J S Charting

J S Charting

Intro

JSCharting is a javascript based charting layer. It works similar to zoomer and navigator but uses only javascript to render the chart client side. This allows real-time updates of chart properties and data. It uses SVG and VML graphics technologies to increase browser compatibility and chart interactivity which yields a richer experience on devices such as phones and tablets running iOS, Android and more.

Benefits:

- Browser Compatibility
- Renders Client-Side
- Real-Time updates
- Interactivity
- Animation
- Javascript based events.
- Touch-Screen support.

It is easy to use, by simply setting Chart.JS.Enabled = true a pure javascript chart will render in place of static chart images.

JSCharting is a subset of the full DNC library but many of the more important features are supported. The feature differences are discussed in more detail below.

What's new in v8.0

Version 8.0 is a major update to the previous JSCharting implementation. Some new features include:

- Label Tokens and formatting are practically in feature parity with image charts
- JS Navigator for finance charts and zooming.
- Client side API for js charts closely matches DNC API that makes reading the generated JS code easier.
- The new version of JSC is much easier to work with.

⚠ This new version introduces a new client side API, if existing charts are using JS to manipulate the chart, this code may have to be modified.

Supported features

- Export to raster and vector images
- Printing support
- Element Selection
- Series Visibility
- Zooming
- Tooltips
- JS Navigator
  - Multiple chart areas
- Chart Types
  - Combo
  - ComboHorizontal
  - Pie
  - Pies
  - PiesNested
  - Donut
  - Donuts
- Series Types
  - Bar/Column
  - Line
  - Spline

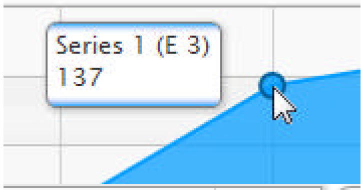
- AreaLine
- AreaSpline
- Marker
- Bubble
- Candlestick (Using JS Navigator)
- OHLC Bars (Using JS Navigator)

- Real-Time updating
- Zooming

User Interface (UI)

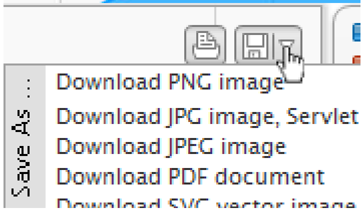
The JSCharting user interface is inherently more interactive and dynamic. It offers a number of features described below.

Tooltips
The tooltips have a highlight of the element color so it's easy to see which data point it relates to. Tooltips are also optimized to work with touch devices.

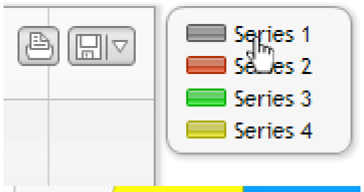


Element Highlighting on Hover
When the mouse hovers on elements, they are highlighted.

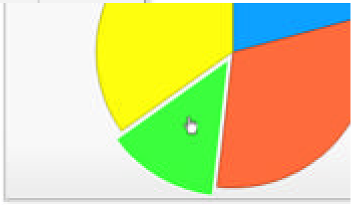
Export / Print buttons
Each JSCharting chart can have two buttons. One sends the chart to the printer and the other allows saving the chart as a raster or vector image. Several popular image formats are supported including PDF, PNG, and Jpeg.



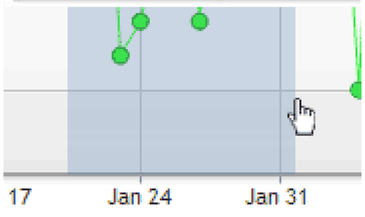
Legend Entries
The legend entries are clickable to hide series in any JSCharting chart.



Element selection
When element selection is enabled, clicking on elements selects them and the selected elements can be detected through javascript. With a pie chart, clicking on elements selects them and explodes the selected pie slice.



Zooming
Dynamic zooming is also available when enabled. It allows the user to zoom in and out of the charts range.



Server Side Programming Interface (API)

JSCharting is driven by the current .netCHARTING API. Simply enabling JSCharting will render the chart in the client-side browser instead of generating an image on the server. JSCharting offers the ability to interact with the chart in the browser through Javascript.

Enabling JSCharting
To enable JSCharting, the following code can be used:

```
[C#]
Chart.JS.Enabled = true;

[Visual Basic]
Chart.JS.Enabled = true
```

Initial Animation

When the JSCharting loads, it animates the series drawing. The duration of this animation can be specified in milliseconds and set as in the following code snippet:

```
[C#]
Chart.JS.InitialAnimationDuration = 1000;


[Visual Basic]
Chart.JS.InitialAnimationDuration = 1000
```

Element Selection

Element selection can be enabled or disabled with the following code:

```
[C#]
Chart.JS.EnableElementSelection = true;


[Visual Basic]
Chart.JS.EnableElementSelection = true
```

Language Strings

There are a number of strings used in JSCharting written in english by default. These strings can be modified for use in other languages or personalized. The **Chart.JS.LanguageStrings** class offers a number of properties to modify all the different strings necessary. The following code snippet modifies the print button tooltip text:

```
[C#]
Chart.JS.LanguageStrings.PrintButtonTooltip = "Print";


[Visual Basic]
Chart.JS.LanguageStrings.PrintButtonTooltip = "Print"
```

See Sample: **JSCLanguageStrings (http://www.dotnetcharting.com/gallery/view.aspx?id=JSCLanguageStrings)**

Buttons

The print and export buttons are very customizable. They can be positioned in different locations in the chart, use different colors, and diabled. The options for these buttons are accessed through the **Chart.JS.Buttons** class. The following code snippet specifies a different color and position for the buttons:

```
[C#]
Chart.JS.Buttons.OutlineHover.Color = Color.FromArgb(158,0,0);
Chart.JS.Buttons.Orientation = dotnetCHARTING.Orientation.TopRight;
Chart.JS.Buttons.Offset = new Point(0, -35);


[Visual Basic]
Chart.JS.Buttons.OutlineHover.Color = Color.FromArgb(158,0,0)
Chart.JS.Buttons.Orientation = dotnetCHARTING.Orientation.TopRight
Chart.JS.Buttons.Offset = New Point(0, -35)
```

See sample: **JSCExportButtons (http://www.dotnetcharting.com/gallery/view.aspx?id=JSCExportButtons)**

Zooming

To enable zooming for a javascript chart, the Chart.JS.AxisToZoom property can be utilized. It takes a string with possible values of: "X", "Y", "XY".

```
[C#]
Chart.JS.AxisToZoom = "X";


[Visual Basic]
Chart.JS.AxisToZoom = "X"
```

See Sample: **JSCZooming (http://www.dotnetcharting.com/gallery/view.aspx?id=JSCZooming)**

Client Side Programming Interface (API)

JSCharting can interact with Javascript in the browser to update dynamically and notify other javascript code of chart based events.

In order to interact JSCharting, a control ID must be specified for the chart on the server side so it can be referred to in javascript. The following code can be used to achieve this:

```
[C#]
Chart.JS.ControlID = "jscChart";


[Visual Basic]
Chart.JS.ControlID = "jscChart"
```

The following Javascript snippets assumes the above control id setting is used.

Reading the JS output

When Chart.Debug=true, the generated JS will be legible. When debug is disabled, whitespace is removed.

Unsupported JS settings

There are a few JS API settings that cannot be set through the DNC API such as point states like hover and selected. For this and other such cases, a method is provided that applies settings given a path a value.

```
[C#]
Chart.JS.Settings.Add("defaultSeries.defaultPoint.states.hover.outline.color", "red");


[Visual Basic]
Chart.JS.Settings.Add("defaultSeries.defaultPoint.states.hover.outline.color", "red")
```

This method takes only strings values, however, any type can be specified. If a value of "true" is passed, it will be a boolean in JS. To use a string with that value, use "'true'" with single quotes. The same applies to numbers. If a JS variable that holds the value in JS must be referenced, prefix the string with "js:". This will prevent quote wrapping and will be used in JS as is. For example, if in JS there is a variable defined as var colorRed = 'red'; to refer to this variable in a string setting server side, use "js:colorRed". This allows passing functions as well for cases like label callbacks. For example: "js:function(txt){return 'text:'+txt}"

See Sample: **JSCSettings (http://www.dotnetcharting.com/gallery/view.aspx?id=JSCSettings) JSCLabelCallbacks (http://www.dotnetcharting.com/gallery/view.aspx?id=JSCLabelCallbacks)**

Axis and AxisMarkers

The following axis related javascript usage is demonstrated in sample: **JscDynamicStylingUpdates (http://www.dotnetcharting.com/gallery/view.aspx?id=JscDynamicStylingUpdates)**

Getting Axis and Data Scale Ranges

First, the Chart.JS.ControlID must be specified, and an ID must be assigned to the axis server side.

```
[C#]
Chart.JS.ControlID = "JscChart";
Chart.YAxis.JsID = "axy";
Chart.XAxis.JsID = "axx";


[Visual Basic]
Chart.JS.ControlID = "JscChart"
Chart.YAxis.JsID = "axy"
Chart.XAxis.JsID = "axx";
```

Then in JS, the axis range can be acquired using code such as:

```
var range= JscChart.get('axy').getRange();
```

This returns a value such as:

```
{ min:0, max:100, dataMin:5, dataMax:80}
```

The result contains the following values:

extremes.min - Axis scale minimum.
extremes.max - Axis scale maximum.
extremes.dataMin - Data scale range minimum.
extremes.dataMax - Data scale range maximum.

Setting Axis Scale Range

Similar code can be used to apply a new axis scale range.

```
JscChart.get('axy').setRange(0,2);
```

To set a date time based scale, use:

```
JscChart.get('axy').setRange(new Date(2011, 0, 2), new Date(2011, 0, 5));
```

Adding and Removing Axis Markers dynamically.

Axis markers can be added and removed in realtime through Javascript.

The following code snippet adds a range axis marker to the x axis.

```
JscChart.get('axx').addMarker({
value:[new Date(2011, 0, 2),new Date(2011, 0, 5)],
color: '#ACFFC5',
id: 'am1'
});
```

In order to add and remove a single value axis marker which draws as a line instead of an area, the following code can be used.

```
JscChart.get('axx').addMarker({
value: new Date(2011, 0, 2),
color: '#ACFFC5', lineWidth:2,
id: 'am1'
});;
```

To remove this axis marker, the following code can be used:

```
JscChart.get('axx').removeMarker('am1');
```

Real-Time Data Update

Adding a point to a series

To add a point to a series in real time the following javascript function can be used.

```
series.addPoint( pointOptions, [ shift ], [ animation ] )
```

options - Takes an array, or point object. If it is an array, it will be interpreted as x and y values respectively.

The point object contains the following basic properties:

- color
- name
- x
- y
- id

shift - When shift is true, one point is shifted off the start of the series as one is appended to the end. Use this option for live charts monitoring a value over time.

animation - true/ false, or the animation duration in milliseconds.

Note: The series must be assigned an ID server side in order to refer to it later.

```
var series = JscChart.get('s1');
series.addPoint([2, 6]);
```

To use a point object the follwing code can be used:

```
var series = JscChart.get('s1');
```

```
series.addPoint({
  color: 'Red'
  x: 2,
  y: 6
});
```

The point object has the following available functions:

- remove()
- Select(bool select, bool accumulate)
  select - When true, the point is selected. When false, the point is unselected. When null or undefined, the selection state is toggled.
  accumulate - When true, the selection is added to other selected points. When false, other selected points are deselected. Internally in Highcharts, selected points are accumulated on Control, Shift or Cmd clicking the point.
- update(pointOptions)

  pointOptions - The setting this point will be updated with.

Adding a series to a chart
To add a new series to a chart the following type of code snippet can be used:

```
JscChart.addSeries({
 points: [194.1, 95.6, 54.4, 29.9, 71.5]
});
```

See sample **JscLiveDataUpdate (http://www.dotnetcharting.com/gallery/view.aspx?id=JscLiveDataUpdate)** for an example of how points are added to a chart in javascript.

Get a Selected Point
To get the points currectly selected the follwing code can be used:

```
var selectedPoints = JscChart.getSelectedPoints();
```

This returns an array of selected point objects.

The point object has the following available functions:

- remove()
- Select(bool select, bool accumulate)
  select - When true, the point is selected. When false, the point is unselected. When null or undefined, the selection state is toggled.
  accumulate - When true, the selection is added to other selected points. When false, other selected points are deselected. Internally in Highcharts, selected points are accumulated on Control, Shift or Cmd clicking the point.
- update(pointOptions)

  pointOptions - The setting this point will be updated with.

The JS point object has the following properties:

- chart- A reference to the parent JS chart object.
- series - A reference to the parent JS series object.
- name - The name of this point.
- id- The point's ID.
- x - The x value.
- y - The y value.

Events

Events can be specified server side through the hotspot class and include javascript handlers.

The following code snippet sets a click event handler for the chart and includes a function that adds a point at the clicked location to the chart.

```
[C#]
Chart.DefaultElement.Hotspot.Attributes.Custom.Add("click", "js: function(){if (this.serie
```

```
[Visual Basic]
Chart.DefaultElement.Hotspot.Attributes.Custom.Add("click", "js: function(){if (this.serie
```

When adding the click event, a function handler with one parameter can be used. The parameter contains the click event information. The above shows how to get the clicked axis values based on the event. The 'this' keyword refers to the javascript chart object.

When using a setting for JSC, prefixing the string value with "js:" indicates the string should be treated as a JS literal. Some label text settings can include js callback functions to process the text when needed.

See sample **JscClickAddPoint (http://www.dotnetcharting.com/gallery/view.aspx?id=JscClickAddPoint)** for an example.

Chart level events

- **click** (fires when the chart is clicked)
  Keyword this refers to the chart object.
  One parameter 'event' is pased to a JS function handler. event.xAxis and event.yAxis contain arrays of axes on the chart.
- **addSeries** (Fires when a series is added to the chart)
  Keyword this refers to the chart object.
  One parameter 'event' is pased to a JS function handler. event.options gives access to the series being added to the chart.
- **selection** (fires when an area of the chart has been selected 'zooming')
  Keyword this refers to the chart object.
  One parameter 'event' is pased to a JS function handler.
  event.preventDefault() will stop the zooming from occuring.
  event.xAxis and event.yAxis contain arrays of axes on the chart.

Events can be specified for an element as well. The following code snippet adds a handler to elements that removes them when clicked:

```
[C#]
Chart.DefaultElement.Hotspot.Attributes.Custom.Add("click", "if (this.series.data.length >
```

```
[Visual Basic]
Chart.DefaultElement.Hotspot.Attributes.Custom.Add("click", "if (this.series.data.length >
```

See sample **JscClickAddPoint (http://www.dotnetcharting.com/gallery/view.aspx?id=JscClickAddPoint)** for an example.

Element level Events

- **click** (fires when an element is clicked)
  Keyword this refers to the element object.
  One parameter 'event' is pased to a JS function handler.
- **mouseOver** (fires when the mouse hovers an element)
  Keyword this refers to the element object.
  One parameter 'event' is pased to a JS function handler.
- **mouseOut** (fires when the mouse leaves an element)
  Keyword this refers to the element object.
  One parameter 'event' is pased to a JS function handler.
- **remove** (fires when the element is removed)
  Keyword this refers to the element object.
  One parameter 'event' is pased to a JS function handler.
- **select** (fires when the element is selected)
  Keyword this refers to the element object.
  One parameter 'event' is pased to a JS function handler.
- **unselect** (fires when the element is unselected)
  Keyword this refers to the element object.
  One parameter 'event' is pased to a JS function handler.
- **update** (fires when the element is updated programatically)
  Keyword this refers to the element object.
  One parameter 'event' is pased to a JS function handler.