, and the Playstation 3.
Column bar chart	This sample depicts a clustered column bar chart. This is similar to the "Clustered Chart" sample, although the horizontal option of the bar chart is set to false here.
Clustered bar chart	This sample depicts a clustered column bar chart. This is similar to the "Clustered Chart" sample, although the horizontal option of the bar chart is set to false here.
Stacked bar chart	This sample creates a stacked bar chart.
Stacked percentage bar chart	This sample illustrates a stacked percentage chart. To get this result, all you need to do is set the is100Percent option to true and the stacked property to true.
External datasource	This sample illustrates how to create a chart using data from an external data source. In this example, we are using data from the Netflix OData feed.
Data binding	This sample demonstrates how to binding bar chart simply by setting the DataSourceID.
Fixed label width	This sample depicts a bar chart with the width of its labels fixed. It also illustrates how simple it is to rotate labels and align text.

Task-Based Help

The task-based help assumes that you are familiar with programming in ASP.NET and know how to use controls in general. By following the steps outlined in the help, you will be able to create projects demonstrating a variety of **C1BarChart** features, and get a good sense of what the C1BarChart control can do.

Axis Tasks

This section contains procedures for customizing the Axis's appearance such as changing its font style, font color, and font size as well formatting the axis label's rotation, width, and alignment.

Rotating the Axis Label

This topic shows how to rotate the axis label at design time and programmatically using the Rotation property.

To set the y-axis labels at design time:

1. In design time, select the BarChart control and navigate to its properties window.
2. Expand the Axis->Y->Labels->AxisLabelStyle nodes and set the Rotation property to 60.

To set the y-axis labels programmatically:

To programmatically rotate the Y-axis annotation to a 60 degree angle so it appears slanted toward the use the following code:

To write code in Visual Basic

Visual Basic

```
BarChart.Axis.Y.Labels.AxisLabelStyle.Rotation = 60
```

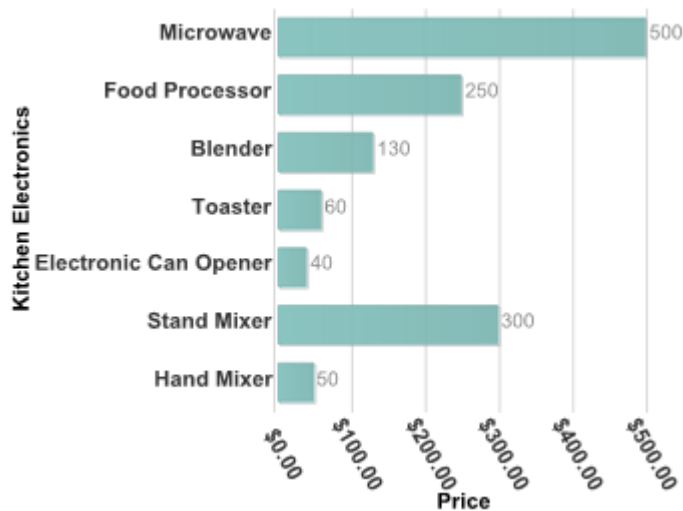
To write code in C#

C#

```
BarChart.Axis.Y.Labels.AxisLabelStyle.Rotation = 60;
```

This topic illustrates the following:

The following image shows the Y-Axis labels rotated to a 60 degree angle:



Formatting the Axis Labels

The Axis labels in the BarChart can be formatted using the following properties: Labels.Width, Labels.TextAlign, and Labels.Style.Rotation.

To format the axis label's width and alignment in design view:

1. Add a reference to the **C1.Web.Wijmo.Controls.Design.3.dll** to your project.
2. Add the **C1BarChart** from the toolbox to your page. For more information on adding the C1BarChart icon to your Toolbox, see [Getting Started with ASP.NET Web Forms Edition](#).
3. In the C1BarChart properties window expand the X->Labels node and set the TextAlign property to Center and the Width property to 150.

Note: This will align the X-Axis labels to the center and create a fixed width of 150 for each label.

To format the chart labels width and alignment in source view:

To format the chart labels width and alignment in source view, click the source tab and add the following code in your source file:

Add the following within the <X></X> tags:

To write code in Source View

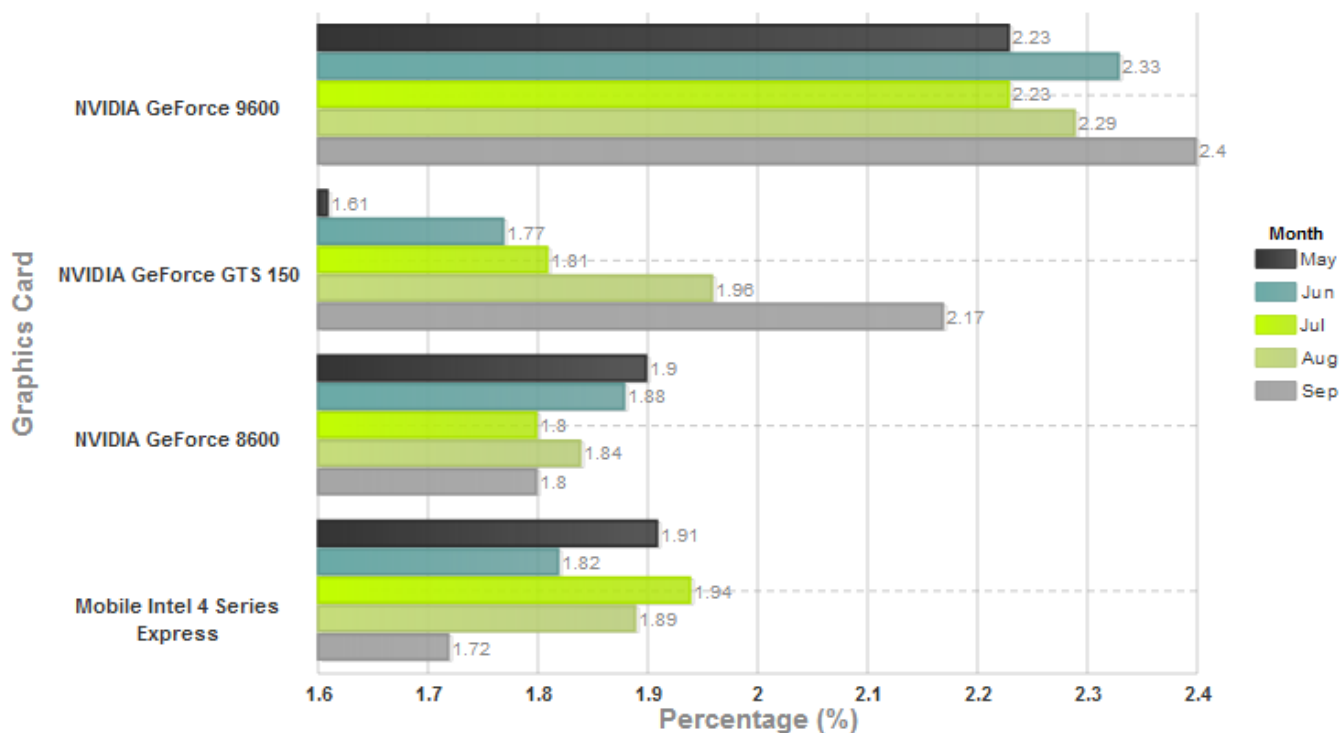
```
<Labels Width="150" TextAlign="Center">
<Style Rotation="0"></Style>
```

This topic illustrates the following:

The X-Axis labels which appear where the Y-Axis labels are since a Bar chart is an inverted column chart

Note: This will align the X-Axis labels to the center and create a fixed width of 150 for each label.

Steam Top 4 Video Cards - Hardware Survey



Using the Origin to Display Positive and Negative Axes Values

Use the Origin property to tell the axis where to draw its starting point. This is optimal for displaying positive and negative values on the same chart.

To set the Origin property in design view:

1. Add a reference to the **C1.Web.Wijmo.Controls.Design.3.dll** to your project.
2. Add the **C1BarChart** from the toolbox to your page. For more information on adding the C1BarChart icon to your Toolbox, see [Getting Started with ASP.NET Web Forms Edition](#).
3. In the C1BarChart properties window expand the **Axis->X** node and set the Origin property to 2.6.
4. In the C1BarChart properties window expand the **Axis->Y** node and set the **Origin** property to **0** and **Alignment** to **Far**.

To set the Origin property in source view:

1. Add a reference to the **C1.Web.Wijmo.Controls.Design.3.dll** to your project.
2. Add the **C1BarChart** from the toolbox to your page. For more information on adding the C1BarChart icon to your Toolbox, see [Getting Started with ASP.NET Web Forms](#).
3. Click the Source tab and add the following code within the <Axis> tags:

To write code in Source View

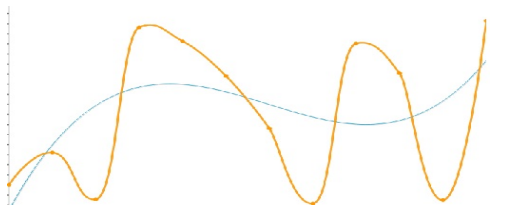
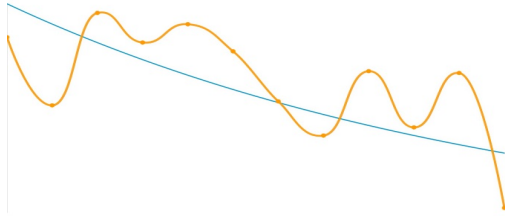
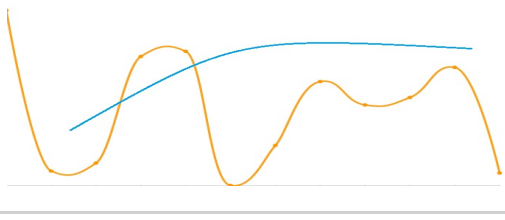
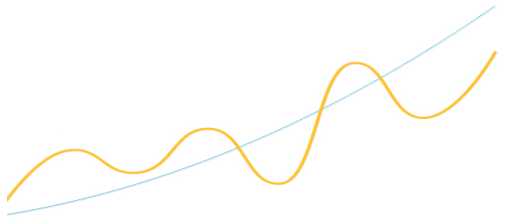
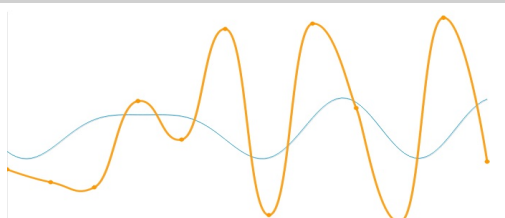
```
<Axis>
  <X Origin="2.6"></X>
  <Y Alignment="Far" Visible="true" Origin="0"></Y>
</Axis>
```


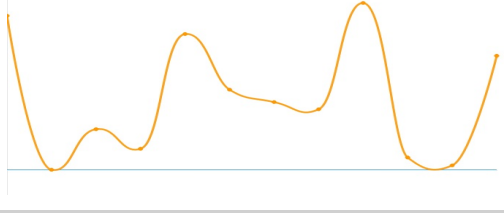
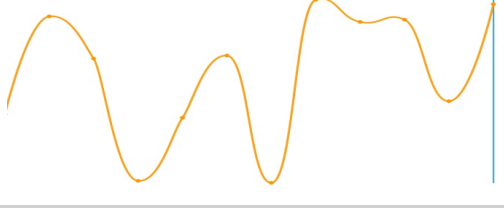
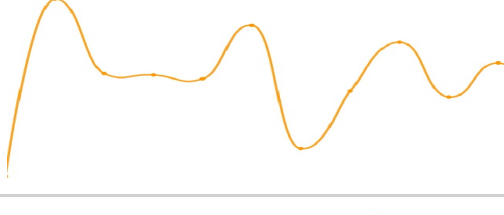
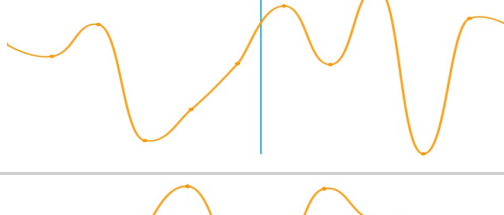
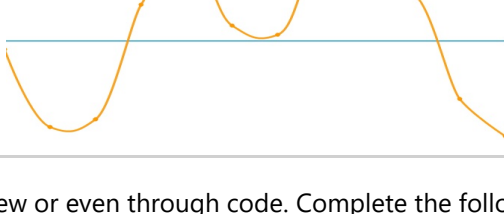
Trendline

Trendlines are used to represent trends in data and to examine problems of prediction. Trendlines are commonly used with price charts or financial charts, but they can also be used with a variety of technical analysis charts such as **MACD** (moving average convergence/divergence) which is a trading indicator used in technical analysis of stock prices, or **RSI** (relative strength index) which is a technical indicator used in the analysis of financial markets.

Types of Trendlines

The following table displays the supported FitTypes. Each trend type is drawn based on the calculation formula of its type.

FitType	Description	Preview
Polynom	A twisted line that is used when data oscillates. It is useful for analyzing gains and losses over a large data set.	
Exponent	A curved line that is convenient to use when data values rise or fall at increasingly higher rates. You cannot create an exponential trendline if your data contains zero or negative values.	
Logarithmic	A best fit curved line used for better visualization of data. Used when the rate of change in the data increases or decreases quickly and then levels out. It can also use positive and negative values.	
Power	A curved line that is best used with data sets that compare calculation that increase at a peculiar rate. For example, the acceleration of a vehicle at one-second intervals.	
Fourier	A way to display a wave like function as a combination of simple sine waves. It is created by using the fourier series formula.	

Min X	Takes the minimum value of X from the chart and draws a trendline using it.	
Min Y	Takes the minimum value of Y from the chart and draws a trendline using it	
Max X	Takes the maximum value of X from the chart and draws a trendline using it.	
Max Y	Takes the maximum value of Y from the chart and draws a trendline using it.	
Average X	Calculates the average value of X from the chart data and draws a trendline.	
Average Y	Calculates the average value of Y from the chart data and draws a trendline.	

TrendLines can be added through the designer, in the source view or even through code. Complete the following steps to add a trendline to a BarChart.

In the Designer

These steps assume that you have already added data to the BarChart. Please see [BarChart for ASP.NET Web Forms Quick Start](#) for more information.

1. Select the **BarChart** control and click the smart tag to open the **BarChart Tasks Menu**.
2. Select **SeriesList** from the BarChart Tasks Menu. The **BarChartSeries Collection** Editor appears.
3. Click the **Add** button to add a new series. Set the **isTrendline** property for the new series to True.
4. Expand the **Trendline Series** property group. You will see the following properties.

TrendlineSeries	C1.Web.Wijmo.Control
Data	C1.Web.Wijmo.Contro
FitType	MaxY
Order	4
SampleCount	150

- **FitType** - Specifies the type of the trendline.
 - **SampleCount** - Specifies the sample count for function calculation for trendline. Note it only works if the FitType is polynom, power, exponent, logarithmic and fourier.
 - **Order** - Specifies the number of terms in polynom equation. It works only if the FitType is set to polynom, power, exponent, logarithmic or fourier.
5. Expand the TrendlineSeries.Data property group. Click the ellipses button (...) next to the TrendlineSeries.Data.X.Values property. This will open the **ChartXData Collection Editor**.
 6. Enter the values to be displayed on the X axis and click **OK**.
 7. Click the ellipses button (...) next to the TrendlineSeries.Data.Y.Values property. This will open the **ChartYData Collection Editor**.
 8. Enter the values to be displayed on the Y axis and click **OK**.
 9. Set the **FitType** property, **Order** property and **SampleCount** property as per your choice.
 10. Click **OK** to close the BarChartSeries Collection Editor.

 **Note:** For **Power** and **Logarithmic** trendlines, only numeric values are allowed on the axes.

In Source View

Add the following markup within the <SeriesList></SeriesList> tags, to add a trendline to the chart.

Markup

```
<cc1:BarChartSeries LegendEntry="True" IsTrendline="true">
<TrendlineSeries FitType="Polynom">
  <Data>
    <X>
      <Values>
        <cc1:ChartXData StringValue="QTR1" />
        <cc1:ChartXData StringValue="QTR2" />
        <cc1:ChartXData StringValue="QTR3" />
        <cc1:ChartXData StringValue="QTR4" />
      </Values>
    </X>
    <Y>
      <Values>
        <cc1:ChartYData DoubleValue="13" />
        <cc1:ChartYData DoubleValue="4" />
        <cc1:ChartYData DoubleValue="18" />
        <cc1:ChartYData DoubleValue="8" />
      </Values>
    </Y>
  </Data>
</TrendlineSeries>
</cc1:BarChartSeries>
```

In Code

Add the following code to the **Page_Load** event, to add a Trendline to the BarChart Control.

To write code in C#

```
// Create new series
var seriesTrendline = new BarChartSeries();
seriesTrendline.IsTrendline = true;
seriesTrendline.Label = "Trendline";
seriesTrendline.TrendlineSeries.FitType = TrendlineFitType.Polynom;
seriesTrendline.TrendlineSeries.Order = 4;
seriesTrendline.TrendlineSeries.SampleCount = 100;

// Add series to the chart
this.C1BarChart1.SeriesList.Add(seriesTrendline);

// Add X Data
seriesTrendline.TrendlineSeries.Data.X.Add("QTR1");
seriesTrendline.TrendlineSeries.Data.X.Add("QTR2");
seriesTrendline.TrendlineSeries.Data.X.Add("QTR3");
seriesTrendline.TrendlineSeries.Data.X.Add("QTR4");

// Add Y Data
seriesTrendline.TrendlineSeries.Data.Y.Add(13);
seriesTrendline.TrendlineSeries.Data.Y.Add(4);
seriesTrendline.TrendlineSeries.Data.Y.Add(18);
seriesTrendline.TrendlineSeries.Data.Y.Add(8);
```

To write code in Visual Basic

```
' Create new series
Dim seriesTrendline = New BarChartSeries()
seriesTrendline.IsTrendline = True
seriesTrendline.Label = "Trendline"
seriesTrendline.TrendlineSeries.FitType = TrendlineFitType.Polynom
seriesTrendline.TrendlineSeries.Order = 4
seriesTrendline.TrendlineSeries.SampleCount = 100

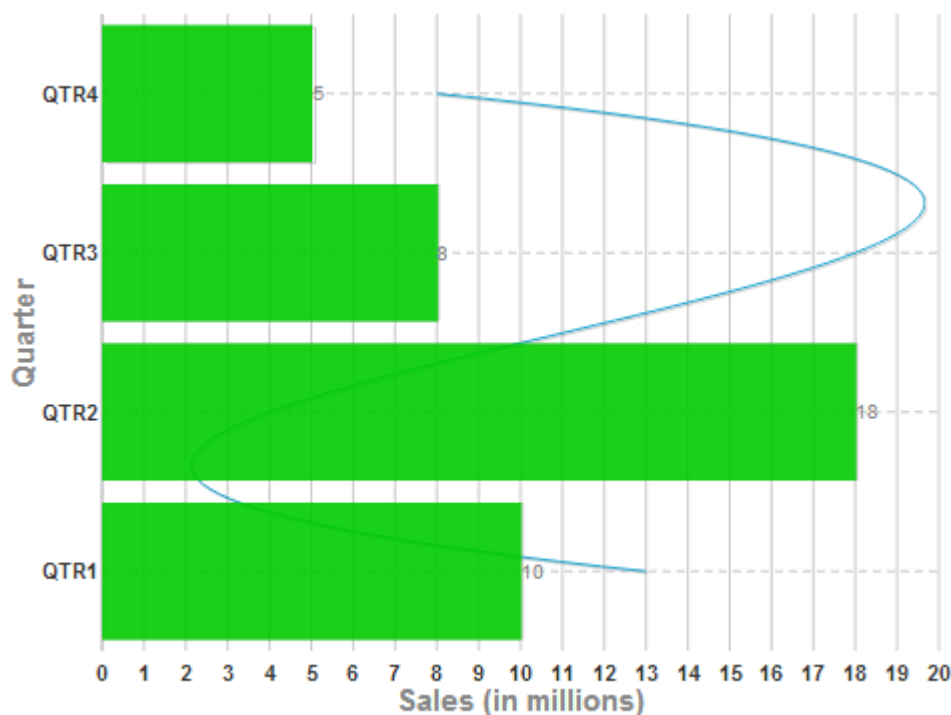
' Add series to the chart
Me.C1BarChart1.SeriesList.Add(seriesTrendline)

' Add X Data
seriesTrendline.TrendlineSeries.Data.X.Add("QTR1")
seriesTrendline.TrendlineSeries.Data.X.Add("QTR2")
seriesTrendline.TrendlineSeries.Data.X.Add("QTR3")
seriesTrendline.TrendlineSeries.Data.X.Add("QTR4")

' Add Y Data
seriesTrendline.TrendlineSeries.Data.Y.Add(13)
seriesTrendline.TrendlineSeries.Data.Y.Add(4)
seriesTrendline.TrendlineSeries.Data.Y.Add(18)
seriesTrendline.TrendlineSeries.Data.Y.Add(8)
```

What You've Accomplished

When you run the project, notice that a blue trendline appears in the BarChart control.



Binding the BarChart to an Access Data Source

To bind the **C1BarChart** to an access data source using the **DataSourceID**, **C1ChartBinding.XField**, **C1ChartBinding.XFieldType**, **C1ChartBinding.YField**, and **C1ChartBinding.YfieldType** properties, complete the following:

1. Add a reference to the **C1.Web.Wijmo.Controls.Design.3.dll** to your project.
2. Add the **C1BarChart** from the toolbox to your page. For more information on adding the C1BarChart icon to your Toolbox, see [Getting Started with ASP.NET Web Forms Edition](#).
3. Add the **C1NWind.mdb** file to the App_Data folder within your project.

 **Note:** The C1Nwind.mdb file can be found in **Documents\ComponentOne Samples\Common**.

4. Click on the **Source** tab and enter the following in your source file after the ending **</cc1:C1BarChart>** tag:

To write code in Source View

```
<asp:AccessDataSource ID="AccessDataSource1" runat="server"
    DataFile="~/App_Data/C1NWind.mdb"
    SelectCommand="select CategoryName, sum(ProductSales) as Sales from
    (SELECT DISTINCTROW Categories.CategoryName as CategoryName,
    Products.ProductName, Sum([Order Details Extended].ExtendedPrice) AS ProductSales
    FROM Categories INNER JOIN (Products INNER JOIN (Orders INNER JOIN [Order Details
    Extended] ON Orders.OrderID = [Order Details Extended].OrderID) ON
    Products.ProductID = [Order Details Extended].ProductID) ON Categories.CategoryID
    = Products.CategoryID
    WHERE (((Orders.OrderDate) Between #1/1/95# And #12/31/95#))
    GROUP BY Categories.CategoryID, Categories.CategoryName, Products.ProductName
    ORDER BY Products.ProductName) group by CategoryName;">
</asp:AccessDataSource>
```

5. Set the **DataSourceID** property to **AccessDatSource1**, **Height** to **475**, and **Width** to **756** like following:

To write code in Source View

```
<cc1:C1BarChart ID="C1BarChart1" runat="server" DataSourceID="AccessDataSource1"
Height="475" Width = "756">
```

6. Set the BarChart Header **Text** property like the following:

To write code in Source View

```
<Header Text="Sales"></Header>
```

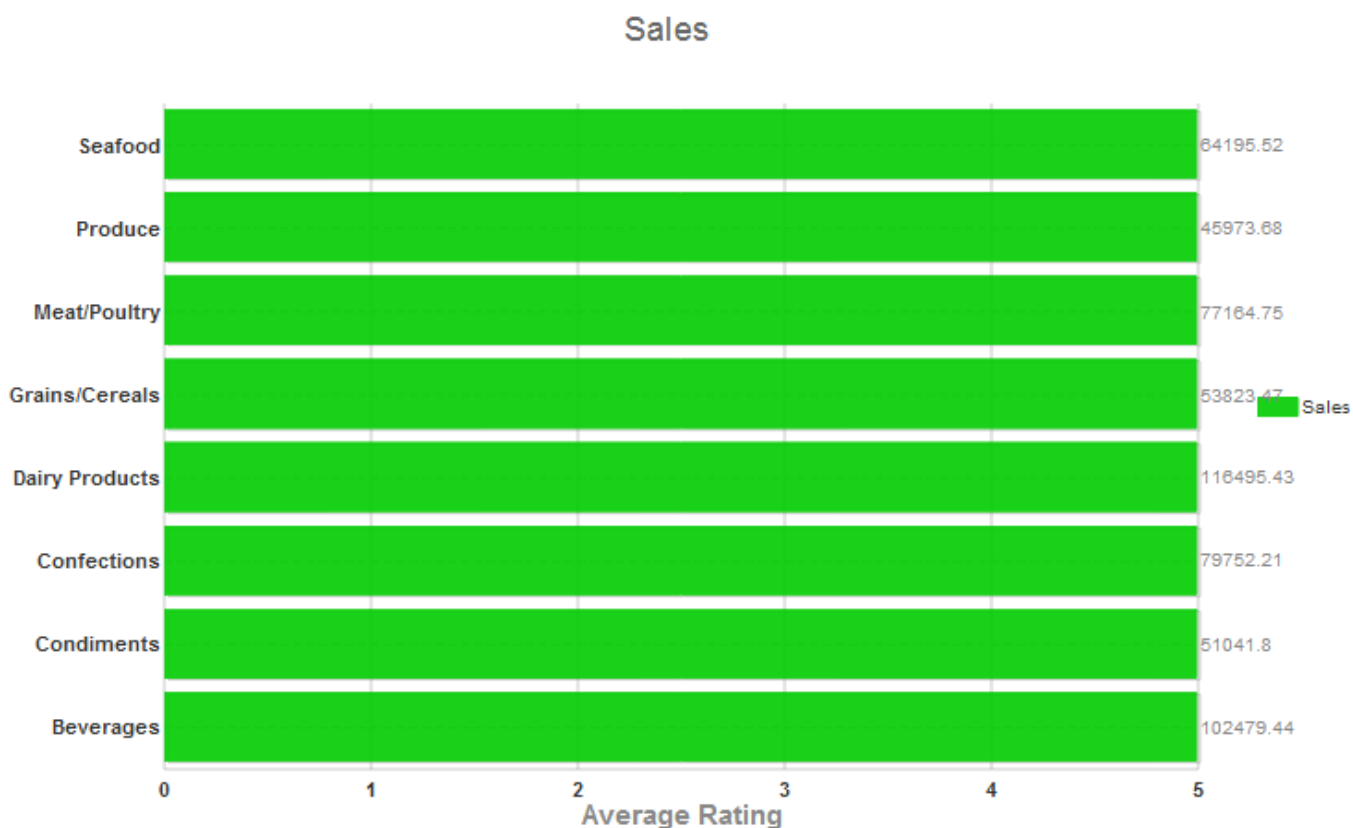
7. Define the DataBindings for the **C1ChartBinding.XField**, **C1ChartBinding.XFieldType**, **C1ChartBinding.YField**, and **C1ChartBinding.YFieldType** before the </cc1:C1BarChart> like the following:

To write code in Source View

```
<DataBindings>
    <cc1:C1ChartBinding XField="CategoryName" XFieldType="String"
YField="Sales" YFieldType="Number" />
</DataBindings>
```

This topic illustrates the following:

The following image shows the C1BarChart bound to an external database:



Creating a Column Chart

To create a column chart, set the **Horizontal** property to **False** to convert the bar chart to a column chart.

To set the **Horizontal** property in design time:

1. Add a reference to the **C1.Web.Wijmo.Controls.Design.3.dll** to your project.
2. Add the **C1BarChart** from the toolbox to your page. For more information on adding the C1BarChart icon to

your Toolbox, see [Getting Started with ASP.NET Web Forms Edition](#).

3. In the C1BarChart properties window expand the X->Labels node and set the TextAlign property to Center and the Width property to 150.

To set the **Horizontal** property in source view:

1. Add a reference to the **C1.Web.Wijmo.Controls.Design.3.dll** to your project.
2. Add the **C1BarChart** from the toolbox to your page. For more information on adding the C1BarChart icon to your Toolbox, see [Getting Started with ASP.NET Web Forms](#).
3. Click the source tab and add the Horizontal property within the <cc1:C1BarChart> so it appears like the following:

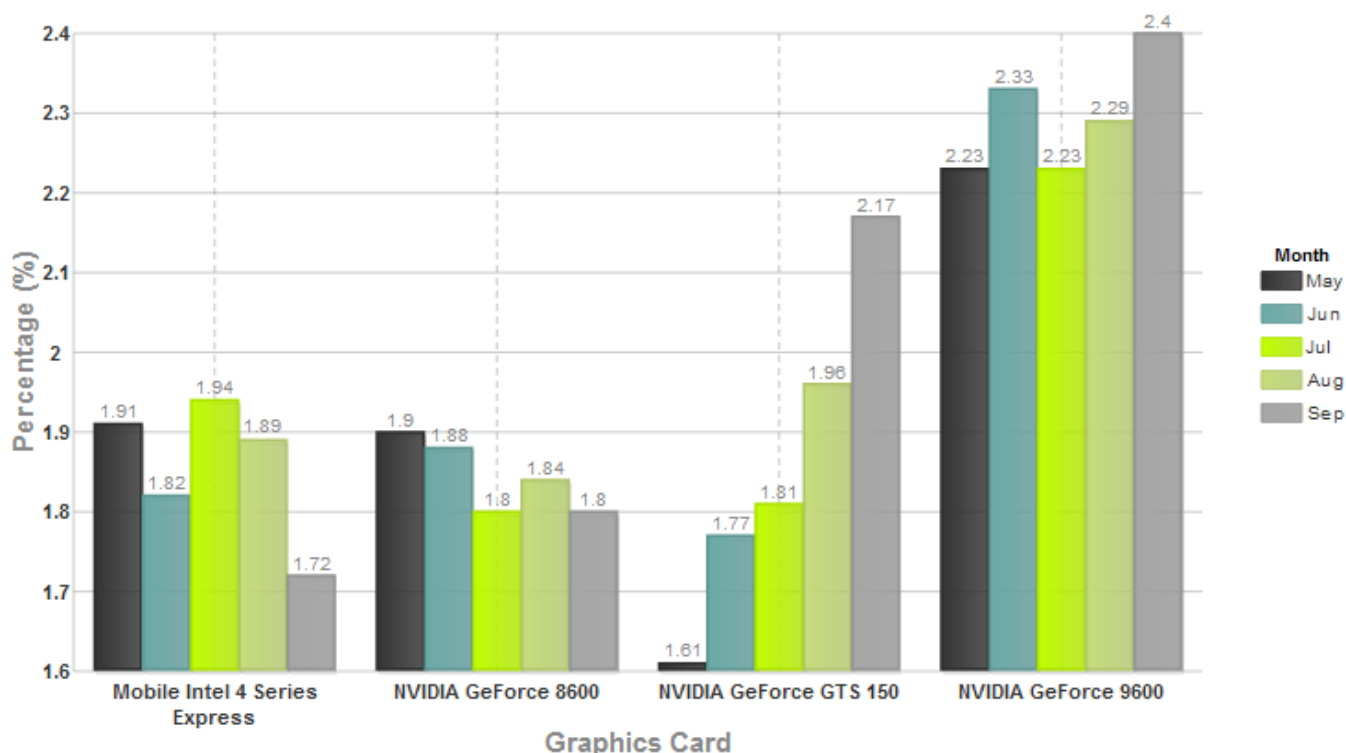
To write code in Source View

```
<cc1:C1BarChart ID="C1BarChart1" Width="756" Height="475" runat="server"
    Horizontal="False">
```

This topic illustrates the following:

The following chart appears as a Column chart with the X and Y axes reversed:

Steam Top 4 Video Cards - Hardware Survey



Styling the Series

The BarChartSeries can be styled using the **SeriesStyle** property.

To style the series in design time:

1. Add a reference to the **C1.Web.Wijmo.Controls.Design.3.dll** to your project.
2. Add the **C1BarChart** from the toolbox to your page. For more information on adding the C1BarChart icon to your Toolbox, see [Getting Started with ASP.NET Web Forms Edition](#).
3. In the C1BarChart properties window click on the ellipsis button next to the SeriesStyles property.

The **ChartStyle Collection Editor** appears.

4. Click **Add** to add a new ChartStyle member.
Each ChartStyle will represent a particular BarChartSeries. For example, the first ChartStyle member will represent the series for the first BarChartSeries.
5. In the **ChartStyle Collection Editor** expand the Fill node and click on the ellipsis button next to ColorBegin and enter the following value in the Value textbox: Hex={FF,66,CC}.
6. Click **OK** to save and close the **More Colors** dialog box.
7. In the **ChartStyle Collection Editor** expand the Fill node and click on the ellipsis button next to **ColorBegin** and enter the following value in the **Value** textbox: Hex={FF,66,CC}.
8. Click **OK** to save and close the **More Colors** dialog box.
9. Set the Type to LinearGradient and Opacity to 0.8.
10. Click on the ellipsis button next to Stroke and enter the following value in the Value textbox: Hex={FF,00,66}.
11. Click **OK** to save and close the **More Colors** dialog box.

To style the series in source view:

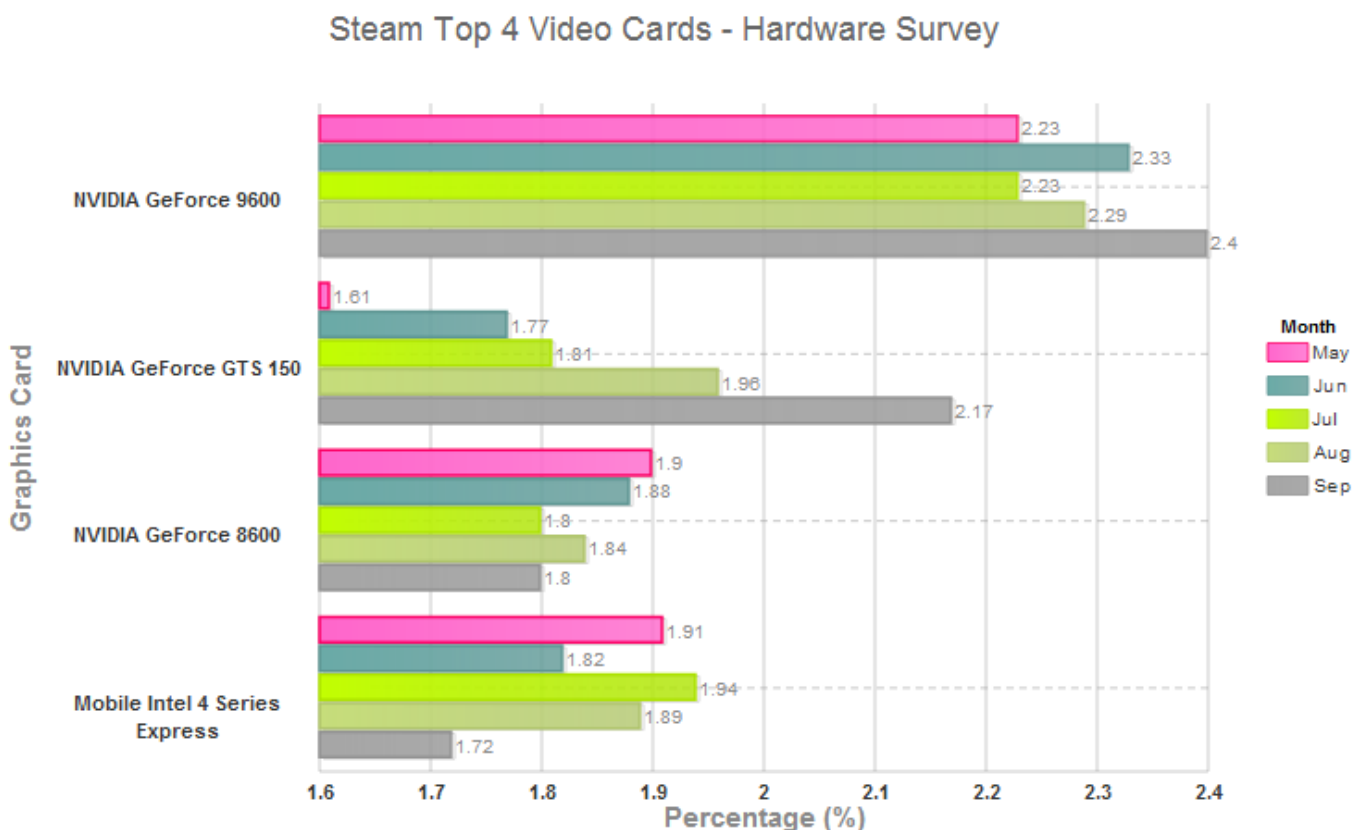
1. Add a reference to the **C1.Web.Wijmo.Controls.Design.3.dll** to your project.
2. Add the **C1BarChart** from the toolbox to your page. For more information on adding the C1BarChart icon to your Toolbox, see [Getting Started with ASP.NET Web Forms](#).
3. Click the Source tab and add the following code within the <SeriesStyle> tag:

To write code in Source View

```
<cc1:ChartStyle Opacity="0.8" Stroke="#FF0066" StrokeWidth="1.5">
  <Fill Type="LinearGradient" ColorBegin="#FF66CC" ColorEnd="#FF66CC"></Fill>
</cc1:ChartStyle>
```

This topic illustrates the following:

The first BarChartSeries style is modified:



Client-Side Tasks

This section provides procedures for applying client-side script to the C1BarChart.

For a list of available client-side members in the C1BarChart see [BarChart for ASP.NET Web Forms Client-Side Reference](#).

Creating a Bar Chart from an External Data Source from the Client Side

To create a Bar chart from an external data source from the client-side, complete the following:

1. Add a reference to the **C1.Web.Wijmo.Controls.Design.4.dll** to your project.
2. Add the **C1BarChart** from the toolbox to your page. For more information on adding the C1BarChart icon to your Toolbox, see [Getting Started with ASP.NET Web Forms Edition](#).
3. Click on the **Source** tab and enter the following code in your source file after the ending body tag:

To write code in Source View

HTML

```
<script type = "text/javascript"
    function hintContent() {
        return this.data.label + '\n ' + this.y + '';
    }
    $(document).ready(function () {
        $.support.cors = true;
        var oData =
"http://demo.componentone.com/aspnet/Northwind/northwind.svc/Products?
$formats=json&$top=10&$orderby=Unit_Price%20desc";

        $.ajax({
            crossDomain: true,
            header: { "content-type": "application/javascript" },
            url: oData,
            jsonp: "$callback",
            success: callback
        });
    });

function callback(result) {
    // unwrap result
    var names = [];
    var prices = [];
    var products = result["d"];
    for (var i = 0; i < products.length; i++) {
        names.push(products[i].Product_Name);
        prices.push(products[i].Unit_Price);
    }

    $("#<%= C1BarChart1.ClientID %>").c1barchart("option", "seriesList", [
    {
```

```
        label: "Prices",  
        legendEntry: true,  
        data: {  
            x: names,  
            y: prices  
        }  
    }  
});  
</script>
```

4. Define the Axis Y **Text**, **Min**, **Max**, and **AutoMax**, as follows:

To write code in Source View

HTML

```
<Axis>  
    <Y Text>="Prices" AutoMin="true" AutoMax="true" Compass="West"></Y>  
    <X Text="Products"></X>  
</Axis>
```

5. Define the Header **Text** and Hint **Content**, as follows:

To write code in Source View

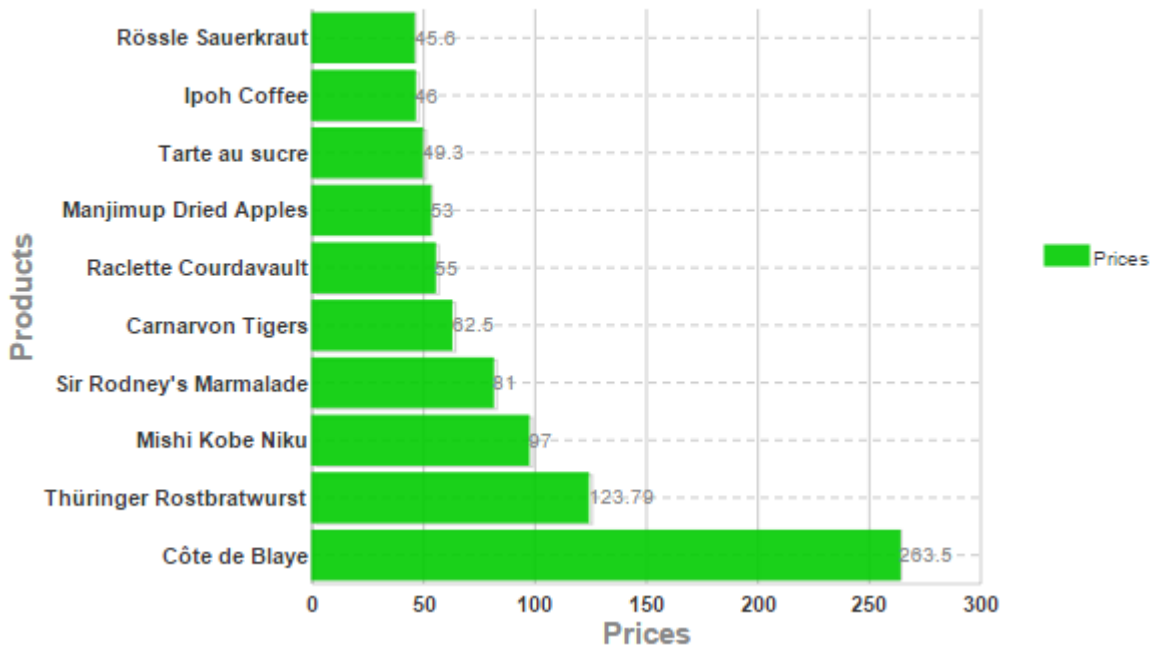
HTML

```
<Header> Text="Top 10 Products by Unit Price - Northwind OData"></Header>  
<Hint><Content Function="hintContent" />  
</Hint>
```

6. Set the BarChart **Height** to **475** and BarChart **Width** to **756**.
7. Run and build the project.

This topic illustrates the following:

Top 10 Products by Unit Price - Northwind OData



Client-Side Tutorials

The following tutorials will walk you through completing more complex applications using the C1BarChart control's client-side scripting.

Drilling Down in BarChart Data

Using markup and jQuery script, you can create a chart which allows users to drill down in the data on a click event. In this topic, you'll use both markup and client-side script to create a C1BarChart control, and perform the drill-down task by handling the click event and passing the new data on the client-side.

You can find the basis for this topic and a sample in the [Drill-down Wijmo Barchart](#) blog post. The database file added to the application below can be found in the sample's .zip file. The sample is called C1Chart - Drilling Down.

This topic assumes that you began with an empty ASP.NET project. Complete the following steps to create a drill-down barchart:

Step 1 of 4: Set up your application

Step 1 of Set up your application

In this step, you'll create your application and add folders to your application. You'll also add a database file and two code files to the folders.

1. Add a reference to the **C1.Web.Wijmo.4.dll** assembly to your project.
2. Right-click your project name and select **Add | Web Form**. Enter a name for your Web Form, in this case enter **Name**, and click **OK**.
3. Right-click your application name and select **Add | Add ASP.NET Folder | App_Code**.
4. Right-click the application name again and select **Add | Add ASP.NET Folder | App_Data** from the list.
5. Right-click the **App_Code** file. There are two ways to add a code file to the **App_Code** file:
 - a. Select **Add Existing File** from the list.
 1. Browse to locate the **C1Chart - Drilling Down** sample and open the **App_Code** file in the sample.
 2. Select the **Order.cs** code file and click **OK**. The file will be added to the application.
 - b. Select **Add | Code File** from the list. Name the code file **Orders.cs**. Add the following code to the file when it opens:

To write code in C#

```
C#
using System.Collections.Generic;
using System.Linq;
using System.Web;

///<summary>
/// Summary description for Orders
/// </summary>
public class Orders
{

    private double amount;
    private string year,month,day;
```

```
public string Year
{
    get
    {
        return year;
    }
    set
    {
        year = value;
    }
}
public string Month
{
    get
    {
        switch (month)
        {
            case "1": month = "Jan"; break;
            case "2": month = "Feb"; break;
            case "3": month = "Mar"; break;
            case "4": month = "Apr"; break;
            case "5": month = "May"; break;
            case "6": month = "Jun"; break;
            case "7": month = "Jul"; break;
            case "8": month = "Aug"; break;
            case "9": month = "Sep"; break;
            case "10": month = "Oct"; break;
            case "11": month = "Nov"; break;
            case "12": month = "Dec"; break;
        }
        return month;
    }
    set
    {
        month = value;
    }
}
public string Day
{
    get
    {
        return day;
    }
    set
    {
        day = value;
    }
}
```

```
    }

    public double OrderAmount
    {
        get
        {
            return amount;
        }
        set
        {
            amount = value;
        }
    }
}
```

6. Right-click the **App_Code** file again. There are two ways to add the second code file to the application:
 - a. Select **Add Existing File** from the list.
 1. Browse to locate the **C1Chart - Drilling Down** sample and open the **App_Code** file in the sample.
 2. Select the **GetOrders.cs** code file and click **OK**. The file will be added to the application.
 - b. Select **Add | Code File** from the list. Name the code file **GetOrders.cs**. Add the following code to the file when it opens:

To write code in C#

```
C#
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;
using System.Data;
using System.Data.OleDb;
using System.Web.Script.Services;

/// <summary>
/// Summary description for GetOrders
/// </summary>
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
// To allow this Web Service to be called from script, using ASP.NET AJAX,
// uncomment the following line.
[System.Web.Script.Services.ScriptService]
public class GetOrders : System.Web.Services.WebService {

    public GetOrders () {

        //Uncomment the following line if using designed components
        //InitializeComponent();
    }

    [WebMethod]
```



```
[ScriptMethod(ResponseFormat = ResponseFormat.Json)]
public List<Orders> GetDataOnLoad()
{
    OleDbConnection con = new
OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" +
Server.MapPath("~/App_Data/OrdersDataBase.mdb"));
    OleDbCommand cmd = new OleDbCommand("Select Year(OrderDate),
Sum(OrderAmount) from OrdersByDate where Year(OrderDate) In (Select
Distinct(Year(OrderDate)) from OrdersByDate) Group By Year(OrderDate)",
con);

    OleDbDataAdapter da = new OleDbDataAdapter(cmd);
    DataTable dt = new DataTable();
    da.Fill(dt);
    List<Orders> orders = new List<Orders>();
    for (int i = 0; i < dt.Rows.Count; i++)
    {
        Orders od = new Orders();
        od.Year = Convert.ToString(dt.Rows[i][0]);
        od.OrderAmount = Convert.ToDouble(dt.Rows[i][1]);
        orders.Add(od);
    }
    return orders;
}

[WebMethod]
[ScriptMethod(ResponseFormat = ResponseFormat.Json)]
public List<Orders> GetOrderByMonth(string Year)
{
    OleDbConnection con = new
OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" +
Server.MapPath("~/App_Data/OrdersDataBase.mdb"));
    OleDbCommand cmd = new OleDbCommand("Select Month(OrderDate),
Sum(OrderAmount) from OrdersByDate where Month(OrderDate) In (Select
Distinct Month(OrderDate) from OrdersByDate) and Year(OrderDate)="+ Year +
" Group By Month(OrderDate)", con);
    OleDbDataAdapter da = new OleDbDataAdapter(cmd);
    DataTable dtMonths = new DataTable();
    da.Fill(dtMonths);
    List<Orders> orders = new List<Orders>();
    for (int i = 0; i < dtMonths.Rows.Count; i++)
    {
        Orders od = new Orders();
        od.Month = Convert.ToString(dtMonths.Rows[i][0]);
        od.OrderAmount = Convert.ToDouble(dtMonths.Rows[i][1]);
        orders.Add(od);
    }
    return orders;
}

[WebMethod]
[ScriptMethod(ResponseFormat = ResponseFormat.Json)]
public List<Orders> GetOrderByDay(string Month, string Year)
```

```

    {
        switch (Month)
        {
            case "Jan": Month = "1"; break;
            case "Feb": Month = "2"; break;
            case "Mar": Month = "3"; break;
            case "Apr": Month = "4"; break;
            case "May": Month = "5"; break;
            case "Jun": Month = "6"; break;
            case "Jul": Month = "7"; break;
            case "Aug": Month = "8"; break;
            case "Sep": Month = "9"; break;
            case "Oct": Month = "10"; break;
            case "Nov": Month = "11"; break;
            case "Dec": Month = "12"; break;
        }
        OleDbConnection con = new
OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" +
Server.MapPath("~/App_Data/OrdersDataBase.mdb"));
        OleDbCommand cmd = new OleDbCommand("Select Day(OrderDate),
Sum(OrderAmount) from OrdersByDate where Day(OrderDate) In (Select Distinct
Day(OrderDate) from OrdersByDate) and Year(OrderDate)=" + Year + " and
Month(OrderDate)="+Month+" Group By Day(OrderDate)", con);
        OleDbDataAdapter da = new OleDbDataAdapter(cmd);
        DataTable dtDays = new DataTable();
        da.Fill(dtDays);
        List<Orders> orders = new List<Orders>();
        for (int i = 0; i < dtDays.Rows.Count; i++)
        {
            Orders od = new Orders();
            od.Day = Convert.ToString(dtDays.Rows[i][0]);
            od.OrderAmount = Convert.ToDouble(dtDays.Rows[i][1]);
            orders.Add(od);
        }
        return orders;
    }
}

```

7. Right-click the application name again to add the web service that the application needs. There are two ways to add this file:
- Select **Add Existing File** from the list.
 - Browse to the **C1Chart - Drilling Down** sample.
 - Select the **GetOrders.asmx** file and click **OK**. The file will be added to the application.
 - Select **Add | New Items | WebService.asmx** from the list. Name the code file **GetOrders.asmx**.
 - In the Solution Explorer, open the GetOrders.asmx node and delete the GetOrders.asmx.cs file. The file isn't necessary for the application.
 - Double-click **GetOrders.asmx** to open the file. Replace the existing code with the following syntax:

To write code in Source View

```
<%http://helpcentral.componentone.com/nethelp/c1studioWeb%>
```

8. Right-click the **App_Data** folder and select **Add | Existing Item** from the list. Browse to the location where you saved the **C1Chart - Drill Down** sample and locate and select the **OrdersDataBase.mdb** item. Click **OK** to add

the database to your application.

Step 2 of 4: Create the C1BarChart control

In this step, you will add the markup that will create the [C1BarChart](#) control.

1. If it's not already open, open the Source View of the **Main.aspx** page and locate the `<body>` tags.
2. Since you are inserting the markup to create a **C1BarChart** control, add the following to the top of your page to register the **C1Chart** assembly:

To write code in Source View

```
<%@ Register assembly="C1.Web.Wijmo.Controls.4"
namespace="C1.Web.Wijmo.Controls.C1Chart" TagPrefix="cc1" %>
```

3. Locate the set of `<div>` tags within the `<body>` `</body>` tag set and insert the following markup within the tags:

Markup to Add

```
<cc1:C1BarChart ID="C1BarChart1" runat="server" Horizontal="false" >
    <Header Text="Order Details"></Header>
    <TextStyle Fill-Color="#b2b2b2" FontWeight="bold" FontSize="15">
</TextStyle>
    <SeriesHoverStyles>
        <cc1:ChartStyle StrokeWidth="1.5" Opacity="1" ></cc1:ChartStyle>
    </SeriesHoverStyles>
    <Axis>
        <X>
            <Labels>
                <Style Fill-Color="#7f7f7f" FontSize="11"></Style>
            </Labels>
        </X>
        <Y Compass="West">
            <GridMajor>
                <Style Stroke="#353539" StrokeDashArray="-"></Style>
            </GridMajor>
            <Labels>
                <Style Fill-Color="#242529" FontSize="11"></Style>
            </Labels>
        </Y>
    </Axis>
</cc1:C1BarChart>
```

4. Directly below the C1BarChart markup, add the following:

To write code in Source View

```
<asp:HiddenField ID="HiddenField1" runat="server" ClientIDMode="Static" />
```

5. The complete markup you added to your application should resemble the following:

To write code in Source View

```
<body>
    <form id="form1" runat="server">
        <div>
            <cc1:C1BarChart ID="C1BarChart1" runat="server" Horizontal="false" >
                <Header Text="Order Details"></Header>
                <TextStyle Fill-Color="#b2b2b2" FontWeight="bold" FontSize="15">
</TextStyle>
                <SeriesHoverStyles>
```

```

        <ccl:ChartStyle StrokeWidth="1.5" Opacity="1" >:</ccl:ChartStyle>
    </SeriesHoverStyles>
    <Axis>
        <X>
            <Labels>
                <Style Fill-Color="#7f7f7f" FontSize="11"></Style>
            </Labels>
        </X>
        <Y Compass="West">
            <GridMajor>
                <Style Stroke="#353539" StrokeDashArray="-"></Style>
            </GridMajor>
            <Labels>
                <Style Fill-Color="#242529" FontSize="11"></Style>
            </Labels>
        </Y>
    </Axis>
    <:/ccl:C1BarChart>
    <asp:HiddenField ID="HiddenField1" runat="server" ClientIDMode="Static"
/>
    </div>
</form>
</body>

```

In this step, you added the markup to create the [C1BarChart](#) control. In the next step, you'll add the script that will populate the chart data and control the drill-down action.

Step 3 of 4: Add script

In this step, you'll add the script that will populate the chart with data. The script you'll add will also control the drill-down action.

1. Locate the `<head>` `</head>` tags in your **Main.aspx** file. You'll add all the script you need within these tags.
2. Add the references to Wijmo's client side within the `<head>` tags:

To write code in Source View

```

<!--jQuery References-->
<script src="http://code.jquery.com/jquery-1.9.1.min.js" type="text/javascript">
</script>
<script src="http://code.jquery.com/ui/1.10.1/jquery-ui.min.js"
type="text/javascript"></script>

<!--Theme-->
<link href="http://cdn.wijmo.com/themes/aristo/jquery-wijmo.css" rel="stylesheet"
type="text/css" />

<!--Wijmo Widgets JavaScript-->
<script src="http://cdn.wijmo.com/jquery.wijmo-open.all.3.20132.9.min.js"
type="text/javascript"></script>
<script src="http://cdn.wijmo.com/jquery.wijmo-pro.all.3.20132.9.min.js"
type="text/javascript"></script>

```

3. Directly below the references, add the following tag set:

To write code in Source View

```

<script type="text/javascript"></script>

```

4. Add the following script within the tag set you added in the previous step. This will populate your chart with AJAX data, set some of the chart's properties, and return an alert if the status comes back with an error:

To write code in Source View

```

$(document).ready(function () {

    //ajax call to get the data
    $.ajax({
        type: "POST",
        url: "GetOrders.aspx/GetDataOnLoad",
        contentType: "application/json; charset=utf-8",
        dataType: "json",
        data: {},
        success: function (data) {
            var arr = [];
            try {
                //push the data in an array
                $.each(data.d, function (i, elem) {
                    arr.push({
                        Year: elem.Year,
                        OrderAmount: elem.OrderAmount
                    });
                });
                //initialize the barchart widget
                $("#C1BarChart1").clbarchart({
                    shadow: false,
                    hint: { content: function () { return this.data.label
+ '\n ' + this.y + ' '; } },
                    //set the datasource of the BarChart
                    dataSource: arr,
                    seriesList: [{
                        label: "Yearly Amount of Orders",
                        legendEntry: true,
                        data: { x: { bind: "Year" }, y: { bind:
"OrderAmount" } }
                    }],
                    seriesStyles: [{ fill: "180-#ff9900-#ff6600", stroke:
"#ff7800", opacity: 0.8 }],
                });
            }
            catch (e) {
                alert(e);
                return;
            }
        },
        error: function (result, status) {
            if (status = "error") {
                alert(status);
            }
        }
    });
});

```

- Next, add the script that contains the click event for your first drill-down level:

To write code in Source View

```

$("#C1BarChart1").clbarchart({
    click: function (sender, args) {

        //ajax call to get the data
        $.ajax({
            type: "POST",
            url: "GetOrders.aspx/GetOrderByMonth",

```

```

        contentType: "application/json; charset=utf-8",
        dataType: "json",
        //pass the year selected
        data: "{Year:'" + args.x + "'}",
        success: function (data) {
            var arr = [];
            try {
                //push the data in an array
                $.each(data.d, function (i, elem) {
                    arr.push({
                        Month: elem.Month,
                        OrderAmount: elem.OrderAmount
                    });
                });
                //set the datasource of the BarChart
                $("#ClBarChart1").clbarchart({
                    horizontal: false,
                    dataSource: arr,
                    seriesList: [{
                        label: "Orders By Month in year " +
args.x,
                        legendEntry: true,
                        data: { x: { bind: "Month" }, y: { bind:
"OrderAmount" } }
                    }],
                    //attach new handler with click event
                    click: OrdersByDay
                });
                //Save the selected year for further drilling down
                $("#HiddenField1")[0].value = args.x;
            }
            catch (e) {
                alert(e);
                return;
            }
        },
        error: function (result, status) {
            if (status = "error") {
                alert(status);
            }
        }
    });
});
});

```

- The last section of script to add controls the final drill-down level for the chart. The script will call the appropriate data to fill the chart on a click event:

To write code in Source View

```

//method invoked when Month is selected to drill down to days in a month
function OrdersByDay(sender, args) {

    var year = $("#HiddenField1")[0].value;
    var month = args.x;
    //ajax call to get the data
    $.ajax({
        type: "POST",
        url: "GetOrders.aspx/GetOrderByDay",
        contentType: "application/json; charset=utf-8",
        dataType: "json",

```

```

data: '{"Month":"' + month + '", "Year":"' + year + '"}',
success: function (data) {
    var arr = [];
    try {
        //push the data in an array
        $.each(data.d, function (i, elem) {

            arr.push({
                Day: elem.Day,
                OrderAmount: elem.OrderAmount
            });
        });
        //set the datasource of the BarChart
        $("#C1BarChart1").clbarchart("destroy");
        $("#C1BarChart1").clbarchart({
            shadow: false,
            textStyle: { fill: "#b2b2b2", "font-weight": "bold",
"font-size": 15 },
            axis: {
                y: {
                    labels: { style: { fill: "#242529", "font-
size": 11 } },
                    gridMajor: { style: { stroke: "#353539",
"stroke-dasharray": "- " } }
                },
                x: {
                    labels: {
                        style: { fill: "#7f7f7f", "font-size": 11
}
                    }
                }
            },
            hint: { content: function () { return this.data.label
+ '\n ' + this.y + ' '; } },
            header: { text: "Order Details" },
            horizontal: false,
            dataSource: arr,
            seriesList: [{
                label: "Orders By Days in Month " + month + ", " +
year,
                legendEntry: true,
                data: { x: { bind: "Day" }, y: { bind:
"OrderAmount" } }
            }],
            seriesStyles: [{ fill: "180-#ff9900-#ff6600", stroke:
"#ff7800", opacity: 0.8 }],
            seriesHoverStyles: [{ "stroke-width": 1.5, opacity: 1
}
        ]
    });
    $("#C1BarChart1").clbarchart("redraw");
}
catch (e) {
    alert(e);
    return;
}
},

error: function (result, status) {
    if (status = "error") {
        alert(status);
    }
}
}

```

```
});  
}
```

In this step, you added the script to populate the chart with data, and to control the drill-down action. In the next step, you will run your application and test some of the run-time capabilities.

Step 4 of 4: Run Your Application

In this step, you'll run your application and test some of the application's run-time capabilities.

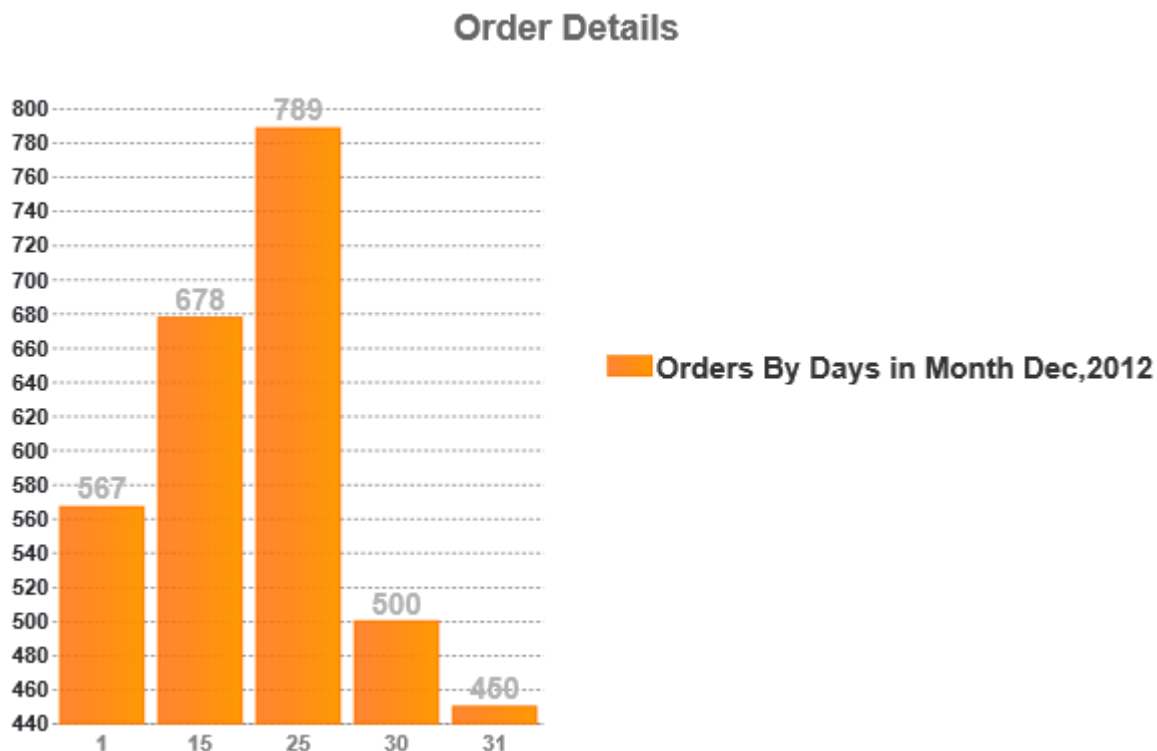
1. Press F5 or start debugging to run your application. Initially, your application should resemble the following image:



2. If you click on one of the chart's bars, for this example the 2012 bar, your application will appear as in the following image:



3. You can drill-down one more level by clicking on one of the months. For this example, select Dec:



Congratulations! You have completed the Drilling Down in BarChart Data tutorial! In this tutorial, you created an ASP.NET Web Forms Edition application, added references to Wijmo, and added markup and script to the application. When you ran your application, you observed the drill-down capabilities of the application.

Loading Data Conditionally

This tutorial will walk you through creating an application which allows you to load data based on certain conditions. You will use markup to create the [C1BarChart](#) and [C1ComboBox](#) controls used in this tutorial and script to handle the client-side events. A purely client-side example can be found in the [Conditionally Load Data in Wijbarchart](#) blog post.

This topic assumes that you began with an empty ASP.NET project. Complete the following steps to create a the application:

Step 1 of 4: Set Up Your Application

Step 1 of 4: Set Up Your Application

In this step, you'll set up your application, including adding the appropriate assembly references to your application, adding a **Main.aspx** WebForm, and adding client-side references.

1. When your application opens, locate the **References** folder in the Solution Explorer. Right-click the folder and select **Add Reference** from the context menu.
2. Browse to locate the **C1.Web.Wijmo.Controls.4.dll** assembly reference. Select it and then select **OK** to add it to your application.
3. Right-click your application name and select **Add | WebForm** from the context menu. Name the new WebForm **Main**, and select **OK**. The new WebForm should open.
4. Locate the `<head>` `</head>` tags on your page. Insert your cursor between the tags and add the following client-side references:

To write code in Source View

```
<!--jQuery References-->
  <script src="http://code.jquery.com/jquery-1.9.1.min.js"
type="text/javascript"></script>
  <script src="http://code.jquery.com/ui/1.10.1/jquery-ui.min.js"
type="text/javascript"></script>
<!--Theme-->
  <link href="http://cdn.wijmo.com/themes/aristo/jquery-wijmo.css"
rel="stylesheet"
  type="text/css" />
  <!--Wijmo Widgets CSS-->
  <link href="http://cdn.wijmo.com/jquery.wijmo-pro.all.3.20133.20.min.css"
rel="stylesheet"
  type="text/css" />
  <!--Wijmo Widgets JavaScript-->
  <script src="http://cdn.wijmo.com/jquery.wijmo-open.all.3.20133.20.min.js"
type="text/javascript"></script>
  <script src="http://cdn.wijmo.com/jquery.wijmo-pro.all.3.20133.20.min.js"
type="text/javascript"></script>
```

5. Below the references, add the following script. This will add some CSS styling to your application:

To write code in Source View

```
<style type="text/css">
  #path
  {
    color: Red;
    border: 1px solid #ddd;
    margin: 10px;
    padding: 0;
  }
</style>
```

Step 2 of 4: Create the Server-Side Controls

In this step, you will add the server-side controls to your application.

1. Place your cursor between the `<body>` `</body>` tags. Locate the **C1BarChart** control in the Visual Studio Toolbox and double-click to add it to the page. The markup added should resemble the following:

To write code in Source View

```
<cc1:C1BarChart ID="C1BarChart1" runat="server">
    <:/cc1:C1BarChart>
```

2. Place your cursor directly below the **C1BarChart** markup. Locate the **C1ComboBox** control in the Visual Studio Toolbox and double-click to add it to the page. The markup added should resemble the following:

To write code in Source View

```
<cc1:C1ComboBox ID="C1ComboBox1" runat="server"><:/cc1:C1ComboBox>
```

3. Next, we'll set some of the **C1BarChart** properties. Place your cursor between the `<cc1:BarChart>` `<:/cc1:BarChart>` tags and insert the following markup. This will format the Header Text:

To write code in Source View

```
<Header Text="Hardware Distribution"></Header>
```

4. Place your cursor within the `<cc1:C1ComboBox>` tag, just after the `runat="server"` option. Edit the markup so that it resembles the following:

To write code in Source View

```
<cc1:C1ComboBox ID="tagsinput" runat="server" HidingAnimation-Animated-Effect="bounce"
    ShowingAnimation-Animated-Effect="bounce" HidingAnimation-Duration="600"
    ShowingAnimation-Duration="600"><:/cc1:C1ComboBox>
```

In this step, you added two controls to your application and set some of their properties in the markup. In the next step, you'll add the script that will allow you to load data conditionally.

Step 3 of 4: Add Script

In this step, you will add the script that will allow you to load data conditionally.

1. Place your cursor below the client-side references that you added in Step 1. Add the following markup:

To write code in Source View

```
<script id="scriptInit" type="text/javascript">
</script>
```

2. The first section of script you'll add populates the **C1ComboBox** with data:

To write code in Source View

```
$(document).ready(function () {
    var staticSeries = [];
    var count = 0;
    $("#tagsinput").c1combobox(
    {
        selectedIndex: -1,
        data: [
```

```

        {
            label: 'West',
            value: 'West'
        },
        {
            label: 'Central',
            value: 'Central'
        },
        {
            label: 'East',
            value: 'East'
        }
    ],

```

- The next part of the script handles the `selectedIndexChanging` event. This allows the data in the `C1BarChart` control to change based on the chosen `C1ComboBoxItem`:

To write code in Source View

```

selectedIndexChanging: function (e, args) {
    var color;
    switch (args.newIndex) {
        case 0: color = "Red"
            break;
        case 1: color = "Blue"
            break;
        case 2: color = "Orange"
    };
    count++;
    var series = $("#C1BarChart1").c1barchart('option',
'seriesList')

    if (count === 1) {
        staticSeries = series;
    }
    staticSeries[args.newIndex].visible = true;
    staticSeries[args.newIndex].legendEntry = true;
    $("#C1BarChart1").c1barchart('option', 'seriesList',
[staticSeries[args.newIndex]]);
    $("#C1BarChart1").c1barchart({
        hint: {
            enable: true
        },
        seriesStyles: [{
            fill: color, stroke: color
        }]
    });
    $("#C1BarChart1").c1barchart('redraw');
}
});

```

- The last section of script to add creates the Hint and the SeriesList data for the `C1BarChart` control:

To write code in Source View

```

$("#C1BarChart1").c1barchart({
    axis: {
        y: {
            text: "Total Hardware"
        },
        x: {
            text: ""
        }
    },
    hint: {

```

```

        enable: false,
        content: function () {
            return this.data.label + '\n ' + this.y + '';
        },
        compass: "south", offsetX: 0, offsetY: 0
    },
    seriesList: [{
        visible: false,
        label: "West",
        legendEntry: false,
        data: { x: ['Desktops', 'Notebooks', 'AIO', 'Tablets',
'Phones'], y: [5, 3, 4, 7, 2] }
    }, {
        visible: false,
        label: "Central",
        legendEntry: false,
        data: { x: ['Desktops', 'Notebooks', 'AIO', 'Tablets',
'Phones'], y: [2, 2, 3, 2, 1] }
    }, {
        visible: false,
        label: "East",
        legendEntry: false,
        data: { x: ['Desktops', 'Notebooks', 'AIO', 'Tablets',
'Phones'], y: [3, 4, 4, 2, 5] }
    }
    ]
    });
});

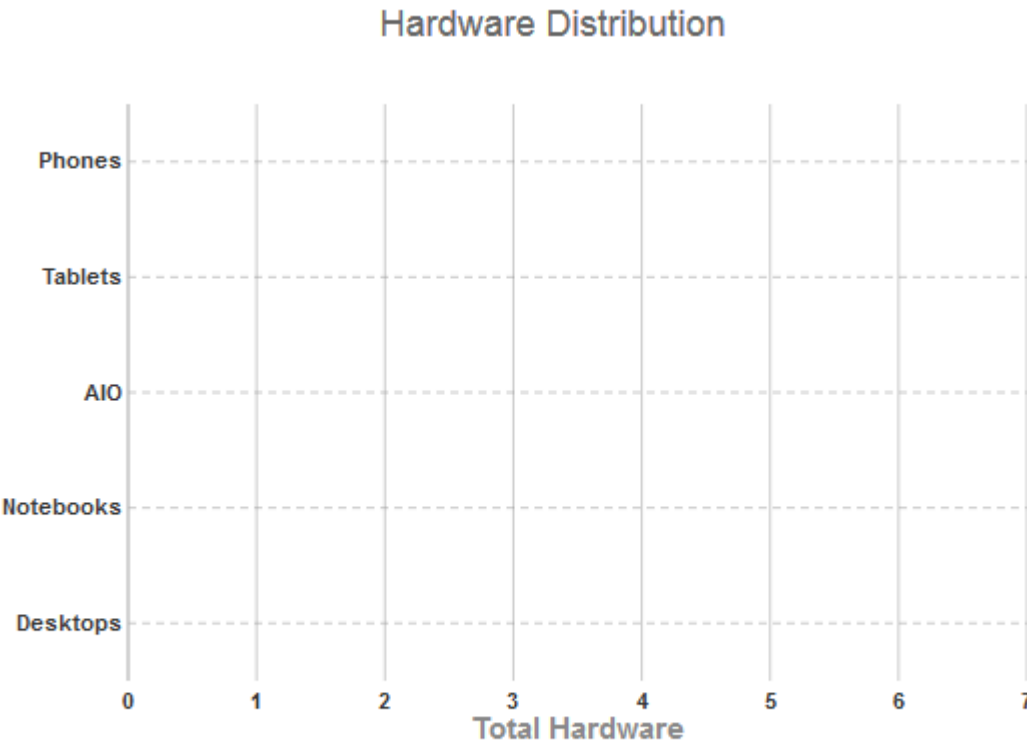
```

In this step, you added the script to handle the **C1ComboBox**'s [selectedIndexChanging](#) event and the [Hint](#) and [SeriesList](#) data for the **C1BarChart** control. In the next step, you'll run your application.

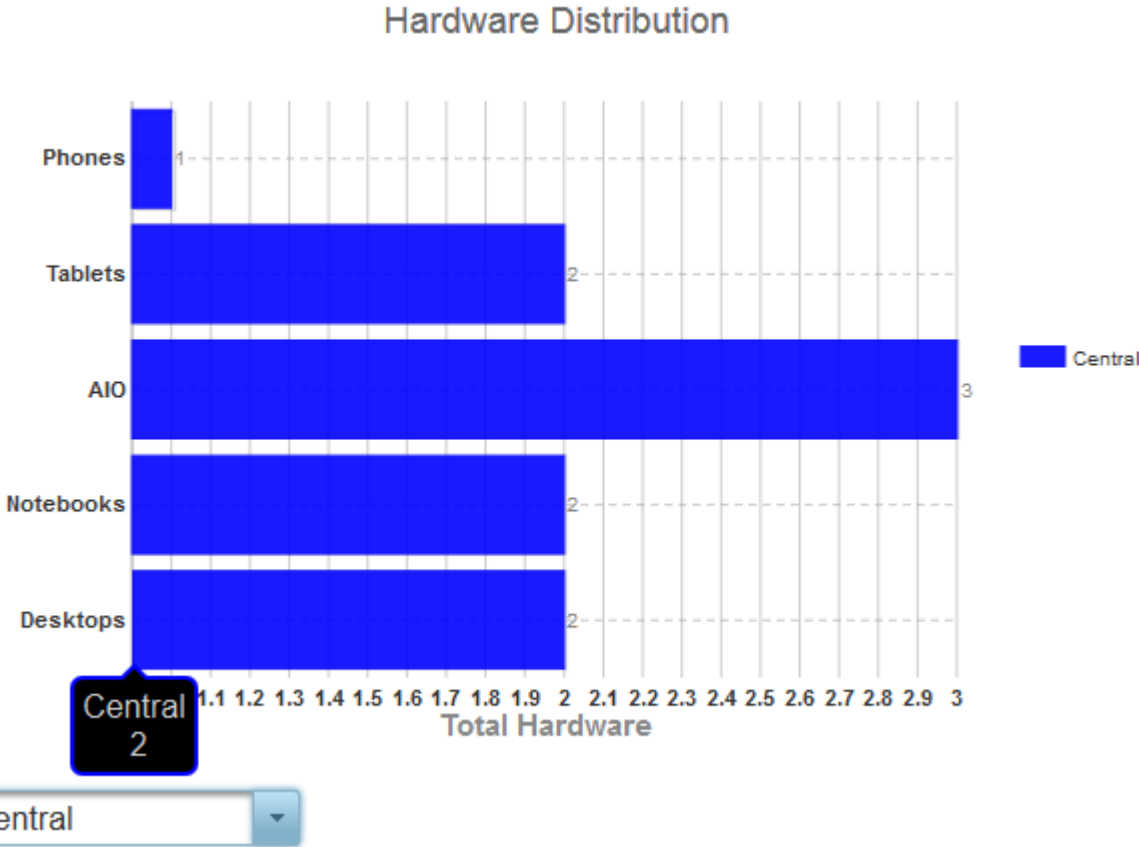
Step 4 of 4: Run Your Application

In this step, you'll run your application.

1. Press F5 or start debugging to run your application. It should initially resemble the following image:



2. Use the ComboBox to select one of the series of data. The data will be loaded into the chart. In this case, Central was chosen:



✔ Congratulations! You have completed the Loading Data Conditionally Tutorial! In this tutorial, you used both markup and script to create a BarChart application which will load data based on certain conditions chosen from a ComboBox control.

Client-Side Reference

As part of the amazing [ComponentOne Web stack](#), the Wijmo jQuery UI widgets are optimized for client-side Web development and utilize the power of jQuery for superior performance and ease of use.

The ComponentOne Wijmo website at <http://wijmo.com/widgets/> provides everything you need to know about Wijmo widgets, including demos and samples, documentation, theming examples, support and more.

The client-side documentation provides an overview, sample markup, options, events, and methods for each widget. To get started with client-side Web development for **BarChart for ASP.NET Web Forms**, click one of the external links to view our Wijmo wiki documentation. Note that the **Overview** topic for each of the widgets applies mainly to the widget, not to the server-side ASP.NET Web Forms control.

BarChart

- [wijbarchart documentation](#)
- [wijbarchart API](#)

Using the Wijmo CDN

You can easily load the client-side Wijmo widgets into your web page using a Content Delivery Network (CDN). CDN makes it quick and easy to use external libraries, and deploy them to your users. A CDN is a network of computers around the world that host content. Ideally, if you're in the United States and you access a webpage using a CDN, you'll get your content from a server based in the US. If you're in India or China, and you access the SAME webpage, the content will come from a server a little closer to your location.

When web browsers load content, they commonly will check to see if they already have a copy of the file cached. By using a CDN, you can benefit from this. If a user had previously visited a site using the same CDN, they will already have a cached version of the files on their machine. Your page will load quicker since it doesn't need to re-download your support content.

Wijmo has had CDN support from the very beginning. You can find the CDN page at <http://wijmo.com/downloads/cdn/>. The markup required for loading Wijmo into your page looks similar to this:

To write code in Source View

```
<!--jQuery References-->
<script src="http://ajax.aspnetcdn.com/ajax/jquery/jquery-1.7.1.min.js"
type="text/javascript"></script>
<script src="http://ajax.aspnetcdn.com/ajax/jquery.ui/1.8.17/jquery-ui.min.js"
type="text/javascript"></script>
<!--Theme-->
<link href="http://cdn.wijmo.com/themes/rocket/jquery-wijmo.css" rel="stylesheet"
type="text/css" title="rocket-jqueryui" />
<!--Wijmo Widgets CSS-->
<link href="http://cdn.wijmo.com/jquery.wijmo-complete.all.2.0.0.min.css"
rel="stylesheet" type="text/css" />
<!--Wijmo Widgets JavaScript-->
<script src="http://cdn.wijmo.com/jquery.wijmo-open.all.2.0.0.min.js"
type="text/javascript"></script>
<script src="http://cdn.wijmo.com/jquery.wijmo-complete.all.2.0.0.min.js"
type="text/javascript"></script>
```

In this markup, you'll notice that some of the .js files are labeled as *.min.js. These files have been minified - in other words, all unnecessary characters have been removed - to make the pages load faster. You will also notice that there are no references to individual .js files. The JavaScript for all widgets, CSS, and jQuery references have been combined into one file, respectively, such as wijmo-complete.2.0.0.min.js. If you want to link to individual .js files, see the **Dependencies** topic for each widget.

With **ASP.NET Web Forms Edition Edition** controls, you can click the **Use CDN** checkbox in the control's **Tasks** menu and specify the **CDN path** if you want to access the client-side widgets.

