TeeChart for Javascript

User Guide

Rev. 1.0, March 2012

Introduction

This document explains in detail how to create charts and graphs using the TeeChart for Javascript library.

Adding charts to your web pages or applications is fun and easy, see the one-minute Starting Guide, play with the Online Demos or discover the full TeeChart API Reference.

Compared with other programming environments (Java, .NET, Delphi VCL / FireMonkey, etc), the TeeChart Javascript api implements an essential subset of features in order to minimize both the script size and coding complexity.

The Tee namespace

Full code is wrapped around a single “Tee” identifier (namespace) to avoid cluttering the Window global space. Several “classes” in an object-oriented manner exist Inside the Tee namespace, being Tee.Chart the main class that defines a chart.

The Canvas

Charts are displayed by default using HTML5 Canvas elements. The main class “Tee.Chart” should be initialized passing the <canvas> element or its id as string:

<canvas id="canvas1" width="300" height="200"></canvas>

Using the canvas id:

var Chart1=new Tee.Chart("canvas1");
...

Or using the element:
var Chart1=new Tee.Chart(document.getElementById("canvas1"));

Hidden charts can be created without passing any canvas parameter:

var Chart1=new Tee.Chart();

**Bounds and position**

The default position of charts inside canvas is 0,0 (left-top corner), and the default size is the same as the associated canvas. You can override them by changing the chart bounds property:

Chart1.bounds.x=50; Chart1.bounds.y=20; Chart1.width=200; Chart1.height=150;

**Format**

Most objects (like Series, Annotations and chart sub-objects) include a “format” property to group all visual attributes in a common place.

The Format class provides the following properties:

**fill**: A color in string format, for example: #123456, rgb(255,0,0), rgba(0,255,0,0.5), “blue”

Chart1.panel.format.fill = “blue”;

**round**: Amount in pixels of roundness of rectangle corners (x: horizontal, y: vertical)

Chart1.panel.format.round = { x:10, y:10 }

**transparency**: Amount of semi-glass effect, from 0 to 1. (default is zero)

Chart1.legend.format.transparency = 0.5;

**shadow**: Properties to control drop-shadow visibility behind filled shapes or stroked lines.

```javascript
var s = Chart1.panel.format.shadow;
s.visible = true;
s.width = 4;       // can be negative for left-side shadow
s.height = 4;     // can be negative for top-side shadow
s.blur = 5;
s.color = “silver”;
```

**gradient**: Attributes to fill shapes using a gradient of colors:
var g = Chart1.panel.format.gradient;
g.visible = true;
g.colors = ["red", "yellow", "green"]; // array of colors
g.stops = [0, 0.1, 1]; // percentages from 0 to 1 for each color (default is null, automatic)

// "bottomtop", "leftright", "rightleft", "radial", "diagonalup", "diagonaldown"
g.direction = "topbottom";

**stroke:** Attributes used to draw lines and outlines around filled shapes:

var s = Chart1.panel.format.stroke;
s.fill = "black";
s.size = 4; // line thickness
s.join = "round"; // segment joins ("round", "miter", "bevel")
s.cap = "square"; // ending line terminators ("square", "butt", "round")
s.dash = [5, 10]; // dot-dash... array of pixels (default is null to draw solid lines)
s.gradient.visible = true; // gradient object with colors, direction, etc

**font:** Properties to display text:

var f = Chart1.title.format.font;
f.style = "22px Verdana"; // or: "bold italic 14px Tahoma", etc, etc
defill = "blue";
f.shadow.visible = true; // text drop-shadow
f.stroke.fill = "white"; // draws an outline around text
f.gradient.visible = true; // fills text with gradient instead of using font.fill
f.textAlign = "center"; // horizontal alignment: "start", "end", "left", "center", "right"

// vertical alignment: "top", "middle", "bottom", "alphabetic", "hanging", "ideographic"
f.baseLine = "top";

**Background**

The chart background appearance is controlled by the "panel" sub-object:

Chart1.panel.transparent = false;
Chart1.panel.format.gradient.visible = true;

**Title and Footer**

There two chart properties are used to display text at top and bottom sides:

Chart1.title.text = "Hello";
Chart1.footer.text = "World";

Multiple-line text is done by adding "\n" line-feed delimiters:
Chart1.title.text = “Hello \n World”;

They are Annotation objects, and such they inherit the “format” sub-properties:

Chart1.title.visible = true;
Chart1.title.transparent = false;
Chart1.title.format.gradient.visible = true;
Chart1.title.format.round.x=20;

**Adding data to series**

Data is added to a chart using “Series” objects. Multiple series can exist in the same chart. Each series can be of a different type (line, area, bar, pie, etc) so you can mix several styles altogether.

Series are added into a chart using the addSeries method:

Chart1.addSeries(new Tee.Bar());

By default series are empty, they contain no data. For testing purposes its handy to add random values, for example:

Chart1.addSeries(new Tee.Line()).addRandom(1000); // Thousand random points

Data can be specified when creating the series:

Chart1.addSeries(new Tee.Pie( [ 10, 20, 30, 40 ] ));

All data is stored under the series “data” property. You can access and modify data directly:

var a = new Tee.Area();
Chart1.addSeries( a );
a.data.values = [ 10, 20, 30, 40 ];

Each series point has an associated text label. By default labels are empty, you can modify them using the data.labels property:

a.data.labels = [ “a”, “b”, “c”, “d” ];

Some series allow specifying point positions or other point parameters. For example, Line and PointXY series can optionally display each line segment or point at a specific “X” axis coordinate:

var p = new Tee.PointXY();
p.data.values = [5, 7, 9];
p.data.x = [0.23, 14, 16];
Chart1.addSeries(p);

Bubble series have a “data.radius” array, and Candle series have data.open, data.close,
data.high and data.low arrays.

**Adding data from other sources**

Several helper functions are provided in a separate script (teechart-table.js) to import data from different sources (see the online demos for details), for example:

From a textarea html element:

```javascript
Chart1.addSeries(new Tee.Bar(document.getElementById("data")));
<textarea id="data" rows="20" cols="20" wrap="off">
 7,Apples
 4
 3,Oranges
 9
 1,Banana
 6,Kiwis
 2</textarea>
```

From a table html element:

```javascript
Chart1.fromTable('table1', Tee.Bar, true, 0,0);
<table id="table1">...</table>
```

From a text file from URL:

```javascript
// Warning: data file url should be at "same origin"
```

From another series in same or different chart:

```javascript
// Simply assign the "data" property from one series to another:
Chart1.series.items[0].data = Chart2.series.items[3].data;
```

From xml formatted text:

```javascript
var b=Chart1.addSeries(new Tee.Bar());
b.loadXML(document.getElementById("xml"));
...
<textarea id="xml" rows="10" cols="60" wrap="off">
<series name="Friends" color="Blue" metric="Quantity">
  <point name="Facebook" value="123"/>
  <point name="Twitter" value="456"/>
  <point name="Google+" value="789"/>
</series>
</textarea>
```
From JSON formatted text:

```javascript
var b=Chart1.addSeries(new Tee.Bar());
b.loadJSON(document.getElementById("json"));
```

**The Legend**

The indicator where series names and series points are displayed is called the legend. Several properties can be used to customize the legend behaviour and appearance.

```javascript
Chart1.legend.visible = true;
Chart1.legend.transparent = false;
Chart1.legend.format.fill = "yellow";
```

Legend position and orientation:

```javascript
Chart1.legend.position = "left"; // "top", "bottom", "right"
Chart1.legend.inverted = false; // draws items in normal order
```

Select what to display at legend:

```javascript
Chart1.legend.legendStyle = "values"; // "auto", "series", "values"
Chart1.legend.labelXStyle = "percentlabel"; // show point value as percent, and point text label
```

Font and formatting:

```javascript
Chart1.legend.format.font.style = "20px Arial";
```

Margins to chart top and chart axes:

```javascript
Chart1.legend.vertical = 20; // percentage from 0 to 100
Chart1.legend.padding = 5; // in pixels
```

Title text on top of items:

```javascript
Chart1.legend.title.text = "My Legend";
Chart1.legend.title.visible = true;
```

Symbols next to legend items:

```javascript
Chart1.legend.symbol.visible = true;
Chart1.legend.symbol.width = 20;
Chart1.legend.symbol.format.shadow.width = 5;
```
Other legend properties:

Chart1.legend.fontColor = true;  // each item text is painted using its point or series color
Chart1.dividing.fill = "blue";  // draws lines between legend items

Series and Colors

Charts have a palette property (an array [ ] of colors in string format). This palette is used by series to paint its points, and by chart legend to display series items symbols.

Chart1.palette = [ “red”, “blue”, “green”, “yellow” ];

Colors in the palette are reused in a circular way when more series or more points in a series exist than the size of the palette.

Series also contain a palette of colors to override the chart palette. By default is a “null” empty array so they share the chart palette.