Axis Tutorial

Table Of Contents

1 Axis Scales

Axis Scales

Introduction
This section will describe the following:

- **Scale Types**
  What the different axis scales offer.
- **Intervals**
  Controlling numeric and time tick intervals.
- **Scale Range**
  Controlling the axis scale start, end, and range.
- **Scale Breaks**
  Using scale breaks.
- **Scale Influentials**
  Allow axis markers and custom axis tick to influence axis scale ranges.

Scale Types

Axis scales dictate more than just the values on an axis. They also specify element behavior such as stacked, or FullStacked. The options for the **Scale enumeration** include:

- Normal
- Range
- Logarithmic
- Time
- Stacked
- FullStacked
- LogarithmicStacked

Between numeric and time scales, .netCHARTING will automatically set the appropriate scale so it does not always need to be specified explicitly. Scales such as Stacked or FullStacked only apply to axes on which the element y values are plotted because these are the values that are stacked. For example with a combo chart, the Chart.YAxis would be the one to specify a stacked scale.

Other properties that control the scale include:

- **LogarithmicBase**
  specifies the logarithmic base of intervals.
- **Percent**
  A '%' sign will be added to the end of tick labels and the axis maximum will snap to 100% if the plotted data's range falls between 50 and 100.
- **InvertScale**
  reverses the sequence of a scale.

Intervals

Intervals come in two flavors, numeric, and time. The latter is more complicated so we'll tackle numeric first.

Controlling the interval is a simple concept. You can specify a numeric interval using code such as:

```
[C#]
Chart.YAxis.Interval = 5;


[Visual Basic]
Chart.YAxis.Interval = 5
```

This will force an interval every at five numeric units. Other interval related properties include:

- **MinimumInterval**
  Specifies the minimum numeric interval.

TIP: This allows you to prevent the axis scale from using intervals that are smaller than a single unit of your data. For example a chart showing votes and the maximum number being 3 may show intervals at .5 which are not desired.

- **TickNumberMaximum**
  Specifies the maximum number of ticks to generate.

Time interval

The basic time interval is controlled with the Axis.TimeInterval property

```
[C#]
Axis.TimeInterval = TimeInterval.Week;


[Visual Basic]
Axis.TimeInterval = TimeInterval.Week
```

Advanced time interval

Or can be more specific using the Axis.TimeIntervalAdvanced property.

Multiplier

Using the advanced time interval version allows you to specify an interval multiplier. For example we can have an interval every 4 days or 2 weeks etc.

```
[C#]
Axis.TimeIntervalAdvanced.Multiplier = 2;


[Visual Basic]
Axis.TimeIntervalAdvanced.Multiplier = 2
```

Custom time span

A completely custom time span can be used as a time interval.

```
[C#]
Axis.TimeIntervalAdvanced.TimeSpan = new TimeSpan(10,5,3);


[Visual Basic]
Axis.TimeIntervalAdvanced.TimeSpan = New TimeSpan(10,5,3)
```

Custom start time

An interval can also start at any given time. These times can be specified through the following properties:

- **StartMonth**
  The month of year at which this interval initially occurs. Value ranges from zero indicating January, to eleven, indicating December.
- **StartDayOfWeek**
  The day of the week at which this interval initially occurs. Value ranges from zero indicating Sunday, to six, indicating Saturday.
- **Start**
  A DateTime object representing the time instant at which this interval initially occurs. The value of this DateTime structure can be specific down to the millisecond.

Ranges

The scale's numeric and time boundaries can be specified using the Axis.ScaleRange property.

```
[C#]
Chart.YAxis.ScaleRange.ValueHigh = 100;
Chart.YAxis.ScaleRange.ValueLow = 2;
//Or
Chart.YAxis.ScaleRange.ValueHigh = new DateTime(2000,12,1);
Chart.YAxis.ScaleRange.ValueLow = new DateTime(2000,1,1);


[Visual Basic]
```

```
Chart.YAxis.ScaleRange.ValueHigh = 100
Chart.YAxis.ScaleRange.ValueLow = 2
'Or
Chart.YAxis.ScaleRange.ValueHigh = New DateTime(2000,12,1)
Chart.YAxis.ScaleRange.ValueLow = New DateTime(2000,1,1)
```

Providing a single high or low value without its counterpart is also permitted.

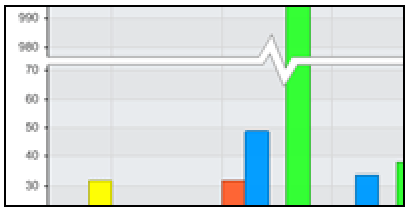
```
[C#]
YAxis.ScaleRange.ValueLow = 2;


[Visual Basic]
YAxis.ScaleRange.ValueLow = 2;
```

Note: Numeric intervals start at zero, therefore, if the minimum value doesn't land on an interval there may be a gap between the scale's edge and the first axis tick.

Time scales can be padded with an equal amount of time on either side of the plotted data to produce the final range. This is achieved using Axis.TimePadding.

```
[C#]
Chart.XAxis.TimePadding = TimeSpan.FromDays(10);


[Visual Basic]
Chart.XAxis.TimePadding = TimeSpan.FromDays(10)
```

Scale Breaks



Scale breaks are discontinuities in an axis' scale. They are useful in the following scenarios:

- When there is a large difference between low and high element values.
- When element value variations are much smaller than the scale range.
- When specific days of a week should be excluded in the scale, for example, weekends are not needed in financial charts. Scale breaks can work with a calendar pattern in this case.

  Sample: AxisScaleBreakCalendarPattern.aspx

For the first case, the chart engine can automatically generate a scale break by setting **Axis.SmartScaleBreak** to true.

Sample: AxisScaleBreaks.aspx

Scale breaks are added manually like so:

```
[C#]
Axis.ScaleBreaks.Add( new ScaleRange(0,50) );
//Or for time:
Axis.ScaleBreaks.Add( new ScaleRange(new DateTime(2000,1,1), new DateTime(2000,12,1) ) );


[Visual Basic]
Axis.ScaleBreaks.Add( New ScaleRange(0,50) )
'Or for time:
Axis.ScaleBreaks.Add( New ScaleRange(New DateTime(2000,1,1), New DateTime(2000,12,1) ) )
```

Sample: AxisScaleBreaksManual.aspx

Scale Break Styles

A number of options are available to achieve the scale break style your charts require. The style can be set with the **ScaleBreakStyle** enumeration at the axis level as shown below.

```
[C#]
Chart.YAxis.ScaleBreakStyle = ScaleBreakStyle.ZigZag;

[Visual Basic]
Chart.YAxis.ScaleBreakStyle = ScaleBreakStyle.ZigZag
```

Sample: AxisScaleBreakStyling.aspx

Scale Influentials

Objects that inherit from ScaleRange, such as AxisMarker and AxisTicks are able to influence the axis' scale range. For instance, a chart showing bandwidth usage over a period of time with an axis marker representing the bandwidth limit may have elements that will never reach or even come close to the quota marker's value. Because of this, the quota marker will not be visible on the chart; however, if AxisMarker.IncludeInAxisScale is true, the marker will tell the axis scale to encompass its value.

Objects that can influence axis scales are:

- AxisMarker
- AxisTick

Sample: AxisAffectScale.aspx

1.1 Scale Synchronization

Introduction

Multiple axes of the same type (numeric or time) and orientation (X or Y) can synchronize their scales in a number of ways. They can sync their entire scale, high or low values, and the origins. This is useful if you are using many axes on a single chart that need to sync or different axes throughout chart areas that need to stay on the same scale.

The usage is simple. Simply specify which axis you want another axis to sync with.

```
[C#]
Chart.YAxis.SynchronizeScale.Add(myNewAxis);

[Visual Basic]
Chart.YAxis.SynchronizeScale.Add(myNewAxis)
```

You can add as many axes to synchronize as you'd like. By default the entire scale will sync but if you would like to change this behavior you can specify a **SynchronizeScaleMode** enumeration.

```
[C#]
Chart.YAxis.SynchronizeScale.Mode = SynchronizeScaleMode.High;

[Visual Basic]
Chart.YAxis.SynchronizeScale.Mode = SynchronizeScaleMode.High
```

This will cause the axes to sync only the high values.

Synchronization Chain

This system also allows for complex sync chains. The above synchronizes the main y axis high value with the high value of myNewAxis. If you would like to synchronize the low value of myNewAxis with another axis you can do this also:

```
[C#]
myNewAxis.SynchronizeScale.Add(myOtherAxis);
myNewAxis.SynchronizeScale.Mode = SynchronizeScaleMode.Low;

[Visual Basic]
```

```
myNewAxis.SynchronizeScale.Add(myOtherAxis);
myNewAxis.SynchronizeScale.Mode = SynchronizeScaleMode.Low
```

This will result in different synchronizations between different axes.

1.2 Elements And Axis Scales

Introduction

While there are many options for scales, .netCHARTING can automatically determine the appropriate scales based on your data. This tutorial will demonstrate how element data influences axis scales.

The Y Axis (value axis)

We will call this the y axis but by '*Value Axis*' we don't literally mean (Y) axis. With ChartType.ComboHorizontal for instance we would be referring to the x axis. For all others however it is the y axis. The element values that influence this axis are

- YValue
- YValueStart
- YDateTime
- YDateTimeStart

The automatically chosen axis scales here are either **Normal** or **Time**. It is determined by whether the YValue (numeric) or YDateTime (time) values are specified for each element.

The following table shows value settings of these element properties that the chart engine will consider not set.

- YValue & YValueStart
  Setting: double.NaN
- YDateTime & YDateTimeStart
  Setting: DateTime.MinValue

The X Axis (Category / Value Axis)

Y Axis using ChartType.ComboHorizontal

The x axis is very powerful, it can operate just like the y axis *Value Axis* as well as a *Category Axis*. If the elements have names (Element.Name) specified, the axis scale will be a category scale. However, if the elements have either XValue or XDateTime properties set, the appropriate value axis scale will be chosen.

The element properties that influence the x axis type are:

- Name
- XValue
- XValueStart
- XDateTime
- XDateTimeStart

Even if all numeric or time value are provided, setting one name of the elements will yield a category axis scale and values will become names.

The data engine will always provide elements with both, names and values if the names represent numeric or time values. By default this creates a category scale. To use a value scale instead, set the axis scale property to the appropriate scale.

Setting the scale through DefaultAxis.Scale will not produce the same result.

Another simple shortcut to force elements to use a value axis instead of their names is to set ChartType.Scatter but this will only provide a Combo chart type.

Smart Category Axis

A *Smart Category Axis* scale is one that contains elements with names but where the names represent values. The 'smart' part will sequence the element names appropriately based on the values they represent.

To use this axis type you must use the DataEngine and specify 'XAxis=valueDBColumn' in its DataFields property.

2 Axis Ticks

Axis Ticks

Introduction
This section will describe the following:

- **Tick Types**
  Different axis tick types.
- **Custom Ticks**
  Using custom ticks and placing them at arbitrary positions.
- **String Tokens**
  Using tokens in axis ticks to describe elements or tick values.
- **Tick Overrides**
  Using ticks to override the properties of a single or multiple ticks.
- **Styling**
  Ticks styling.
- **Tick Alignment**
  Aligning tick labels with tick marks.

The **AxisTick** object encapsulates information such as its value, appearance, and grid properties. It can also be used to override a single or range of ticks on an axis.

The Axis.DefaultTick property offers a tick object that will propagate it's settings to all ticks on the axis. This provides a method of applying settings to all axis ticks quickly.

Another axis tick property is Axis.ZeroTick. This is the axis tick that is added to each numeric axis at its origin (0). This tick can be disabled by setting the property to null.

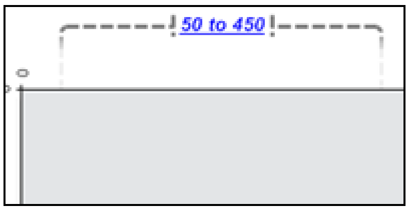
```
[C#]
Chart.XAxis.ZeroTick = null;

[Visual Basic]
Chart.XAxis.ZeroTick = Null
```

Modes
An axis tick can contain a numeric or time value as well as a string value depending on the type of axis used.

Normal Axis Ticks

```
[C#]
AxisTick at = new AxisTick();
At.Value = 10;

[Visual Basic]
Dim at As New AxisTick()
At.Value = 10
```

Range Ticks

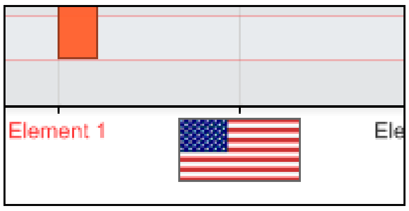


The tick can also represent a numeric or time range.

```
[C#]
At.ValueHigh = 20;
At.ValueLow = 10;
```

```
[Visual Basic]
At.ValueHigh = 20
At.ValueLow = 10
```

Marker Ticks



Both value and range ticks can be represented by an **ElementMarker** instead of a label.

```
[C#]
At.Marker = new ElementMarker("images/myImage.gif");
[Visual Basic]
At.Marker = New ElementMarker("images/myImage.gif")
```

Custom / Automatic ticks

The **element** object contains **XAxisTick** and **YAxisTick** properties. These are null by default, however, when instantiated, they will be drawn on the axis the elements belong to and will represent the element's value.

Sample: ElementTicks.aspx

The AxisMarker also contains a single AxisTick property. When instantiated, it will represent the marker's value or range as a range tick, depending on the marker's value.

Sample: AxisMarkerTicks.aspx

AxisTick.AffectAxisScale can by used by custom ticks to affect the scale of the axis its added to. See Axis Scale > Scale Influentials above.

Adding Custom Ticks to an axis:

Custom axis ticks can be added to any value axis like so:

```
[C#]
Chart.XAxis.ExtraTicks.Add(At);

[Visual Basic]
Chart.XAxis.ExtraTicks.Add(At)
```

To quickly add multiple axis ticks you can also list them as parameters in the Add method.

```
[C#]
Chart.XAxis.ExtraTicks.Add(at1, at2, at3, at4);

[Visual Basic]
Chart.XAxis.ExtraTicks.Add(at1, at2, at3, at4)
```

Tokens

Tick labels on a category axis can use tokens to display information about element groups they represent. For example the default category tick's label is "%Name" which will show the element's name.

Ticks assigned to elements (Element.YAxisTick & Element.XAxisTick) can also use tokens to describe the elements they represent.

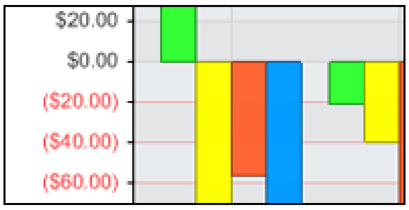
Sample: elementTemplate.aspx

(New in 3.1) Tick labels in a numeric or time axis can also use tokens. A single value tick uses the token '%Value'. A range tick can use either the value token or any tokens related to the ScaleRange object. This functionality is useful with time scales because it allows you to break the time label into multiple lines as the following sample demonstrates:

Sample: AxisTickToken.aspx

See also: **Token Reference ('Tokens' in the on-line documentation)**

Overrides



A tick added to an axis that already contains a tick with the same value will update the old tick properties with its own. For example, if an axis shows a tick at 20 we can override it to display some different text using this code:

```
[C#]
AxisTick at = new AxisTick();
At.Value = 20;
At.Label.Text = "Twenty";
XAxis.ExtraTicks.Add(At);

[Visual Basic]
Dim at As New AxisTick()
At.Value = 20
At.Label.Text = "Twenty"
XAxis.ExtraTicks.Add(At)
```

If a tick at 20 didn't exist, it would be added as an additional tick.

A tick can also be used to propagate its settings to a range of ticks. Consider a scale from -100 to 100. If we wanted to color all the negative values red we could do so with the following code:

```
[C#]
AxisTick at = new AxisTick();
At.ValueHigh = -1;
At.ValueLow = -100;
At.Label.Color = Color.Red;
At.OverrideTicks = true;
XAxis.ExtraTicks.Add(At);

[Visual Basic]
Dim at As New AxisTick()
At.ValueHigh = -1
At.ValueLow = -100
At.Label.Color = Color.Red
At.OverrideTicks = True;
XAxis.ExtraTicks.Add(At)
```

Notice that the AxisTick.OverrideTick setting is required so the axis knows whether to treat the tick as an override or regular range tick.

Styling

The **AxisTick** object can control its appearance as well as the grid line that stems from it. The TickLine property can also take advantage of the Line.Length property.

Some styling examples:

```
[C#]
AxisTick at = new AxisTick();
At.Label.Font = new Font("Verdana",10);
At.Line.Length = 8;
At.GridLine.DashStyle = Dash;
```

```
[Visual Basic]
Dim at As New AxisTick()
At.Label.Font = New Font("Verdana",10)
At.Line.Length = 8
At.GridLine.DashStyle = Dash
```

The axis contains properties that also modify the axis tick appearance.

- **TickLabelMode**
  This enumeration allows you to specify the layout behavior of ticks.
- **TickLabelPadding**
  A numeric value in pixels that indicates the distance between a tick line and the label.
- **TickLabelSeparatorLine**
  Between each tick mark a line can be drawn to accentuate tick label separation. This line is automatically enabled when tick labels decide to wrap.

Tick Alignment
Tick marks can also be drawn between each tick label instead of above it. This is done by setting Axis.CenterTickMarks to false. When this is the case, only Axis.DefaultTick properties are used instead of individual tick line properties because not every tick mark will pertain to a label.

Tip: Centered tick marks are commonly used with category axes [Group A , Group B, …] and sometimes with time axes [May, June, …]

2.1 Minor Ticks

Introduction

Minor axis ticks are used to divide scale intervals into smaller logical intervals. This can be done automatically or specified manually.

Smart Minor Ticks (Automatic)

Smart minor ticks will automatically determine a logical minor interval considering the scale's major intervals. For example, a time interval of three months will automatically place a minor tick for every month. To use this feature simply set an axis' **SmartMinorTicks** property to true.

```
[C#]
Chart.XAxis.SmartMinorTicks = true;

[Visual Basic]
Chart.XAxis.SmartMinorTicks = True
```

Sample: AxisSmartMinorTicks.aspx

- Smart minor ticks are only generated when the interval is determined automatically as apposed to explicitly specified .
- If smart minor ticks are less than the specified **Axis.MinimumInterval**, they will not be generated.

Manual Minor Ticks

Minor tick intervals can be controlled through the following axis properties:

- **MinorInterval**
  Gets or sets a numeric value that determines the interval of minor axis ticks.
- **MinorTicksPerInterval**
  Gets or sets the number of minor axis ticks generated between major axis ticks.
- **MinorTimeIntervalAdvanced**
  Gets or sets a **TimeIntervalAdvanced** object that determines the interval of minor axis ticks on a time scale.

Minor Tick Manipulation

While the minor ticks don't traditionally have labels or any special styling, in .netCHARTING terms it's just another axis tick. If needed, the minor ticks can have labels, extra long or thick lines etc. All the functionality of an axis tick is available to minor ticks. One difference is that minor ticks will not influence the grid with regards to alternating grid colors of an axis (**AlternateGridBackground**).

Sample: AxisMinorTicksTrick.aspx

⚠ Logarithmic axis scales do not support minor ticks.

2.2 Axis Time Label Automation

Introduction

This feature is yet another charting innovation brought to you only by .netCHARTING. Axis time label automation is a collection of smart features that accurately describe the time segment shown on a time axis scale. With these user friendly and automatic features, your time based charts will be more intuitive and readable.

How smart is it?

Simply put, quite. This feature is aware of things such as whether your time span crosses into another day, month, or year and will point these things out to the viewer. It is also aware of how specific your data is; meaning, whether it's describing minutes, days, months and so on. This is a complicated task with countless permutations where you may require a behavior other than the default. Fortunately, this feature is also very well equipped with plenty of options that allow meticulous customization.

If this sounds useful for your projects, let's explore the topic a bit further.

Hierarchy & API

There are two separate mechanisms involved with time label automation. One generates single value labels and the other generates range ticks. The behavior or modes of these features are controlled with two properties:

- **Axis.TimeScaleLabels.Mode** (Single Value Mechanism)
  Enumeration Members: **TimeScaleLabelMode**
- **Axis.TimeScaleLabels.RangeMode** (Range Value Mechanism)
  Enumeration Members: **TimeScaleLabelRangeMode**

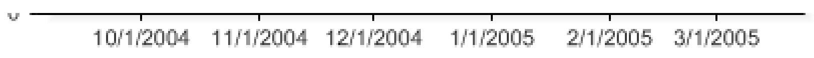
Additional options of these features are available through the **Axis.TimeScaleLabels** property.

Single Value Time Labels

Dynamic Mode

An automatically determined or specified time interval can be used with this mode. This mode will determine what the axis ticks are describing and apply the format strings and axis tick properties that correspond to the time instance of each axis tick. The format strings and axis ticks are found in the **TimeScaleLabelInfo** class and are covered in depth below.

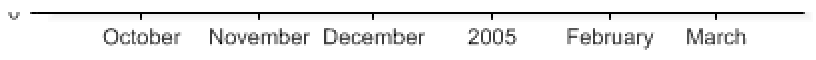
Let's start with an example. This is a normal time axis without any time automation. Only the time interval is automatically determined:



Now let's see how the automation engine will handle this scale in 'dynamic' mode.

```
[C#]
Chart.XAxis.TimeScaleLabels.Mode = TimeScaleLabelMode.Dynamic;
```

```
[Visual Basic]
Chart.XAxis.TimeScaleLabels.Mode = TimeScaleLabelMode.Dynamic
```



As you can see, the ticks are evaluated and for the most part have been determined to represent months, therefore, they are labeled with the month's name. One of the ticks, however, represents the year and is formatted to show the four digit year.

Smart Mode

The above, 'dynamic mode' uses static formats specified in the **TimeScaleLabelInfo** class. The smart mode will do the same, however, only if they are specified by the user. Otherwise the formats will be determined automatically based on a number of factors, resulting in a detailed and well laid out axis. Considering all the different factors, the smart time automation engine will choose from over 100 formats and templates to produce the most appropriate result.

```
[C#]
Chart.XAxis.TimeScaleLabels.Mode = TimeScaleLabelMode.Smart;
```

```
[Visual Basic]
Chart.XAxis.TimeScaleLabels.Mode = TimeScaleLabelMode.Smart
```

Hidden Mode

This mode will prevent any single value axis ticks from being shown on the axis. It is useful in conjunction with range time labels when the range time labels sufficiently describe the time segment and single value ticks are not desired.

```
[C#]
Chart.XAxis.TimeScaleLabels.Mode = TimeScaleLabelMode.Hidden;
```

```
[Visual Basic]
Chart.XAxis.TimeScaleLabels.Mode = TimeScaleLabelMode.Hidden
```

Ranged Time Labels

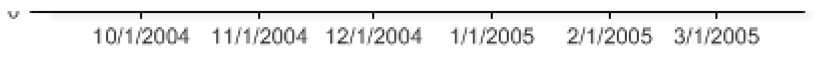
Default Mode

The default mode does not generate any axis ticks, unless, the time intervals to base those axis ticks on are specified. Adding range time intervals is explained below.

Dynamic Mode

When the TimeScaleLabelRangeMode.Dynamic mode is specified, the time span is analyzed to determine what time intervals should be used to generate ranged axis ticks. The range ticks are then generated based on those intervals. If range time intervals are specified by the user this mode will not generate axis ticks based on any additional automatic time intervals. In effect, it will behave like the **Default** mode.

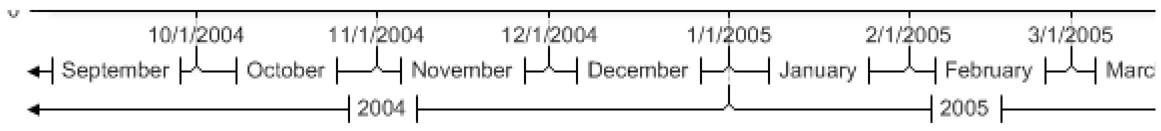
Consider the following axis scale:



The chart will be widened below so everything is shown properly.

```
[C#]
Chart.XAxis.TimeScaleLabels.RangeMode = TimeScaleLabelRangeMode.Dynamic;

[Visual Basic]
Chart.XAxis.TimeScaleLabels.RangeMode = TimeScaleLabelRangeMode.Dynamic
```

The resulting axis:

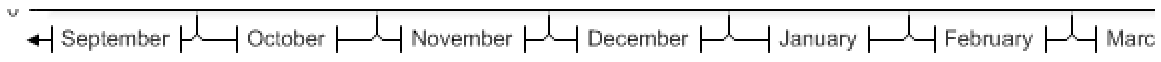


Because the axis ticks represent months and span into a different year, range axis ticks were created for both of these units of time.

Specifying Range Time Intervals

Time intervals can be added using the **Axis.TimeScaleLabels.RangeIntervals** property. When specified, they are always generated and the dynamically determined time intervals are omitted. This code snippet will also demonstrate using *TimeScaleLabelMode.Hidden* to hide the single value ticks.

```
[C#]
Chart.XAxis.TimeScaleLabels.Mode = TimeScaleLabelMode.Hidden;
Chart.XAxis.TimeScaleLabels.RangeIntervals.Add(TimeInterval.Month);

[Visual Basic]
Chart.XAxis.TimeScaleLabels.Mode = TimeScaleLabelMode.Hidden
Chart.XAxis.TimeScaleLabels.RangeIntervals.Add(TimeInterval.Month)
```



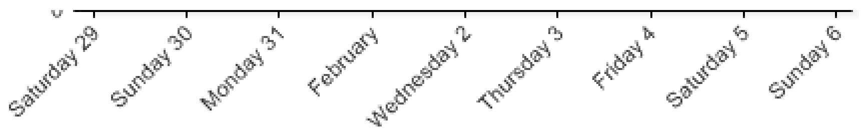
TIP: Multiple range intervals can be added simultaneously using a single line of code:
Chart.XAxis.TimeScaleLabels.RangeIntervals.Add(TimeInterval.Months, TimeInterval.Years)

Custom Styling and Formatting

The time label automation mechanism will describe the time segment accurately, but, it may not describe it exactly as you'd like. This section describes the options available for this customization.

The axis property **TimeScaleLabels** is an **Axis.TimeScaleLabelInfo** class. This class contains format strings and axis tick objects that correspond to time units used by these features. Because time automation is based mainly on time instances, the generated labels can be fully controlled through objects you may already be familiar with.

For example, let's consider this is our time scale using dynamic mode:

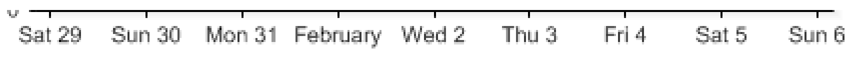


Format Strings

The labels are long and this is causing them to angle. The day labels don't NEED to spell the whole word out, therefore, the days format string will be modified to only abbreviate the day names using the following code:

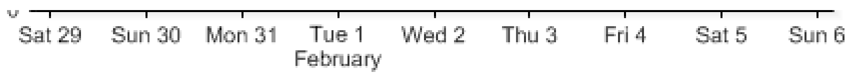
```
[C#]
Chart.XAxis.TimeScaleLabels.DayFormatString = "ddd d";
```

```
[Visual Basic]
Chart.XAxis.TimeScaleLabels.DayFormatString = "ddd d"
```



Label Templates
Now that the labels aren't angled, the there is another thing that stands out a bit, or rather, doesn't stand out enough. The February tick can be easily missed and it doesn't tell us what day it is describing. If we change its format to show all this information, this label will be long and cause the labels to angle again. To overt this problem, we will instead manipulate the month's axis tick label to show all the information we need and add a new line character to draw the label on two lines and keep it narrow.

```
[C#]
Chart.XAxis.TimeScaleLabels.MonthTick.Label.Text = "<%Value,ddd d>\n<%Value,MMMM>";

[Visual Basic]
Chart.XAxis.TimeScaleLabels.MonthTick.Label.Text = "<%Value,ddd d>\n<%Value,MMMM>"
```

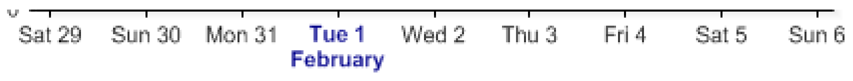


Tip: It may also be useful to use a smaller font to create more room for axis ticks.

Styling

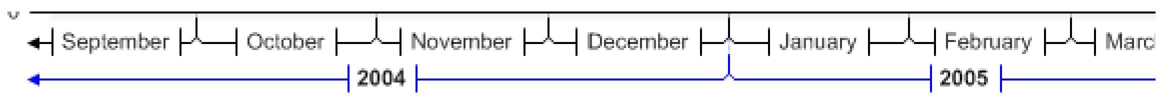
Now the scale looks much better. Still, for those of you who always give 110%, let's go a step further and make those month labels really stand out. As we've seen in the above example, an AxisTick is exposed for each time unit. This gives us complete control over its appearance. This code will modify the month label's font and color.

```
[C#]
Chart.XAxis.TimeScaleLabels.MonthTick.Label.Font = new Font("Arial",8,FontStyle.Bold);
Chart.XAxis.TimeScaleLabels.MonthTick.Label.Color = Color.DarkBlue;

[Visual Basic]
Chart.XAxis.TimeScaleLabels.MonthTick.Label.Font = New Font("Arial",8,FontStyle.Bold)
Chart.XAxis.TimeScaleLabels.MonthTick.Label.Color = Color.DarkBlue
```



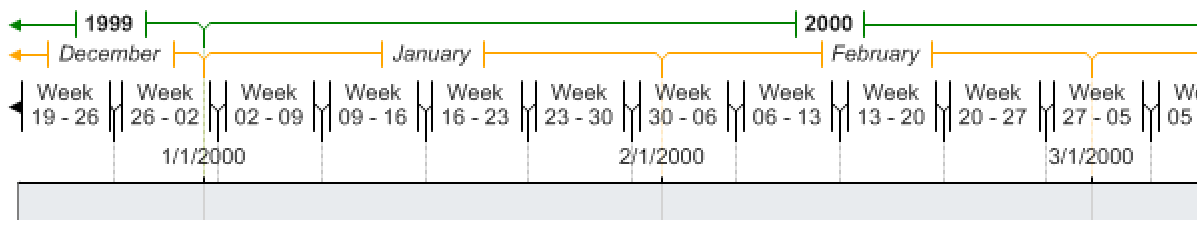
Range ticks can be customized in the same manner as the single value ticks. This code modifies the YearTick's font and line color.

```
[C#]
Chart.XAxis.TimeScaleLabels.YearTick.Label.Font = new Font("Arial",8,FontStyle.Bold);
Chart.XAxis.TimeScaleLabels.YearTick.Line.Color = Color.Blue;

[Visual Basic]
Chart.XAxis.TimeScaleLabels.YearTick.Label.Font = new Font("Arial",8,FontStyle.Bold)
Chart.XAxis.TimeScaleLabels.YearTick.Line.Color = Color.Blue
```



Test your skills

Now that you're a time label automation expert, let's test your skills and see if you can figure out the code used to generate this chart axis:



Answer (code)

```
[C#]
 Chart.XAxis.FormatString = "d";

 Chart.XAxis.TimeLabelAutomation.RangeIntervals.Add(TimeInterval.Weeks, TimeInterval.Montl
 Chart.XAxis.TimeLabelAutomation.WeekTick.Label.Text = "Week\n<%Low,dd> - <%High,dd>";
 Chart.XAxis.TimeLabelAutomation.MonthTick.Line.Color = Color.Orange;
 Chart.XAxis.TimeLabelAutomation.YearTick.Line.Color = Color.Green;
 Chart.XAxis.TimeLabelAutomation.YearTick.Label.Font = new Font("Arial",8,FontStyle.Bold),
 Chart.XAxis.TimeLabelAutomation.MonthTick.Label.Font = new Font("Arial",8,FontStyle.Ital:
```

```
[Visual Basic]
 Chart.XAxis.FormatString = "d"

 Chart.XAxis.TimeLabelAutomation.RangeIntervals.Add(TimeInterval.Weeks, TimeInterval.Montl
 Chart.XAxis.TimeLabelAutomation.WeekTick.Label.Text = "Week\n<%Low,dd> - <%High,dd>"
 Chart.XAxis.TimeLabelAutomation.MonthTick.Line.Color = Color.Orange
 Chart.XAxis.TimeLabelAutomation.YearTick.Line.Color = Color.Green
 Chart.XAxis.TimeLabelAutomation.YearTick.Label.Font = new Font("Arial",8,FontStyle.Bold)
 Chart.XAxis.TimeLabelAutomation.MonthTick.Label.Font = new Font("Arial",8,FontStyle.Ital:
```

2.3 Axis Calculated Ticks

Introduction

Previous to version 3.2, to create an axis tick representing a value such as the average of the plotted data, the value had to be calculated, then, an axis tick was manually added to the axis. This process has now been simplified down to one simple line of code.

Calculated Axis Tick

The following code adds a calculated tick to an axis.

```
[C#]
Chart.YAxis.AddCalculatedTick(Calculation.Average);
```

```
[Visual Basic]
Chart.YAxis.AddCalculatedTick(Calculation.Average)
```

As you can see, this code does not specify any data to perform the calculation on. In this case, the data plotted on the axis is calculated.

Data Specific

A calculation based on only one series of the data shown on an axis is often desired. When this is the case, a single series or any number of series can be specified to base the calculation on.

The following code demonstrates specifying a series:

```
[C#]
Chart.YAxis.AddCalculatedTick(Calculation.Average, mySeries);

[Visual Basic]
Chart.YAxis.AddCalculatedTick(Calculation.Average, mySeries)
```

To specify multiple series, place them into a series collection which can be specified as a method parameter.

Using a template

The resulting axis tick will show the calculated value, however, if additional formatting or textual modifications are required, a template can be specified. The '%Value' token in the template will be replaced with the resulting value. The following examples demonstrate template usage.

```
[C#]
Chart.YAxis.AddCalculatedTick("<%Value,Currency>", Calculation.Average);
Chart.YAxis.AddCalculatedTick("Average: %Value", Calculation.Average);
// Resulting Ticks:
// $123
// Average: 123

[Visual Basic]
Chart.YAxis.AddCalculatedTick("<%Value,Currency>", Calculation.Average)
Chart.YAxis.AddCalculatedTick("Average: %Value", Calculation.Average)
' Resulting Ticks:
' $123
' Average: 123
```

Calculated Ticks of Shadow Axes (Section 4.1)

Calculated axis ticks can also be generated for shadow axes. Despite the data not being bound to the actual axis, a shadow axis will base it's calculations on the parent axis' data.

A common task could be to have an original axis show the scale, and a shadow axis on the other side of the chart can show the calculated axis ticks as shown below.

```
[C#]
Axis axisReplica = Chart.XAxis.Calculate("Replica",true,Orientation.Right);
axisReplica.AddCalculatedTick(Calculation.Average);
Chart.AxisCollection.Add(axisReplica);

[Visual Basic]
Dim axisReplica As Axis = Chart.XAxis.Calculate("Replica",True,Orientation.Right)
axisReplica.AddCalculatedTick(Calculation.Average)
Chart.AxisCollection.Add(axisReplica)
```

Sample: AxisShadowCalculatedTicks.aspx

3 Axis Markers

Introduction

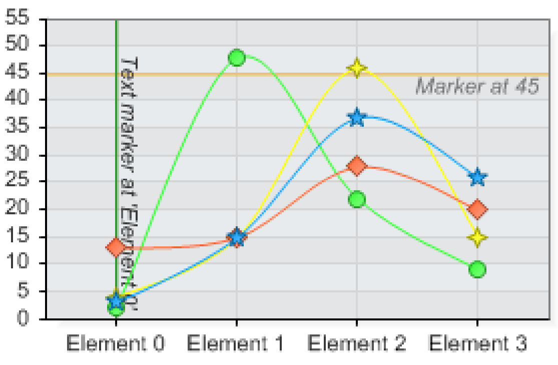
The **AxisMarker** class can be used to highlight parts of a chart area along an axis. The axis marker can have a single value, a range, or even a **CalendarPattern**. An axis marker can also be attached to an element in cases where the marker is wanted on a *Category Axis*.

Axis markers can also be drawn on top of the chart elements when in 2D mode. This behavior can be specified by setting the **AxisMarker.BringToFront** property to true.

Single Value AxisMarker

An axis marker can take a number, DateTime, or string value to define it's position. This sample code snippet uses a number and string to insert markers.

```
[C#]
AxisMarker am1 = new AxisMarker("Marker at 45",new Line(Color.Orange),45);
Chart.YAxis.Markers.Add(am1);
AxisMarker am2 = new AxisMarker("Text marker at 'Element 0'",new Line(Color.Green),"Elemen
Chart.XAxis.Markers.Add(am2);

[Visual Basic]
Dim am1 As New AxisMarker("Marker at 45",new Line(Color.Orange),45)
Chart.YAxis.Markers.Add(am1)
Dim am2 As New AxisMarker("Text marker at 'Element 0'",new Line(Color.Green),"Element 0")
Chart.XAxis.Markers.Add(am2)
```

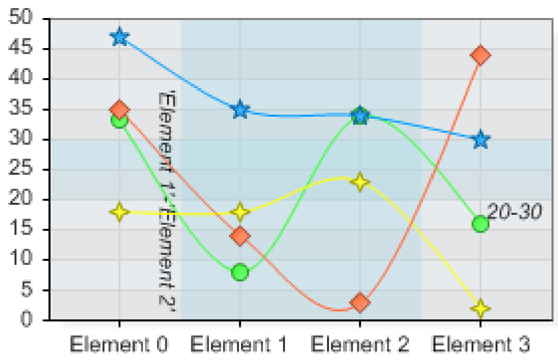


Range AxisMarker

Axis markers can also highlight a range on the chart. Even a string range.

⚠ String ranges are determined by string comparison where 2 is larger than 12.

```
[C#]
AxisMarker am3 = new AxisMarker("20-30",new Background(Color.FromArgb(100,Color.LightBlue
Chart.YAxis.Markers.Add(am3);
AxisMarker am4 = new AxisMarker("'Element 1'-'Element 2'",new Background(Color.FromArgb(1
am4.Label.LineAlignment = StringAlignment.Near;
Chart.XAxis.Markers.Add(am4);

[Visual Basic]
Dim am3 As New AxisMarker("20-30",New Background(Color.FromArgb(100,Color.LightBlue)),20,
Chart.YAxis.Markers.Add(am3)
Dim am4 As New AxisMarker("'Element 1'-'Element 2'",New Background(Color.FromArgb(100,Col
am4.Label.LineAlignment = StringAlignment.Near
Chart.XAxis.Markers.Add(am4)
```

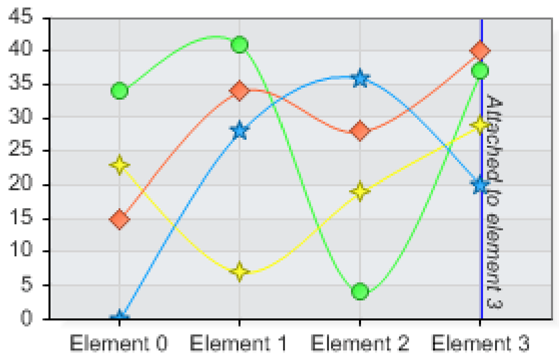


Attached to an element

To attach an axis marker to an element simply instantiate one for the element in question.

Markers attached to an element only appear on category axes and only if one is available.

```
[C#]
mySC[0].Elements[3].AxisMarker = new AxisMarker("Attached to element 3",new Line(Color.Bl

[Visual Basic]
mySC[0].Elements[3].AxisMarker = New AxisMarker("Attached to element 3",new Line(Color.Bl
```



AxisMarkerAdvanced.aspx

Calendar Pattern AxisMarker

A **calendar pattern** can be used to specify sections of time the axis marker is drawn on a time axis.

This example highlights the weekdays on a time x axis:

```
[C#]
CalendarPattern cp = new CalendarPattern(TimeInterval.Day,TimeInterval.Week,"0111110");
cp.AdjustmentUnit = TimeInterval.Day;
AxisMarker am = new AxisMarker("",new Background(Color.FromArgb(100,Color.Orange)),0,0);
am.CalendarPattern = cp;
am.LegendEntry = new LegendEntry("Week Days","");
Chart.XAxis.Markers.Add(am);

[Visual Basic]
Dim cp As New CalendarPattern(TimeInterval.Day,TimeInterval.Week,"0111110")
cp.AdjustmentUnit = TimeInterval.Day
Dim am As New AxisMarker("",New Background(Color.FromArgb(100,Color.Orange)),0,0)
am.CalendarPattern = cp
am.LegendEntry = New LegendEntry("Week Days","")
Chart.XAxis.Markers.Add(am)
```

See Also: **Calendar Pattern Tutorial ('Calendar Patterns' in the on-line documentation)**

Sample: calendarPattern.aspx

4 Multiple Axes

Multiple Axes

.netCHARTING employs a straightforward yet effective system to utilize multiple axes. Each series can have it's own x and y axis. By default each series on a chart inherits the main x and y axis

Main Axes

- Chart.YAxis
- Chart.XAxis

Axes are defined for series using the following properties:

- Series.XAxis
- Series.YAxis

The relationships and identification of axes is maintained by axis object instances. This means instead of specifying an axis name you would specify it by using the axis object itself.

For example, here both series will use the same x axis:

```
[C#]
Series1.XAxis = new Axis();
Series2.XAxis = Series1.XAxis;

[Visual Basic]
Series1.XAxis = New Axis()
Series2.XAxis = Series1.XAxis
```

This example will show how two different axes are specified for two series.

```
[C#]
DataEngine de = new DataEngine();
//(... get data into the engine ...)
SeriesCollection sc = de.GetSeries();
// Let's assume the data engine generated 2 series.
// Specify different axes for each series.
sc[0].XAxis = new Axis();
sc[0].YAxis = new Axis();
sc[1].XAxis = new Axis();
sc[1].YAxis = new Axis();
// Add the series collection to the chart.
Chart.SeriesCollection.Add(sc);



[Visual Basic]
Dim de As New DataEngine()
'(... get data into the engine ...)
Dim sc As SeriesCollection = de.GetSeries()
' Let's assume the data engine generated 2 series.
' Specify different axes for each series.
sc(0).XAxis = New Axis()
sc(0).YAxis = New Axis()
sc(1).XAxis = New Axis()
sc(1).YAxis = New Axis()
' Add the series collection to the chart.
Chart.SeriesCollection.Add(sc)
```

Samples:

- RateChart.aspx
- SeriesAxisBinding.aspx
- SeriesAxisBindingDB.aspx

- InvisibleAxis.aspx

See also: ***Z Axis effect using multiple axes (Section 5.1)***

4.1 Calculated Axes

Introduction

A Calculated axis is a *Shadow Axis* that allows you to add an axis to the chart that mimics another axis that already exists but with a change in behavior.

The different behavioral changes and forms of shadow axes are:

- Numeric Unit Conversion
- Time Interval Conversion
- Custom tick label manipulation method
- Completely custom replica axis.

⚠ Calculated axes do not fill alternating grid areas.

Unit Conversion

.netCHARTING is equipped with calculations for converting between 1092 different units. If you have an axis with a scale that represents a some measure, it can be easily converted to another unit. For example feet to inches and so on.

The process for adding a calculated axis is simple:

- Instantiate a calculated axis by calling the **Calculate** method of an existing axis.
- Add it to the Chart.**AxisCollection** collection.

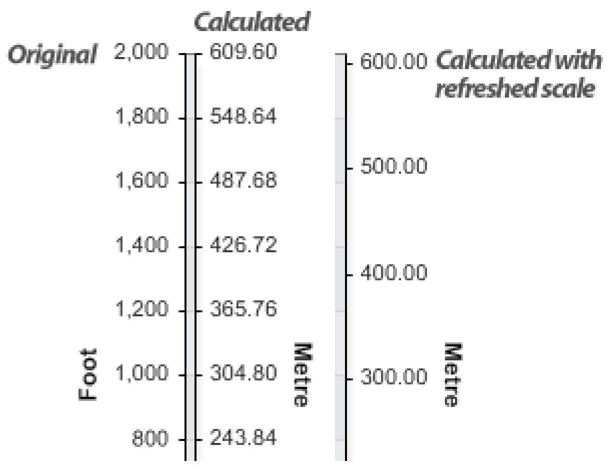
```
[C#]
Axis inchAxis = Chart.YAxis.Calculate("Inches",Length.Feet, Length.Inch);
Chart.AxisCollection.Add(inchAxis);
```

```
[Visual Basic]
Dim inchAxis As Axis = Chart.YAxis.Calculate("Inches",Length.Feet, Length.Inch)
Chart.AxisCollection.Add(inchAxis)
```

That is all. The axis will automatically show up on the chart by the original axis.

⚠ Do not add axes to AxisCollection that are not calculated axes. See **Axis Tutorial > Multiple Axes (Section 4)** for information on how additional axes can be used.

There is another feature worth discussing. The new converted unit axis will by default take the parent axis ticks and convert them to the specified unit. The refresh scale option will allow the axis to come up with its own intervals. They will not match the parent axis tick positions exactly but will behave as if it is the original axis. This option is specified in the **Calculate** method as a parameter and illustrated below.



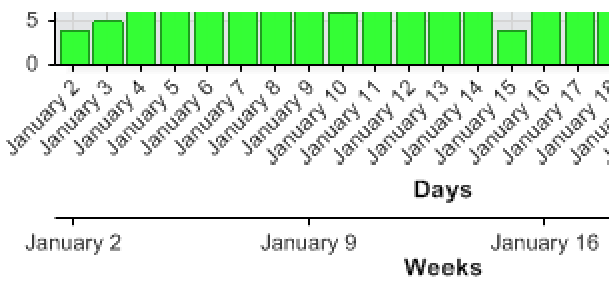
Sample: interactiveUnitConversion.aspx

Time Interval Conversion

This type of conversion allows the resulting axis to show different time intervals than the original axis. For example, the first axis can show days and the calculated axis can show weeks.

```
[C#]
Chart.XAxis.TimeInterval = TimeInterval.Days;
Chart.XAxis.Label.Text = "Days";
Chart.AxisCollection.Add(Chart.XAxis.Calculate("Weeks",TimeInterval.Weeks));
```

```
[Visual Basic]
Chart.XAxis.TimeInterval = TimeInterval.Days
Chart.XAxis.Label.Text = "Days"
Chart.AxisCollection.Add(Chart.XAxis.Calculate("Weeks",TimeInterval.Weeks))
```



Sample: CalculatedTimeAxis.aspx

Custom Tick Label Manipulation Method

The third conversion requires a custom method that takes a string and returns a processed string. This gives you full control over the axis tick label text if necessary.

Sample: customFunction.aspx

To only show the calculated axis on the chart and not it's parent, use *ParentAxis*.Clear()

to make it disappear as shown in the customFunction.aspx sample.

Replica Shadow Axis (New in 3.1)

This form of shadow axis can be very useful because it allows you to specify data related axis ticks that help describe your data. This code demonstrates how to acquire the replica axis.

```
[C#]
Axis axisReplica = Chart.XAxis.Calculate("Replica");
Chart.AxisCollection.Add(axisReplica);
```

```
[Visual Basic]
Dim axisReplica As Axis = Chart.XAxis.Calculate("Replica")
Chart.AxisCollection.Add(axisReplica)
```

Some possible uses of this type of shadow axis are shown in the following samples.

- AxisShadowImportantTicks.aspx
- AxisShadowSeriesNames.aspx
- AxisShadowValues.aspx
- AxisShadowValues2.aspx

After reviewing the above samples you will see that this feature can be useful, however, don't limit yourself to the demonstrated possibilities. The dynamic nature of this feature allows for countless different ways to better describe your data you're plotting.

5 Element Layout Control

Element Layout Control

Introduction
This section will describe the following:

- **Sizing Columns**
  How axes influence the element behavior and appearance.
- **Clustering Columns**
  How axes control column and cylinder clustering.

Besides the obvious, an axis can also control how elements drawn on its scale behave. These behavioral features are determined by the axis that shows element names or x values. For a vertical combo chart this would mean the x axis and y axis for horizontal combo.

Column and Cylinder Widths

- **SpacingPercentage**
  Gets or sets a percentage (0 - 100) which indicates the spacing between columns, cylinders or groups thereof.
- **StaticColumnWidth**
  Gets or sets the static bar width in pixels.

Example: This code specifies the column or cylinder widths for a ChartType.Combo chart.

```
[C#]
Chart.XAxis.SpacingPercentage = 30; // Default is 16
// When this property is set, it takes precedence over spacing percentage.
Chart.XAxis.StaticColumnWidth = 20;


[Visual Basic]
Chart.XAxis.SpacingPercentage = 30 ' Default is 16
' When this property is set, it takes precedence over spacing percentage.
Chart.XAxis.StaticColumnWidth = 20
```

Tip: The column width control also defines the tool tip hotspot width of area line series types.

Clustering Columns / Cylinders
This feature enables columns in 3D modes to cluster (draw side by side) or not (draw one in front of the other). The default behavior is to cluster and column must be clustered in 2D mode.

```
[C#]
Chart.Use3D = true;
Chart.XAxis.ClusterColumns = false;


[Visual Basic]
Chart.Use3D = True
Chart.XAxis.ClusterColumns = False
```

Other options include

- **Position**
  The axis positions when 2 or more axes are drawn on the same side of a ChartArea.
- **ReverseSeriesPositions**
  Indicates whether the positions of series bound to this axis are reversed without reversing legend positions.
- **ReverseSeries**
  Indicates whether the positions of series bound to this axis are reversed.
- **ReverseStack**
  Indicates whether the order of stacked elements is reversed.

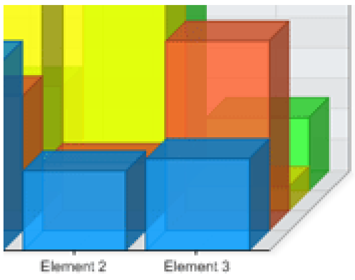
5.1 Advanced: Z Axis Effect

Z-Axis Effect

Using multiple axes with series can yield some interesting results you may not be aware of. We'll explore a situation that simulates a z axis. A basic z axis can be simulated using

```
[C#]
Chart.XAxis.ClusterColumns = false;


[Visual Basic]
Chart.XAxis.ClusterColumns = false
```

This may show a chart that looks like this:

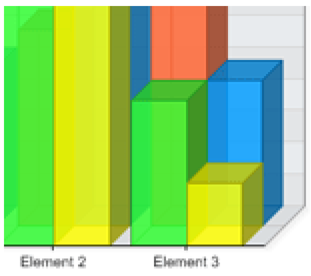


Using X Axes

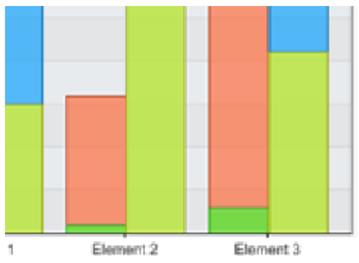
This works well however if we wanted a z axis with two steps and two clustered series on each step we will have to use two X axes. This time we will also omit setting the cluster columns property to false.

```
[C#]
Axis a2 = new Axis();
mySC[0].XAxis = a2;
mySC[1].XAxis = a2;
a2.Clear();


[Visual Basic]
Dim a2 As New Axis()
mySC(0).XAxis = a2
mySC(1).XAxis = a2
a2.Clear()
```

We "Clear()" the second axis so that only the original one is visible.



Tip: This method also allows overlapping c
in 2D mode which is useful in Gantt charts



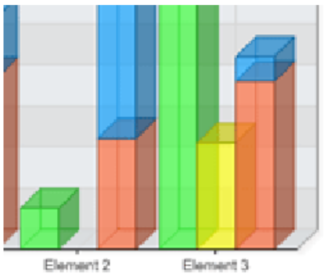
Using Y Axes

Y axes will typically not affect the z axis. For example using the following code where we give two of the four series a new y axis and set the scale to stacked:

```
[C#]
Axis a2 = new Axis();
mySC[0].YAxis = a2;
```

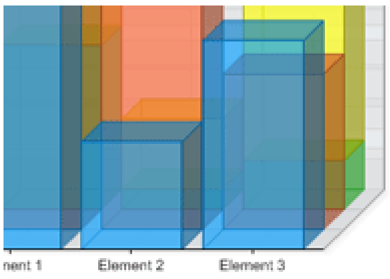
```
mySC[1].YAxis = a2;
a2.Scale = Scale.Stacked;


[Visual Basic]
Dim a2 As New Axis()
mySC(0).YAxis = a2
mySC(1).YAxis = a2
a2.Scale = Scale.Stacked
```

Will yield this result:



Notice that the new y axis is stacked but still the main x axis manages the clustered - unclustered layout. Therefore if we wanted to uncluster the columns we could have multiple z axis steps and some may be stacked while other wont.

```
[C#]
Chart.XAxis.ClusterColumns = false;

[Visual Basic]
Chart.XAxis.ClusterColumns = false
```



Please note that using multiple value axes while hiding one of them may result in one going out of sync with the other. This will result in the bars indicating incorrect values (according to the visible value axis). This issue can be resolved by synchronizing the visible with the invisible axes.

For more information see the **scale synchronization (Section 1.1)** tutorial.

Sample: AxisDualScales.aspx

6 Axis Value Coordinates

Introduction

This new version 4.0 feature enables users to click on a chart and obtain the value of any axis, at the specific click location. This allows users to interact with the chart by selecting a point or range on a given axis.

Usage

When the chart is clicked, the coordinates of the mouse click on the image must be returned. This is achieved by using a custom image tag with the ISMAP attribute or an image type input tag .

⚠ In order to use this feature the default hotspot functionality cannot be used.

For example a working html tag that will return the coordinates will look like this:

<IMG src="dnc-1j4hgd.png" ISMAP>

To get the x y coordinates that can be passed to the axis:

```
[C#]
string coordinates = Page.Request.QueryString[0];


[Visual Basic]
Dim coordinates As String = Page.Request.QueryString(0)
```

OR

<FORM method=get> <INPUT Type=Image SRC="dnc-1j8d9ahj.png" Value=Submit > </FORM>

To get the x y coordinates that can be passed to the axis:

```
[C#]
string coordinates = Page.Request.Params["x"] + "," + Page.Request.Params["y"];


[Visual Basic]
Dim coordinates As String = Page.Request.Params("x") & "," & Page.Request.Params("y")
```

When the coordinates are retrieved they can be passed to the axis.GetValueAtX() or axis.GetValueAtY() method and the value on the axis at this coordinate is returned as an object. If null is returned, the coordinates were not valid for the particular axis.

⚠ In WebForms, because the chart object does not maintain a state through postbacks, the chart must be generated before this method can be called.

For more information on arbitrarily generating the chart, see this **kb article (http://dotnetcharting.com/kb/article.aspx?id=10073)**.

Samples:

- XAxisZoom.aspx
- YAxisZoom.aspx
- AxisValueClick.aspx