Information Builders

WebFOCUS

JavaScript Charting Engine API Guide

DN4501197.0412

u Table of Contents u

Section 1: About Js:Chart

The JavaScript Chart Engine (Js:Chart) draws a chart in an HTML environment using a defined data set and Java Script Object Notation (JSON) definitions. The JSON object(s) define chart properties that control the appearance of the chart and chart objects.

Creating a chart is trivial.

Define the tdgchart.js file in the script tag in your HTML file.

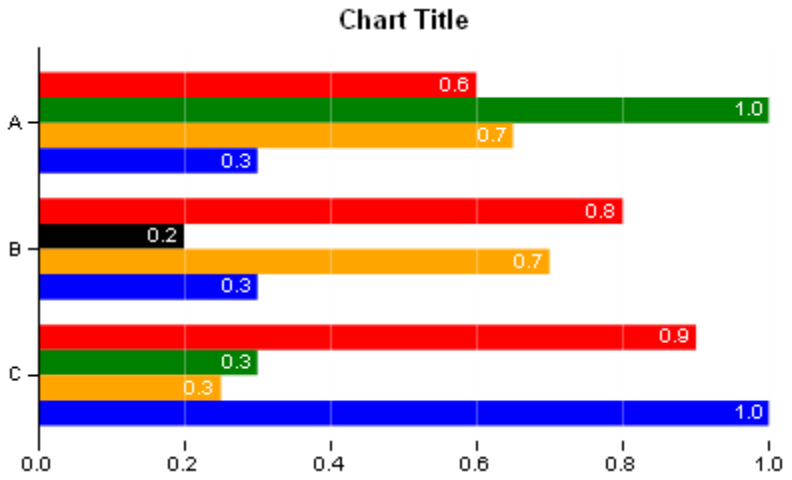
```
<script type="text/javascript" src="tdgchart.js"></script>;
```

Add a ** ** in your HTML code to define where you want to draw the chart. Example:

```
<div class='chart' id='myFirstChart'></div>
```

In any <script> tag, include:

```
var chart = new tdgchart(props);
chart.draw('myFirstChart')
```



JSON objects define values for chart properties. In addition to complete JSON objects, Js:Chart can accept JSON objects that only include one or a few properties. These property values are simply used in place of any existing settings.

Properties can also be set via 'dot' notation. Example:

```
chart.title.text = 'Chart Title';
```

Requirements

A modern browser (Firefox, Safari, Chrome, IE 9) with SVG support and JavaScript enabled

Getting Started

These steps define the information your need to include in your HTML file to draw a chart.

1) Define the tdgchart.js file in the script tag.

```
<script type="text/javascript" src="tdgchart.js"></script>;
```

2) Define the target location in your web page to draw the chart. Example:

```
<div class='chart' id='chart1'>Optional text for unsupported
browsers.</div>;
```

3) Optional: Define the data that will form the chart. Example:

```
var data = [
[0.6, 1.0, 0.65, 0.3],
[0.8, 0.2, 0.7, 0.3],
[0.9, 0.3, 0.25, 1.0]
];
```

Note that different chart types may require more than one value to draw each riser or marker (see the chartType property for details). The default properties file (e.g., properties.js) includes this example data set that will draw a bar, line, or area chart. Your Js:Chart package may include multiple default properties files for different chart types.

4) Optional: Define chart properties. Example:

```
var props = {
   chartType: 'line',
   border: {width: 2, color: 'grey'},
   title: {
         text: 'Js:Chart',
         font: 'bold 12pt Sans-Serif', color:'rgb(0,101,163)'},
   chartFrame: {
         border: {width: 1, color: 'cyan'},
         fill: {color: 'hsla(140, 100%, 50%, 0.1)'}},
   blaProperties: {orientation: 'vertical',lineConnection: 'curved'},
   dataLabels: {visible: false},
   series: [
   {series: 0, color:'rgb(0,142,126)', marker: {visible: true}},
   {series: 1, color:'rgb(152,181,211)', marker: {visible: true}},
   {series: 2, color:'rgb(0,101,163)', marker: {visible: true}},
   {series: 3, color:'rgb(173,204,189)', marker: {visible: true}},
   ],
};
```

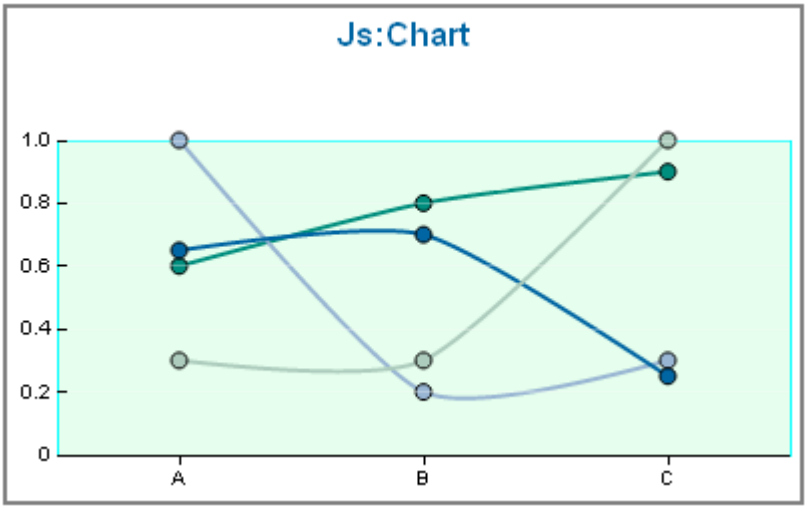
If properties are not specified, property settings from a default properties file (e.g., properties.js) will be used. Your Js:Chart package may include multiple default properties files for different chart types.

5) Create the chart.

```
var chart = new tdgchart(props);
```

6) Draw the chart.

```
chart.draw('chart1');
```



Properties Overview

Default Property Values

The following table shows the values from the default properties file (e.g., properties.js) that will be used when properties are not defined. Your Js:Chart package may include multiple default properties files for different chart types.

Group	Property	Default Value
chart	axisAutoLayout	{rotate45: true,rotate90: true,truncate: true, stagger: true,skip: true}
	border	{width: 0, color: 'transparent', dash: ''}
	catchErrors	true
	chartFrame	fill: {color: 'transparent'},border: {width: 0, color: 'transparent',dash: ''},shadow: false
	chartsPerRow	undefined
	chartType	'bar'
	colorMode	'bySeries'
	colorModeColors	undefined
	data	[[0.6, 0.8, 0.9], [1.0, 0.2, 0.3], [0.65, 0.7, 0.25], [0.3, 0.3, 1.0]]
	dataLabels	visible: true, displayMode: 'x', position: 'top', font: '7.5pt Sans-Serif', color: 'black', numberFormat: '[>9]#,#;[<0]-#.#;[<=9]#.#;[=0]0', formatCallback: undefined
	dataSubset	startGroup: undefined, stopGroup: undefined
	depth	undefined
	fill	color: 'white'
	footnote	text: 'Chart Footnote', visible: false, align: 'center', font: '10pt Sans-Serif', color: 'black', tooltip: undefined
	groupLabels	"ABCDEFGHIJKLMNOPQRSTUVWXYZ".split('')
	height	250
	interaction	{click: undefined, mousedrag: undefined, dblclick: undefined}
	introAnimation	{ enabled: false, duration: 1000 }
	labelPadding	5
	mouseOverIndicator	enabled: false, color: undefined, marker: { color: 'lightblue', size: 5, shape: 'circle', rotation: 0, position: 'top', border: { width: 1, color: 'darkblue', dash: '' } }
	riserBevel	undefined

Group	Property	Default Value
	riserCycleEndLightness	0.8
	riserDepthGap	0.2
	riserShadow	false
	subtitle	text: 'Chart Subtitle', visible: false, align: 'center', font: '10pt Sans-Serif', color: 'black', tooltip: undefined
	swapData	false
	swapDataAndLabels	false
	title	text: 'Chart Title', visible: true, align: 'center', font: 'bold 10pt Sans-Serif', color: 'black', tooltip: undefined
	width	400
legend	backgroundcolor	'transparent'
	labels	font: '7.5pt Sans-Serif', color: 'black'
	lineStyle	{width: 0, color: 'black', dash: ''}
	markerPosition	'left'
	markerSize	8
	maxEntries	undefined
	position	'right'
	shadow	false
	title	visible: false, text: 'Legend Title', font: '10pt Sans-Serif', color: 'black'
	visible	false
	xy	{x:330, y:80}
xaxisOrdinal	bodyLineStyle	width: 1, color: 'transparent', dash: ''
	invert	false
	labels	visible: true, font: '7.5pt Sans-Serif', color: 'black', rotation: undefined
	majorGrid	visible: false, aboveRisers: true, lineStyle: {width: 1, color: 'rgba(255, 255, 255, 0.3)', dash: ''}
	majorGrid:ticks	length: 5, visible: true, style: 'outer', lineStyle: {width: 1, color: 'black'}
	swapChartSide	false
	title	text: 'X Axis Title', visible: false, font: '7.5pt Sans-Serif', color: 'black'
	timeAxis	enabled: false, startTime: undefined, stopTime: undefined, interval: undefined, stepSize: undefined, labelFormat: undefined
xaxisNumeric| yaxis | y2axis	altFrameColor	undefined
	baseLineStyle	width: 1, color: 'black', dash: ''
	bodyLineStyle	width: 1, color: 'transparent', dash: ''
	intervalMode	undefined
	intervalValue	undefined
	invert	false
	labels	visible: true, font: '7.5pt Sans-Serif', color: 'black'
	majorGrid	visible: true, aboveRisers: true, lineStyle: {width: 1, color: 'rgba(255, 255, 255, 0.3)', dash: ''}
	majorGrid:ticks	length: 5, visible: true, style: 'inner', lineStyle: {width: 1, color: 'black'}
	min/max	undefined
	minorGrid	visible: false, count: undefined, lineStyle: { width: 1, color: 'black', dash: ''}

Group	Property	Default Value
	minorGrid: ticks	length: 5, visible: false, style: 'inner', lineStyle: {width: 1, color: 'black'}
	numberFormat	'[>9]#,#;[<0]-#.#;[<=9]#.#;[=0]0'
	swapChartSide	xAxisNumeric=false, yaxis = false, y2axis = N/A
	title	text: 'X | Y | Y2 Axis Title', visible: false, font: '7.5pt Sans-Serif', color: 'black'
zaxisOrdinal	bodyLineStyle	width: 1, color: 'transparent'
	invert	false
	labels	visible: true, font: '7.5pt Sans-Serif', color: 'black'
	majorGrid	visible: true, aboveRisers: true, lineStyle: {width: 1, color: 'rgba(255, 255, 255, 0.3)', dash:''}
	majorGrid: ticks	length: 5, visible: true, style: 'outer', lineStyle: {width: 1, color: 'black'}
	swapChartSide	false
	title	text: 'Z Axis Title', visible: false, font: '10pt Sans-Serif', color: 'black'
series		series: 'all', color: 'blue', showDataValues: true, border: {width: 2}, marker: {size: 8, border: {width: 1, color: 'black'}}, series: 0, color: 'red', series: 1, color: 'green', series: 2, color: 'orange'
blaProperties	barGroupGapWidth	0.2
	comboCharts	{barSeriesLayout: undefined, lineSeriesLayout: undefined, areaSeriesLayout: undefined}
	lineConnection	'linear'
	orientation	'horizontal'
	seriesLayout	'sideBySide'
boxPlotProperties	connectorLine	{width: 1,color: 'black',dash: ''}
	drawHatAsBox	false
	hatWidth	'100%'
	medianLine	{width: 1,color: 'black',dash: ''}
bulletProperties	drawFirstValueAsBar	true
funnelProperties	baseWidth	'20%'
	groupLabel	{visible: false,font: '10pt Sans-Serif',color: 'black'}
	riserGap	0
	topWidth	'90%'
ganttProperties	durationValues	false
	interval	undefined
	labelFormat	undefined
	staggerRisers	true
	startTime	undefined
	stopTime	undefined
gaugeProperties	axisTickLength	"30%"
	axisWidth	"22%"

Group	Property	Default Value
	endAngle	45
	fill	color: 'transparent'
	groupLabel	visible: false, font: '10pt Sans-Serif', color: 'black'
	needleBase	size: "6%", color: '#1f77b4', border: {width: 0,color: 'transparent',dash: ''}
	outerBorder	width: '10%', fill: {color: '#1f77b4'}, border: {width: 0,color: 'transparent',dash: ''}
	secondaryNeedlesAsMarkers	false
	startAngle	135
heatmapProperties	dataColors	['#FFFFCC', '#A1DAB4', '#41B6C4','#2C7FB8','#253494']
histogramProperties	binCount	undefined
	binSize	undefined
	startBinValue	undefined
pieProperties	feelerLine	visible: true, width: 1,color: 'black',dash: ''
	holeSize	0
	label	visible: false,font: '10pt Sans-Serif',color: 'black'
	otherSlice	threshold: undefined, legendLabel: 'Other', color: 'grey', border: {width: 1,color: 'transparent',dash: ''},showDataValues: true,marker: {shape: 'circle', border: {width: 0,color: 'transparent',dash: ''}}
	rotation	0
	skew	0
	totalLabel	visible: false, font: '10pt Sans-Serif', color: 'black'
polarProperties	drawAsArea	false
	extrudeAxisLabels	false
	straightGridLines	false
stockProperties	downRiserColor	'#e2675b'
	hiLowLine	width: 1,color: undefined,dash: ''
	interval	undefined
	labelFormat	undefined
	startTime	undefined
	stopTime	undefined
	upRiserColor	'#77b39a'
threedProperties	rotate	40
	shadeSides	true
	tilt	40
waterfallProperties	appendTotalRiser	true
	connectorLine	width: 1,color: 'black',dash: ''
	negativeRiserColor	'#e2675b'
	otherRiserColor	'#aaaaaa'
	otherRisers	[]
	positiveRiserColor	'#77b39a'
	subtotalRisers	[]

Group	Property	Default Value
	zeroRiserColor	'#7593bd'

Colors & Gradients

Color properties define the color of an object. All objects can be assigned a color and transparency setting. Area and line objects can be assigned a color definition or a gradient definition.

Color Definitions

color: 'string': For color and transparency settings, the string can be one of the following:

- a color name (e.g., 'coral'),
- three RGB values (i.e., 'rgb (r,g,b)'),
- three RGB values and a transparency setting (i.e., 'rgba(r,g,b,a)'),
- three Hue-Saturation-Lightness values (i.e., 'hsl(h,s,l)'),
- three HSL values and a transparency setting (i.e., 'hsla(h,s,l,a)'), or
- hex values (e.g., '#f00' or '#ff00AA').

The World Wide Web Consortium (W3C) web site at http://www.w3.org/TR/css3-color/#colorunits provides details about color specifications.

Gradient Definitions

Gradients can be defined by a string or a JSON object.

JSON Object: This JSON object definition defines a linear or radial gradient:

```
color:  {
   type: 'string',
   start:{
         x: number | 'string',
         y: number | 'string'
   },
   end:{ // Linear Gradients Only
         x: number | 'string',
         y: number | 'string'
   },
   radius: 'string',  // Radial Gradients Only
   stops: [
               {
                     offset: number | 'string',
                     color: 'string'
               },
               ...
         ]
}
```

type: a string that defines the type of gradient: 'linear' or 'radial'

start/x: specifies the starting X-coordinate in the destination object space.

start/y: specifies the starting Y-coordinate in the destination object space.

end/x: If type is 'linear', specifies the ending X-coordinate in the destination object space.

end/y: If type is 'linear', specifies the ending Y-coordinate in the destination object space.

radius: If type is 'radial, specifies the radius of the gradient in the destination object space.

stops: a comma separated list of "offset color" pairs. This array can be any length, and can be a list of objects or arrays.

The *start: x/y* and *end: x/y* parameters can be specified as numbers (e.g., 0/20) or as strings (e.g., '5%'/'20%'). Numbers are interpreted as raw Scalable Vector Graphics (SVG) pixel coordinates. For example, start: x:0/y:0 is the object's top left corner. Negative numbers are not valid; any number less than zero is treated as zero. Numeric start/end coordinates are only useful for objects where you can set the area dimensions (e.g., the chart background area). For example, if the chart background/draw area is set to 200 by 200 pixels and start x:0/y:0 and stop x:100/y:0 is used to define a linear gradient, the gradient will be applied from the left edge to the center of the rectangle. If an object's size is calculated by the library (e.g., legend area, chart frame, risers, etc.), string percentages are typically used. For radial gradients, start: x/y defines the center of the gradient (e.g., start: x: '50%'/y: '50%' draws the gradient in the center of the object).

For radial gradients, the *radius* property defines the distance from the center to the outermost edge of the gradient. For example if an object is 200 by 200 pixels with a gradient start: x:'50%'/y:'50%' and radius: '100%', the outermost gradient edge will be 100 pixels beyond the edges of the object. In this example, 100% of 200 pixels is 200 pixels, so the gradient is actually 400 pixels from extreme left to extreme right. For a radial gradient that starts in the center (start: x:'50%'/y:'50%'), radius: '50%' is normally used to draw the gradient from the center to the outermost edges of the object.

The *stops:offset* values can be numbers or strings with '%'. Only numbers between 0 and 1 are valid, and are treated as percentages. A value 0.5 is the same as "50%". Numbers less than zero are treated as zero, and numbers greater than one are treated as 1.

string: To define a gradient in a string, use one of the following:

For a linear gradient, use a string in the following format:

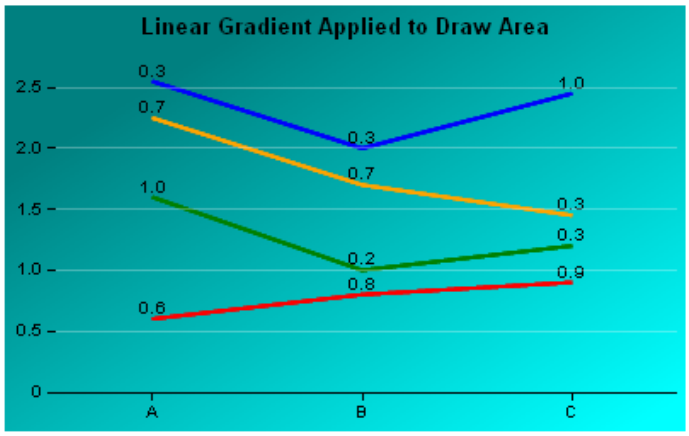
```
linear-gradient(startX, startY, endX, endY, stopsOffset
stopsColor,...stopsOffset stopsColor)
```

For a radial gradient, use a string in the following format:

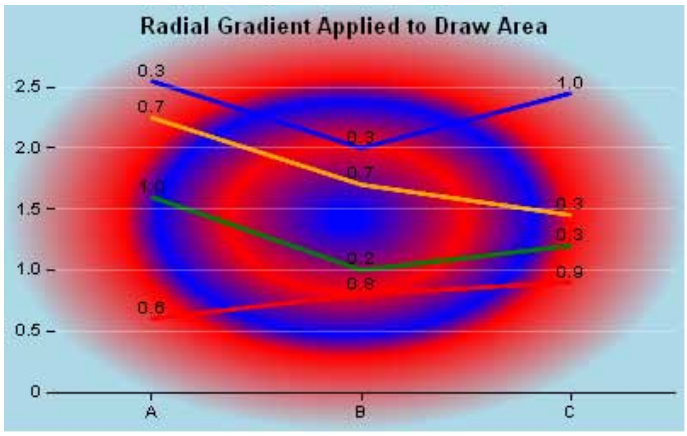
```
radial-gradient(startX, startY, radius, stopsOffset,
stopsColor,...stopsOffset stopsColor)
```

Examples:

```
fill: {
   color: 'linear-gradient(0,0,100%,100%, 20% teal, 95% cyan)'
   },
```




```
fill: {
   color: 'radial-gradient(50%,50%,50%, 20% blue, 35% red, 55% blue,
75% red, 1 lightblue)'
   },
```



You can learn more about SVG and defining gradients at: http://www.w3.org/TR/SVG/pservers.html.

Font Properties

All font properties are specified as a string (e.g., '10pt Sans-Serif') using the format defined at: http://www.w3.org/TR/CSS2/fonts.html#font-shorthand

Formatting Numbers

The numberFormat property can be used to specify the format of numeric labels. It accepts three different types of arguments:

- a traditional JSON object
- a format string
- a function

JSON object: The traditional JSON object includes the following properties to format the numeric labels:

```
numberFormat: {
   mode: 'numeric', // or 'percent', 'currency', 'scientific'
   thousandSep: ',',
   decimalSep: '.',
   decimalPlaces: 2,
   grouping: 'K', // One of 'K', 'M', 'B', 'T'
```

```
   prefix: 'a ',  // added to the front of the label
   suffix: ' b',  // added to the end of the label
}
```

mode: Use 'numeric', 'percent', 'currency', or 'scientific'

thousandSep: Character to separate values above and in multiples of a thousand (e.g., 1,000,000).

decimalSep: Character to separate decimals (e.g., 1.00)

decimalPlaces: Number of decimal places (i.e., number of digits to show to the right of the decimalSep character).

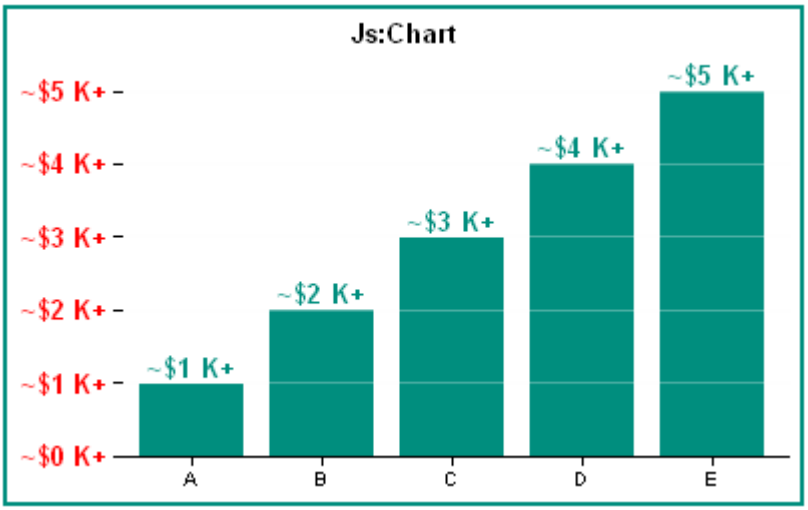
grouping: Use 'K', 'M', 'B', or 'T' to group large numbers by thousands (i.e., 1,000 = 1K), millions (i.e., 1 million = 1M), billions (i.e., 1 billing = 1B), or trillions (i.e., 1 trillion = 1T).

prefix: character(s) to be added to the front of the label

suffix: character(s) to be added to the end of the label

Example:

```
yaxis: {
   labels: {font: 'bold 10pt Sans-Serif', color: 'red'},
   numberFormat: {
         mode: 'currency',
         decimalPlaces: 0,
         grouping: 'K',
         prefix: '~',
         suffix: '+',
   }
}
```



Format String: You can also specify the format of numeric labels using a format string:

```
numberFormat: 'string'
```

The number format string must be in the format defined by the MSDN .NET Framework 4 Custom Numeric Format Strings defined at: http://msdn.microsoft.com/en-us/library/0c899ak8(vs.71).aspx

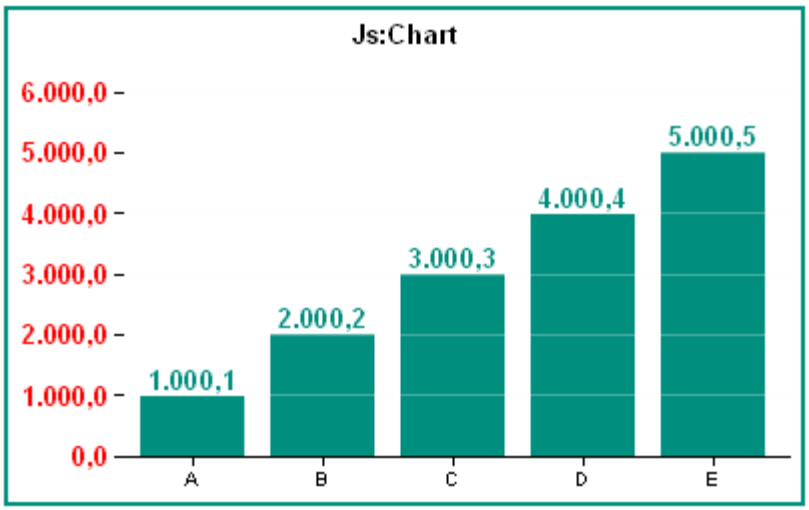
In addition to the format specifiers defined here, you may also use the following characters:

- 't' to specify a thousands separator character.

- 'd' to specify a decimals separator character.
- back tick (`): A percent sign (%) in a format string causes a number to be multiplied by 100 before it is formatted. Add a back tick in front of the percent sign (`%) to disable the multiplication.

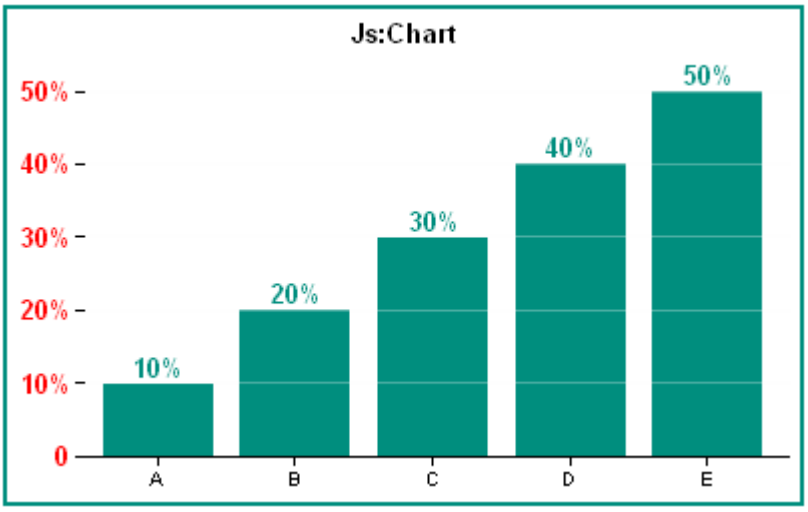
Example:

```
// format 4000.5 as '4.000,5' (European standard)
chart.yaxis.numberFormat = 't.d,#,#.#'
```



Example:

```
// format percentage without multiplier
chart.yaxis.numberFormat = '[>9]#`%;[<0]-#`%;[<=9]#`%;[=0]0'
```



Function: The numberFormat property can also process a user-defined function for fully custom number formatting callbacks. The expected function takes one argument (the number to format) and returns a string. Example:

```
// Return number with a '$' in front
chart.yaxis.numberFormat = function(n){ return '$' + n; };
```

This cannot be used inside the traditional JSON-style blocks { x: y }. It can only be used with the 'dot' notation as shown in this example.

NOTES:

- Do not use a semicolon (;) as a prefix, suffix, thousands separator, or decimal separator character. A semicolon is only valid is for separating multiple conditional formats (e.g., '[>=0]#,#;[<0]-#.#').
- Do not use a pound sign (#), period (.) or question mark (?) as a prefix or suffix character. These are special characters that cannot be interpreted as a string.
- If a percent sign (%) is used as the prefix or suffix character in a JSON object definition, the number format will use the percent mode regardless of the mode setting.
- If you use the string format to specify the range of data to which the number format is applied, make sure the format string includes the entire range of data. Example:

```
numberFormat: '[>9]#,#;[<0]-#.#;[<9]#.#;[=0]0'
```

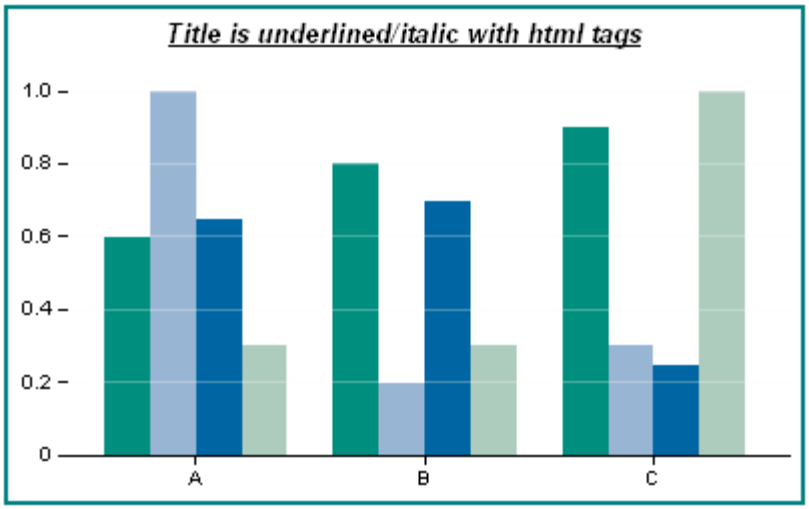
This format string defines the format for values less than nine and greater than nine but does not specify the format for a value equal to nine. The following corrected format string will cover the entire range of data:

```
numberFormat: '[>=9]#,#;[<0]-#.#;[<9]#.#;[=0]0'
```

HTML Codes in Strings

HTML codes can be embedded in any label or title strings: HTML codes in titles and labels are not processed by the library. They are simply passed on to the browser and will appear as they are rendered by the browser. Example:

```
title: {text: '<u>Title is underlined/italic with html tags</u>'},
```



Data & Property Definitions

The data that draws a chart is defined in the form of one or more arrays. Different chart types have different data requirements. The chartType property defines the number and format of values required to draw each chart type.

Null Data Support

You can use null, undefined, or multiple commas (,,) in a data array to identify missing data points. Js:Chart will allocate space for the missing data but the

riser/marker will not appear in the chart. Examples of valid missing data definitions:

```
[[1,2,null,4]]
[[1,2,undefined,4]]
[[1,2,,3]]
```

You cannot replace an entire array of data with null but an empty array is valid:

```
[[1,2], null, [3,4]] // invalid
[[1,2], [], [3,4]] // valid
```

Property Values

All properties can be assigned a value of undefined or null (not quoted strings). When undefined or null is used as a property value, the property is ignored. The default value assigned to the property in the default properties file (e.g., properties.js) is used.

Section 2: Methods

new tdgchart

This method creates a new chart with the properties specified by the input parameter *props*. If properties have not been specified, default properties will be used..

SYNTAX:

```
var chart = new tdgchart(props);
```

PARAMETERS:

props; identifies a variable where properties are defined

chart.draw ()

This method draws a chart.

SYNTAX:

```
chart.draw(aChartName);
```

PARAMETERS:

aChartName; identifies a chart name defined in a <div ... id='chartname'> tag.

EXAMPLE:

```
<div class='chart' id='chart1'>Optional text for unsupported
browsers.</div>;
chart.draw(chart1);
```

chart.set ()

This method sets a group of chart properties.

SYNTAX:

```
chart.set({group:{propertyName: value}});
```

PARAMETERS:

group; a property group name: blaProperties, pieProperties, legend, title, subtitle, footnote, xaxis, yaxis, datalabels, series

propertyName; a property name from the property group

value; a valid property value

EXAMPLE:

```
chart.set({dataLabels:{visible: false}});
chart.draw (chart1)
```

chart.redraw()

If chart.draw('htmlElement') has already been called, this function does a redraw to the same element. It will produce an error if draw() is ever called without a target or if redraw() is called before a target is set.

SYNTAX:

```
redraw();
```

registerEvent()

This method identifies a function to be called when an event (e.g., mouse click) occurs related to a specified object.

SYNTAX:

```
registerEvent(callback, event, object, userInfo, series, group,
misc);
```

PARAMETERS:

callback; the name of a function to be activated when *event* occurs

event; a string corresponding to any SVG event (e.g., 'click', 'mouseover'). A complete list of recognized event strings are defined at http://www.w3.org/TR/SVG/interact.html#SVGEvent

object; a string describing the chart object to register the event with. Object names are defined in the JSON API, and written in dot notation (e.g., 'yaxis.title'). Anything in the JSON API that is an object (not a base property) can be used for detection.

userInfo: (optional) a string of user information to show when *event* occurs

series; a number that identifies a specific instance of an object if the chart contains multiple objects associated with a series (e.g., series#.marker).

group; a number that identifies a specific instance of an object if the chart contains multiple objects associated with a group.

misc; a number that identifies a specific instance of an object

EXAMPLE:

```
function titleClick(obj, objName, userInfo) {
   alert(userInfo);
   if (!zoomed)return;
   byYear(this);
   this.redraw();
   zoomed = false;
}
chart.registerEvent(titleClick, 'click', 'title');
chart.registerEvent(titleClick, 'click', 'subtitle', "my callback");
```

NOTES:

- registerEvent() must be called before you draw the graph. Otherwise, you must issue a redraw in order to have the event hander applied.
- The event parameter is required to get FireFox events (e.g., titleClick(obj, objName, userInfo, event).

setSeriesColors()

This method uses the passed in array of colors as the series colors.

```
setSeriesColors(array)
```

PARAMETERS:

array; an array of colors

EXAMPLE:

```
setSeriesColors(['red', 'blue', 'green'])
```

setSeriesLabels()

This is a convenience function used to set all series labels with one functions.

SYNTAX:

```
setSeriesLabels(labels)
```

PARAMETERS:

labels; an array of series label strings. The array is expected to be initialized to the desired length, with the desired series entries, before calling setSeriesLabels(). If a series does not exist, the associated label is ignored

EXAMPLE:

```
chart.setSeriesLabels(['Widgets', 'Doodads', 'Thingies', 'FooBars']);
```

setSeriesProperty()

This method assigns an array of property values to a property for multiple series.

```
setSeriesProperty(prop, propArray)
```

PARAMETERS:

prop: a string matching a top level series dependent property (like 'color').

propArray a list to set the chosen property to.

EXAMPLE:

```
setSeriesProperty('color', ['red', 'blue', 'green'])
```

Abstraction Methods

buildClassName()

buildClassName is an abstraction method to generate a class name.

SYNTAX:

```
buildClassName = function(obj, s, g, m)
```

PARAMETERS:

obj; a string describing a chart object.

s: series number

g: group number

m: a number that identifies a specific instance of an object

EXAMPLE:

```
.className(function(){return
chart.buildClassName('waterfallProperties-connectorLine',
this.parent.index, this.index);})
.className(function(){return chart.buildClassName('marker',
this.parent.index, this.index);})
```

classNameLookup()

classNameLookup is an abstraction method to generate a class name.

SYNTAX:

```
classNameLookup = function(miscID)
```

PARAMETERS:

miscID: a string defining a chart object.

EXAMPLE:

```
.className(chart.classNameLookup('line'))
.className(chart.classNameLookup('wedge'))
```

Section 3: Properties

Properties define the overall appearance of the chart and individual chart objects.

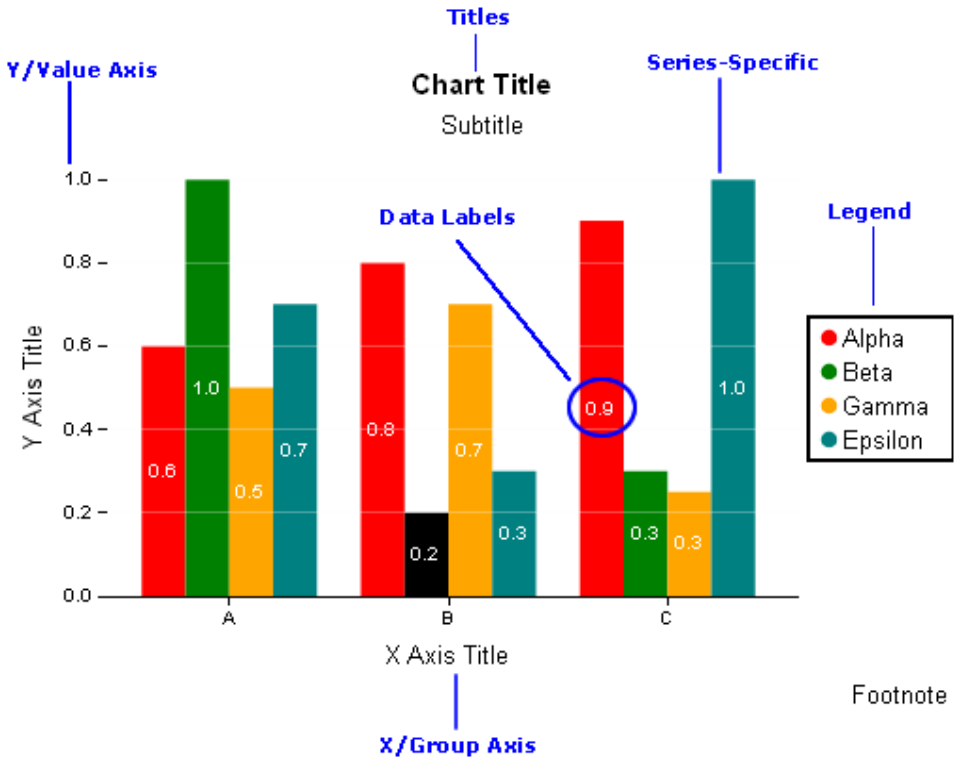


Chart-Wide Properties

These properties define the chart type (e.g., bar, line, area, pie, etc.) and control the general appearance of a chart. This code segment shows the default settings defined in properties.js:

```
chartType: 'bar',
catchErrors: true,
width: 400,
height: 250,
fill: { color: 'white'},
border: { width: 0, color: 'transparent', dash: '' },
chartFrame: {
   fill: {color: 'transparent'},
   border: {width: 0, color: 'transparent', dash: ''},
   shadow: false
},
introAnimation: {
   enabled: false,
   duration: 1000
},
interaction: {
   click: undefined,
   mousedrag: undefined,
   dblclick: undefined
},
mouseOverIndicator: {
   enabled: false,
   color: undefined,
   marker: {
```

```
         color: 'lightblue',
         size: 5,
         shape: 'circle',
         rotation: 0,
         position: 'top',
         border: {
               width: 1,
               color: 'darkblue',
               dash: ''
         }
   }
},
data: [
         [0.6, 0.8, 0.9],[1.0, 0.2, 0.3],
         [0.65, 0.7, 0.25],[0.3, 0.3, 1.0]],
dataSubset: {
   startGroup: undefined,
   stopGroup: undefined
},
swapData: false,
swapDataAndLabels: false,
riserCycleEndLightness: 0.8,
colorMode: 'bySeries',
colorModeColors: undefined,
labelPadding: 5,
chartsPerRow: undefined,
depth: undefined,
riserDepthGap: 0.2,
riserShadow: false,
riserBevel: undefined,
axisAutoLayout: {
   rotate45: true,
   rotate90: true,
   truncate: true,
   stagger: true,
   skip: true
},
dataLabels: {
   visible: true,
   displayMode: 'x',
   position: 'top',
   font: '7.5pt Sans-Serif',
   color: 'black',
   numberFormat: '[>9]#,#;[<0]-#.#;[<=9]#.#;[=0]0',
   formatCallback: undefined
   },
groupLabels: "ABCDEFGHIJKLMNOPQRSTUVWXYZ".split('')
```

axisAutoLayout

These properties control automatic layout of ordinal axis labels.

```
axisAutoLayout: {
   rotate45: boolean,
   rotate90: boolean,
   truncate: boolean,
   stagger: boolean,
   skip: boolean
},
```

PARAMETERS:

rotate45: true/false = allow/do not allow auto-layout to rotate labels 45 degrees to achieve the best automatic layout of ordinal axis labels. The default value is true.

rotate90: true/false = allow/do not allow auto-layout to rotate labels 90 degrees to achieve the best automatic layout of ordinal axis labels. The default value is true.

truncate: true/false = allow/do not allow auto-layout to truncate labels to achieve the best automatic layout of ordinal axis labels. The default value is true.

stagger: true/false = allow/do not allow auto-layout to stagger labels to achieve the best automatic layout of ordinal axis labels. The default value is true.

skip: true/false = allow/do not allow auto-layout to skip labels to achieve the best automatic layout of ordinal axis labels. The default value is true.

border

This property defines the border of the drawing area.

```
border: {
   width: Number,
   color: 'string' | JSON Object,
   dash: 'string'
}
```

PARAMETERS:

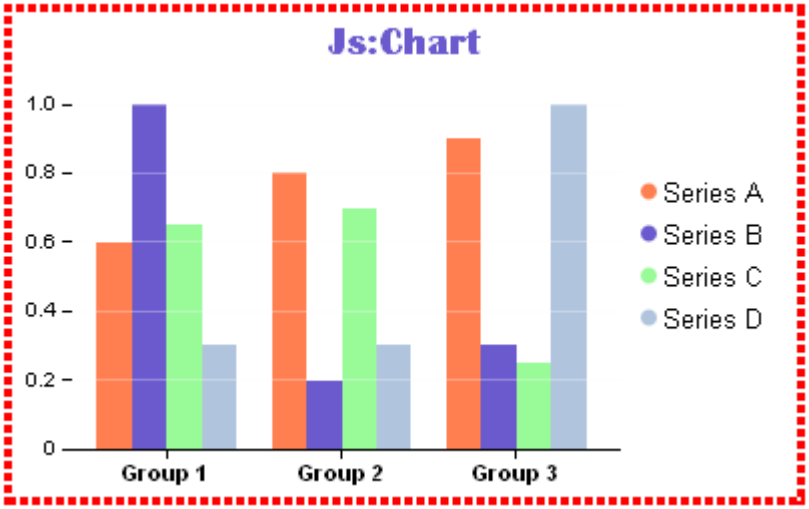
width: a number that defines the width of the border around the drawing area. The default value is zero (no border).

color: a color defined by a keyword or numerical specification string or a gradient defined by a string or JSON object. The default value is transparent. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string. See Gradient Definitions for the format of a linear or radial gradient.

dash: a string that defines the border's dash style. The default value is '' (a solid line). Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes (e.g., dash: '1 1' draws a dotted line).

EXAMPLE:

```
border: {
         width: 4,
         color: 'red',
         dash: '4 2'
}
```



catchErrors

This property defines how Java Script errors are handled. Typical errors can include invalid property settings or misplacement or missing commas and brackets that do not allow the library to parse the Java Script. When catchErrors is false, you can use the Java Script console (e.g., in Chrome Control+Shift+j) to determine the exact location of the error.

```
catchErrors: boolean
```

PARAMETERS:

catchErrors: true/false

true: errors are caught inside the chart library and displayed as HTML text where the chart would otherwise be drawn. Subsequent JavaScript code in the calling application continues to run.

false: errors are not caught inside the library, but are passed on to the calling application. If the calling application does not catch the error, all subsequent JavaScript execution terminates.

The default value is true.

chartFrame

This property defines the fill color and border of the chart frame.

```
chartFrame: {
   fill: {
         color: 'string' | JSON Object
   },
   border: {
         width: Number,
         color: 'string' | JSON Object,
         dash: 'string'
   },
   shadow: boolean | JSON Object
},
```

PARAMETERS:

fill/color: a color defined by a keyword or numerical specification string or a gradient defined by a string or JSON object. The default value is transparent. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string. See Gradient Definitions for the format of a linear or radial gradient.

border/width: Defines the width of the border around the chart frame. The default value is zero (no border).

border/color: a color defined by a keyword or numerical specification string or a gradient defined by a string or JSON object. The default value is transparent. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string. See Gradient Definitions for the format of a linear or radial gradient.

border/dash: A string that defines the border's dash style. The default value is '' (a solid line). Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes (e.g., dash: '1 1' draws a dotted line).

shadow: true/false enables/disables a default shadow on the chart frame. The default value is false. You may also use a JSON object in the following format to define the shadow:

```
shadow:
{
   angle: Number,
   distance: Number,
   blur: Number,
}
```

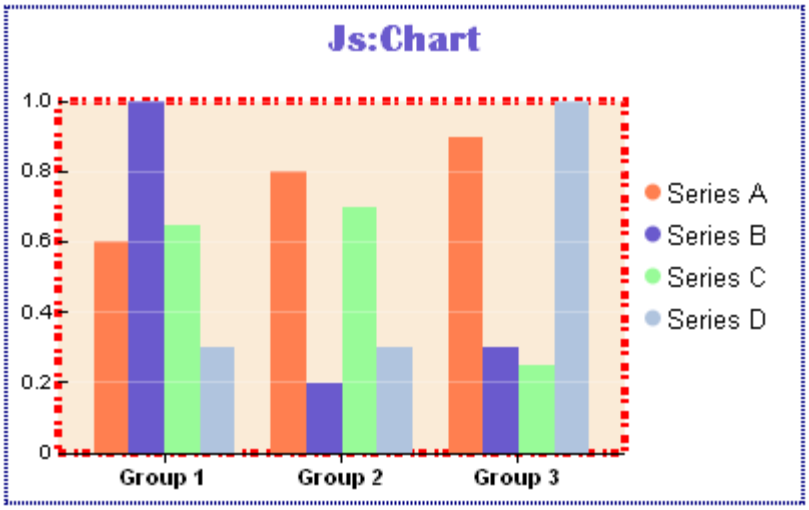
shadow/angle: a number in the range 0...360 defines the position of the light source. 0 = 12 o'clock, 90 = 3 o'clock. The default value is 315.

shadow/distance: a number defines the distance to push the shadow away from parent shape, in the global coordinate space (pixels by default). The default value is 10.

shadow/blur: a number in the range 0...1 defines the hardness of the shadow. 0 = perfectly hard edge, 1 = perfectly soft edge. The default value is zero.

EXAMPLE:

```
chartFrame: {
   fill: {color: 'antiquewhite'},
   border: {
         width: 4,
         color: 'red',
         dash: '2 4 4 2'
   }
},
```



chartsPerRow

For pies, gauges, and funnels that can have multiple charts in a single draw area, this property defines how many charts to draw in a horizontal row.

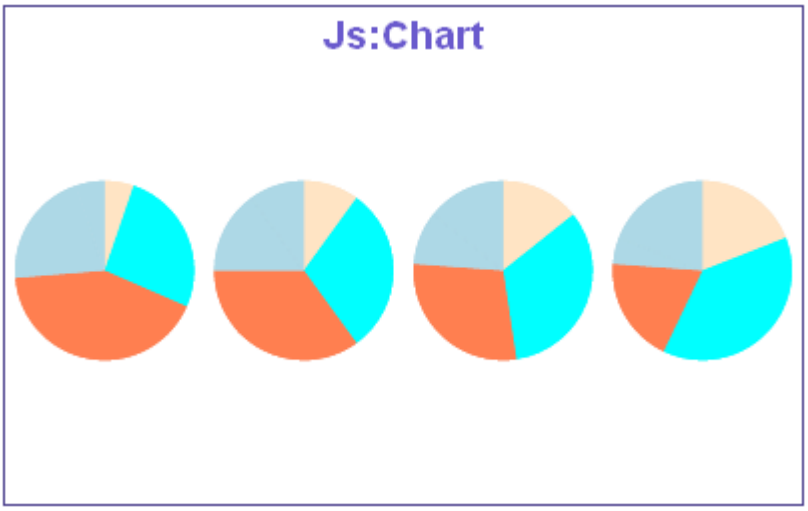
```
chartsPerRow: Number
```

PARAMETERS:

chartsPerRow: number of charts to draw in a horizontal row. The default value is undefined.

EXAMPLE:

```
chartType: 'pie',
chartsPerRow: 4
```



chartType

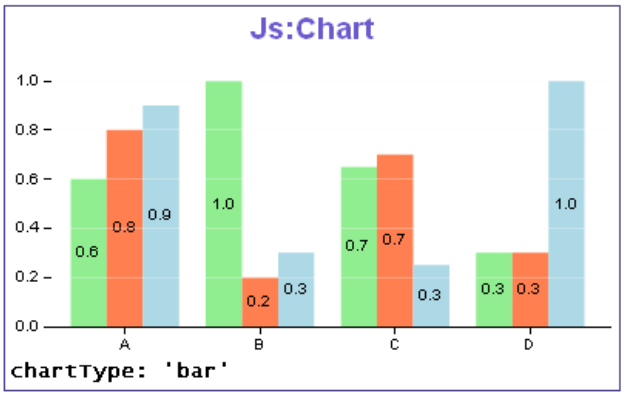
This property defines the chart type.

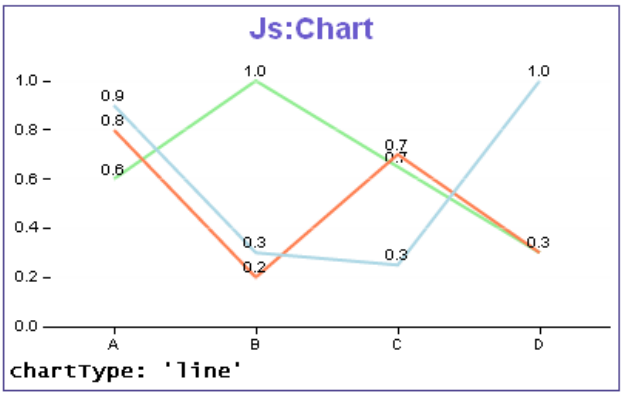
```
chartType: 'string'
```

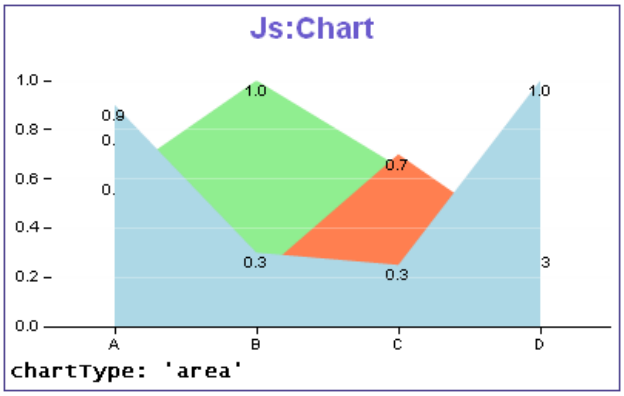
PARAMETERS:

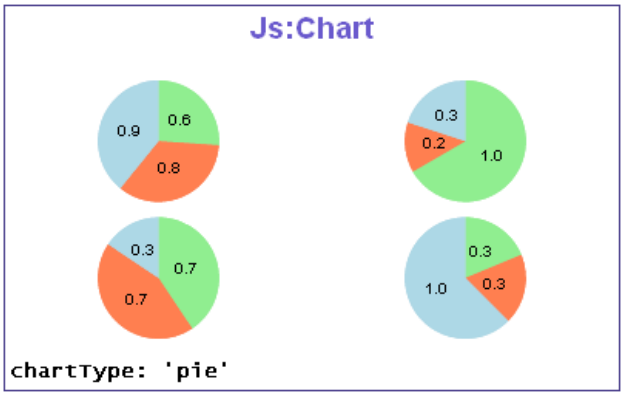
chartType: a string that defines the chart type: 'area', 'area3d', 'bar', 'bar3d', 'boxplot', 'bubble', 'bullet', 'funnel', 'gantt', 'gauge', 'heatmap, 'histogram', 'legend', 'line', 'pareto', 'pie', 'polar', 'radar', 'scatter', 'sparkline', 'stock', 'surface3d', 'treemap', 'waterfall'. The default value is 'bar'.

EXAMPLES:









NOTES:

- chartType: 'area3d', 'bar3d', 'surface3d': Use threedProperties to define the general layout of these charts.
- chartType: 'area': Use blaProperties to define the general layout of an area chart.
- chartType: 'bar': Use blaProperties to define the general layout of a bar chart.
- chartType: 'boxplot: Each series and group requires one array of five values. For example to define 1 series with 3 groups (3 boxes), you need an array of arrays of arrays such as: data = [[[1,2,3,4,5],[2,3,4,5,6],[3,4,5,6,7]]]. For a given series and group box, the first value is the minimum (lower 'hat'), the second defines the box bottom, the third value is median line, the fourth value defines the box top, and the fifth value defines the location of the top hat. Values must be in ascending order. Use boxPlotProperties to define the general appearance of a box plot. Use blaProperties:orientation to draw a horizontal or vertical chart. Box plot fill color and border are defined by the series color and border.

- chartType: 'bubble': Bubble charts require three values (X-Position, Y-Position, Size) to draw each bubble marker. Make sure your data set supplies the correct number of values for each chart type.
- chartType: 'bullet': Bullet is a microchart that is a variation of the bar chart. It is designed to show a quantitative measure against qualitative ranges. For example, a quantitative measure such as profit or revenue could be visualized against quality (good, better, best) and related markers. Chart title and subtitle properties define labels to draw on the edge of the chart. blaProperties: orientation draws the chart in horizontal/vertical format.
- chartType: 'funnel': A funnel chart is basically a pie chart that shows only one group of data at a time. The series in the group are stacked in the funnel with the first series at the top and the last series at the bottom. Use funnelProperties to define the general layout of a funnel chart. Series-specific properties control the color of funnel segments.
- chartType: 'gantt': Gantt charts require either two or three values for each riser. The first value is always a start time. The second value is a stop time (if ganttProperties.durationValues: false), or a duration (if ganttProperties.durationValues: true). The optional third value defines the series each riser belongs to. To define and format time values on the Y-axis, the first and second values must be formatted as time strings. See the examples in the .htm document for details. Use ganttProperties to define the general appearance of a gantt chart.
- chartType: 'gauge': Use gaugeProperties to define the general layout of the gauge. The numeric yaxis properties control scaling, tick marks, and colors assigned to segments of the gauge axis.
- chartType: 'heatmap': Heatmaps expect the same kind of data as Bar/Line/Area charts (an array of arrays). Each internal array forms a row in the heatmap table. Series labels are plotted on the left edge according to settings in zaxisOrdinal properties. Group labels are plotted on the bottom edge according to xaxisOrdinal properties. heatmapProperties control the color of the cells in the heatmap table.
- chartType: 'histogram': A histogram is a graphical representation that shows a visual impression of the distribution of data. Use histogramProperties to control the distribution of data. Use blaProperties:orientation to draw the chart in horizontal or vertical format. Use blaProperties:barGroupGapWidth to control the width of histogram risers. Use yaxis properties to control the format of Y-axis title and labels. Use xaxisOrdinal properties to control the format of X-axis title and labels. In a vertical histogram, the Y-axis is on the left side of the chart and the ordinal X-axis is drawn on the bottom of the chart. In a horizontal histogram, the ordinal X-axis is on the left side of the chart and the Y-axis axis is drawn on the bottom of the chart.
- chartType: 'legend': This chart type draws a chart where the legend *is* the chart. Selecting this chart type will resize the legend to the chosen width and height. Legend.position controls the entries layout: 'bottom' gives a horizontal legend. Any other position draws a vertical legend chart. Horizontal aligns all entries on the left edge, evenly spaced vertically. Vertical aligns entries along the top edge, evenly spaced horizontally. Use legend properties to control the format of the legend chart.
- chartType: 'line': Use blaProperties to define the general layout of a line chart. For Line Charts, use series:marker to control the visibility and format of markers at each data point.

- chartType: 'pareto': A pareto chart is similar to a combo chart. The first series (series zero) draws as an absolute bar chart on the Y-axis. The second series (series one) draws as a cumulative percent line on the Y2-axis.
- chartType: 'pie': Use pieProperties to define the general layout of the pie. Use the series-specific properties to color pie slices, show/hide data values, and explode/delete pie slices.
- chartType: 'polar': A polar chart is a circular scatter chart. Like scatter charts, a polar chart requires two values to draw each marker. polarProperties control the general appearance of a polar chart. yaxis properties control the circular grid around polar chart markers. These properties also control the appearance of axis labels and gridlines. xaxisNumeric properties control the appearance of labels and values on the X-axis (around the outside edge of the circular grid) and X-axis gridlines
- chartType: 'radar': A radar chart is a circular line chart. Radar charts require one value for each line segment (and marker if shown). The yaxis: majorGrid property draws the circular grid around the risers. Group axis (xaxisOrdinal) labels draw around the outside edge of the circle.
- chartType: 'scatter': Scatter charts require two values (X-Position/Y-Position) to draw each marker. Make sure your data set supplies the correct number of values for each chart type.
- chartType: 'sparkline': Sparkline is a microchart that has no titles, labels, or legends. It is a single line chart that is intended to be drawn in a very small area (e.g., height: 20, width: 50). Only the first series of data is plotted. Use the series-specific properties for color / line formatting. Use the yaxis properties to control automatic or manual scale and min / max values. The chart:fill and chart:border properties can be used to format the chart frame. The chart:labelPadding property can be used to define the empty space between the chart border and the line. blaProperties:orientation can be used to draw a horizontal or vertical chart. blaProperties:lineConnection can be used to control the format of the line: linear, curved, stepAfter, stepBefore, or stepBetween.
- chartType: 'stock': A high/low/open/close stock chart requires an array of 4 numbers for each riser: [high, low, open, close]. The first two numbers (high/low) define the high/low line that draws up and down from the box. The second two numbers (open/close) draw the open/close box riser. If either high or low is null, the high-low line is not drawn. If either open or close is null, the open-close box is not drawn. Use stockProperties to control the general appearance of a stock chart.
- chartType: 'treemap': Treemaps are defined by a data object. The object can be nested arbitrarily deep. Any properties in the data object with numeric values will be drawn as a single rectangle in the final chart.
- chartType: 'waterfall': Waterfall charts graphically illustrate the cumulative effect of sequentially introducing positive or negative values to an initial value. The initial and final (or total) values are represented by whole columns drawn from the ordinal axis baseline. Intermediate positive and negative values are drawn as floating columns. Use the properties in the waterfallProperties group to define the general layout of a waterfall chart.

colorMode

This property defines the color mode for drawing chart risers and markers. If colorMode is set to 'byInterpolation' or 'byHeight', the colorModeColors property must be set to an array of colors.

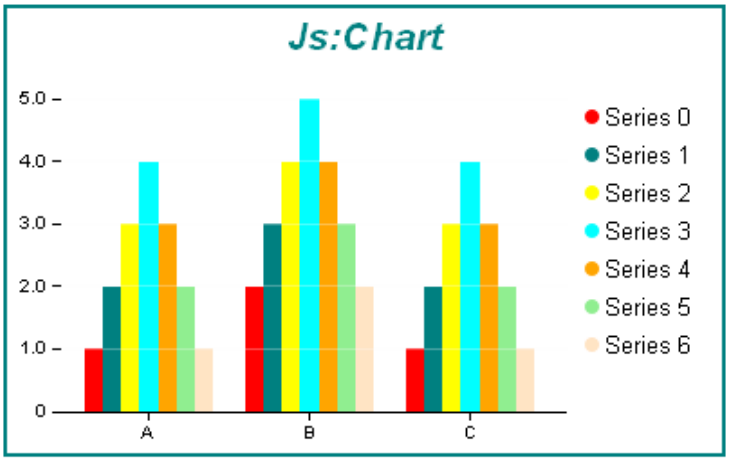
```
colorMode: 'string'
```

PARAMETERS:

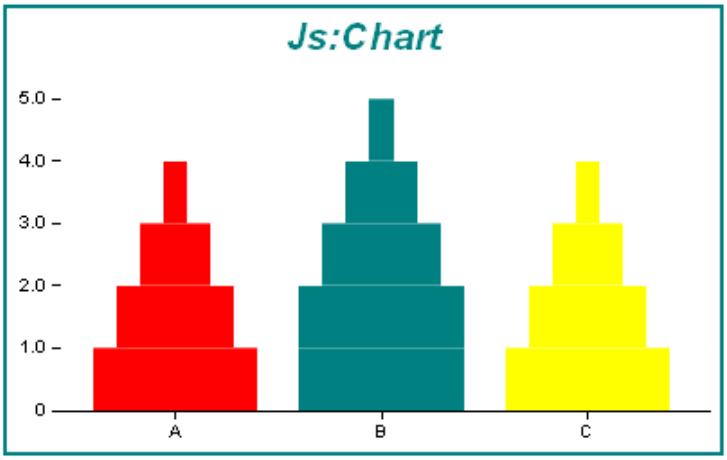
colorMode; a string that defines the chart's color mode: 'bySeries','byGroup', 'byHeight', or 'byInterpolation'. The default value is 'bySeries'. The 'byHeight' setting is only valid for charts with an ordinal axis and a single numeric axis.

EXAMPLE:

```
colorMode: 'bySeries'
series: [
   {series: 0, color: 'red'},
   {series: 1, color: 'teal'},
   {series: 2, color: 'yellow'},
   {series: 3, color: 'cyan'},
   {series: 4, color: 'orange'},
   {series: 5, color: 'lightgreen'},
   {series: 6, color: 'bisque'},
]
```




```
colorMode: 'byGroup'
```



colorModeColors

When the colorMode property is set to 'byInterpolation' or 'beHeight', this property defines the range of colors to apply to risers and markers. When colorMode is 'byInterpolation', the first series is colored the first color in this array, the last series is colored the last color, and the series in between are interpolated accordingly. When colorMode is 'byHeight', the colors defined here create a gradient such that the first color is at the numeric axis minimum, the last color is at the numeric axis maximum, and the remaining colors are interpolated between these. This gradient is then used to color the risers.

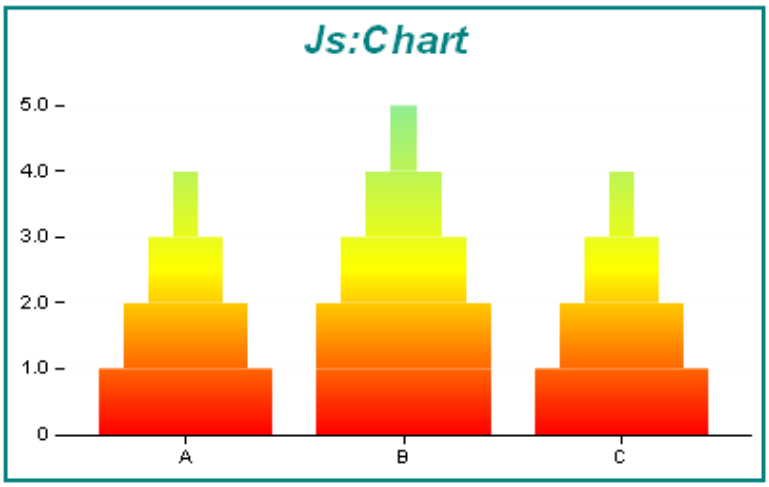
```
colorModeColors: ['string',...'string']
```

PARAMETERS:

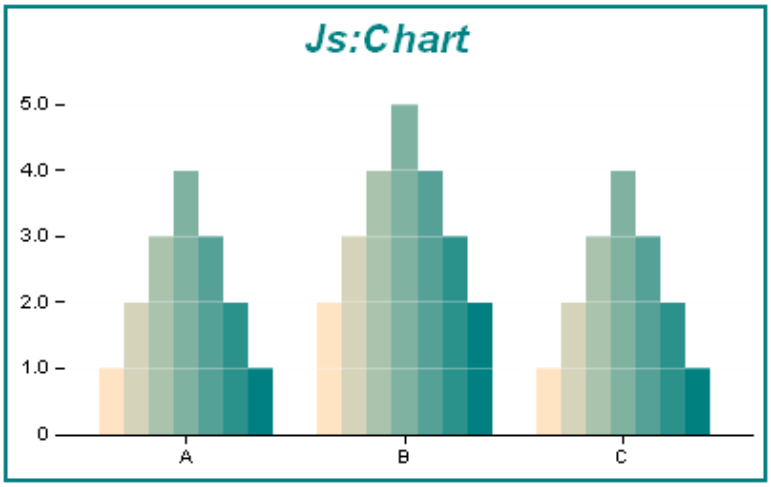
colorModeColors: an array of colors defined by a keyword or numerical specification string. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string. The default value is undefined.

EXAMPLE:

```
colorMode: 'byHeight',
colorModeColors: ['red', 'yellow','lightgreen']
```




```
colorMode: 'byInterpolation',
colorModeColors: ['bisque', 'teal']
```



data

This property defines the values that draw a chart.

```
data: [[value, value, ... value]]
```

data: one or more arrays of values. The number of values and the number of arrays depend on the chart type. See the chartType property for details. The default value is:

```
data: [
         [0.6, 0.8, 0.9],
         [1.0, 0.2, 0.3],
         [0.65, 0.7, 0.25],
         [0.3, 0.3, 1.0]
],
```

These default values will draw a simple bar, line, or area chart. Each value represents a data point in the chart.

This property is affected by the swapData and swapDataAndLabels properties. When these properties are false (the default), each value in an array is presented as riser/marker in a series. When these properties are set to true, each value in an array is presented as a riser/marker in a group.

In all cases, you can use null or undefined to identify a missing data point. See Data & Property Definitions for more information.

dataLabels

These properties control the visibility and format of data text labels for all series. Use the Series-Specific properties to enable/disable data text labels for individual series.

```
dataLabels: {
   visible: true,
   displayMode: 'string',
   position: 'string',
   font: 'string',
   color: 'string',
   numberFormat: JSON Object | 'string' | function(),
   formatCallback: function()
}
dataLabels: {
   font: '7.5pt Sans-Serif',
   color: 'black',
   numberFormat: '[>9]#,#;[<0]-#.#;[<=9]#.#;[=0]0',
   formatCallback: undefined
```

},

PARAMETERS:

visible; true/false controls the visibility of the data text labels. The default value is true. When visible is true, use the series showDataValues property to hide data labels for individual series.

displayMode: This property is only valid for bubble, scatter, and pie charts and defines the data text to show: 'x', 'y', 'z', or '%'. The default value is 'x'.

- For a bubble chart, use 'x', 'y', or 'z' to identify which value to show (X-Position, Y-Position, or Size (z)).
- For a scatter chart, use 'x' or 'y' to identify which value to show (X-Position, Y-Position).
- For a pie chart, use '%' to show the percentage value of a pie slice.

For % mode, you must set the numberFormat property to display the value as a percentage (e.g.,numberFormat: '[>9]#,#%;[<0]-#.#%;[<=9]#.##%;[=0]0').

You can use the formatCallback property and define a function to display multiple values. See the example below under the formatCallback property.

position: a string defines the position of data labels relative to the risers: 'bottom', 'center', 'insideBottom', 'insideTop', 'left', 'outside', 'right', 'top'. The default value is 'top'.

- The 'insideBottom' and 'insideTop' settings are for bar risers only and draw the data labels inside the bottom or top of the riser.
- For pie charts, the only valid settings are 'center' (on pie slices) and 'outside' (outside pie slices with feeler lines).
- For bar risers, positions are orientation aware (e.g., 'top' = 'right' in a horizontal bar).
- For bullet charts, also see the series-specific marker position property that defines the position of the marker relative to data text.

font; a string that defines the size and type face of data text labels. See http://www.w3.org/TR/CSS2/fonts.html#font-shorthand for the format of this string. The default value is font: '7.5pt Sans-Serif'.

color; a string that defines the color of data text labels. See http://www.w3.org/TR/css3-color/#colorunits for the format of this string. The default value is 'black'.

numberFormat: Use a string, JSON object, or function to define the format of data labels. See Properties Overview/Formatting Numbers in Section 1 for details and examples. The default value is '[>9]#,#;[<0]-#.#;[<=9]#.#;[=0]0'.

formatCallback: Identify a function to call for each data label. The default value is undefined. This function can use up to three arguments (data value, series ID, and group ID) and should return a string to be displayed.

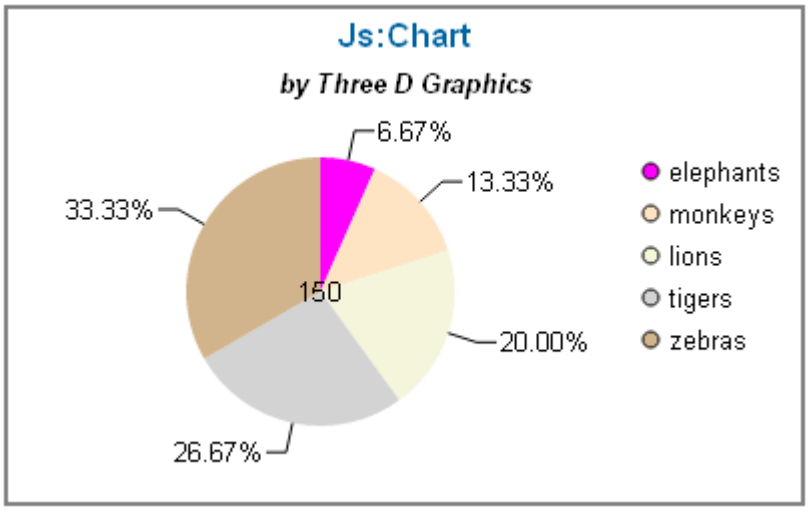
```
function myCallback(dataValue, seriesID, groupID);
```

As an example, you can use a user-defined function to show x, y, and z values in a bubble chart:

```
chart.dataLabels.formatCallback = function(d){
    return d[0] + ', ' + d[1] + ', ' + d[2];
}
```

EXAMPLES:

```
data: [[10,20,30,40,50]],
chartType: 'pie',
legend: {visible: true},
pieProperties: {totalLabel: {visible: true}},
dataLabels: {
   displayMode: '%',
   position: 'outside',
   font: '10pt Sans-Serif',
   numberFormat: '[>9]#,#%;[<0]-#.#%;[<=9]#.##%;[=0]0'
   },
series: [
   {series: 0, color: 'magenta',label: 'elephants'},
   {series: 1, color: 'bisque', label: 'monkeys'},
   {series: 2, color: 'beige', label: 'lions'},
   {series: 3, color: 'lightgrey', label: 'tigers'},
   {series: 4, color: 'tan', label: 'zebras'}
]
```



dataSubset

These properties define the range of data to draw in the chart. It can be used with the interaction properties (e.g., mouseDrag: pan) to define the range of data that is visible following a mouseDrag interaction.

```
dataSubset: {
   startGroup: number,
   stopGroup: number,
}
```

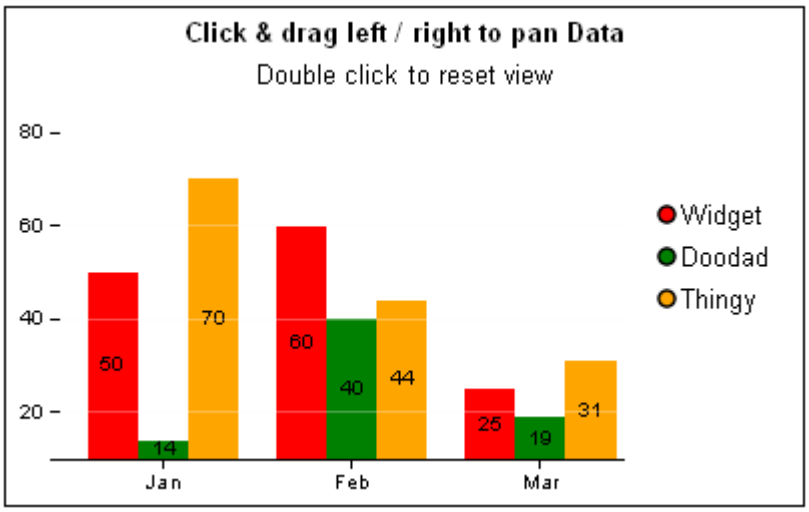
PARAMETERS:

startGroup: zero-based starting group number to show. The default value is undefined.

stopGroup: zero-based number of groups to show. The default value is undefined.

EXAMPLE:

```
chart.interaction.mousedrag = 'pan';
chart.data = [
   [50, 60, 25, 20, 30, 17, 44, 31, 19, 78, 50, 11],
   [14, 40, 19, 60, 38, 44, 23, 21, 30, 55, 77, 47],
   [70, 44, 31, 19, 33, 53, 10, 30, 40, 50, 22, 11]
];
chart.dataSubset.startGroup = 0;
chart.dataSubset.stopGroup = 3;
chart.groupLabels = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul',
'Aug', 'Sep', 'Oct', 'Nov', 'Dec'];
```



depth

This properties applies a 2.5D depth effect to bar charts, line charts, area charts, pie charts, and any 'bar-like' chart (gantt, histogram, etc.).

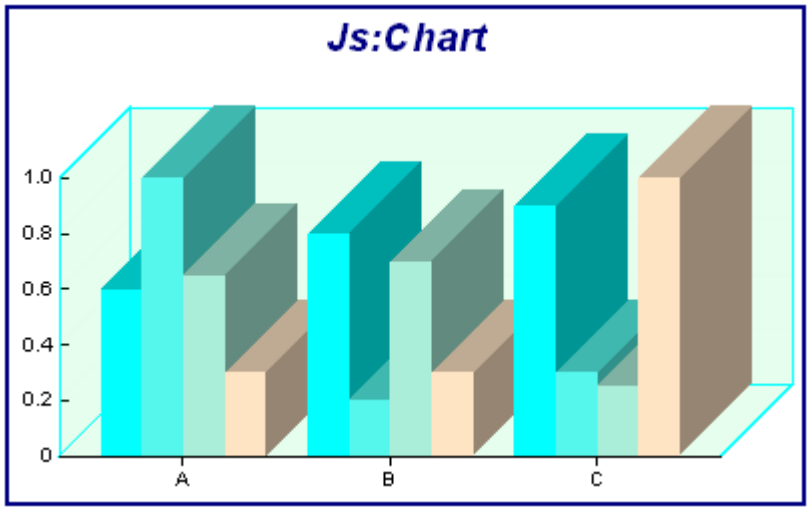
```
depth: Number
```

PARAMETERS:

depth: a number 0...100 that specifies the amount of depth to apply to risers and the chart frame. Use zero or undefined for no 2.5D depth effect. The default value is undefined.

EXAMPLE:

```
chartFrame: {border: {width: 1, color: 'cyan'}, fill: {color:
'hsla(140, 100%, 50%, 0.1)'}},
colorMode: 'byInterpolation',
colorModeColors: ['cyan', 'bisque'],
blaProperties: {orientation: 'vertical'}
depth: 50,
```



fill

This property defines a color or gradient to be applied to the draw area.

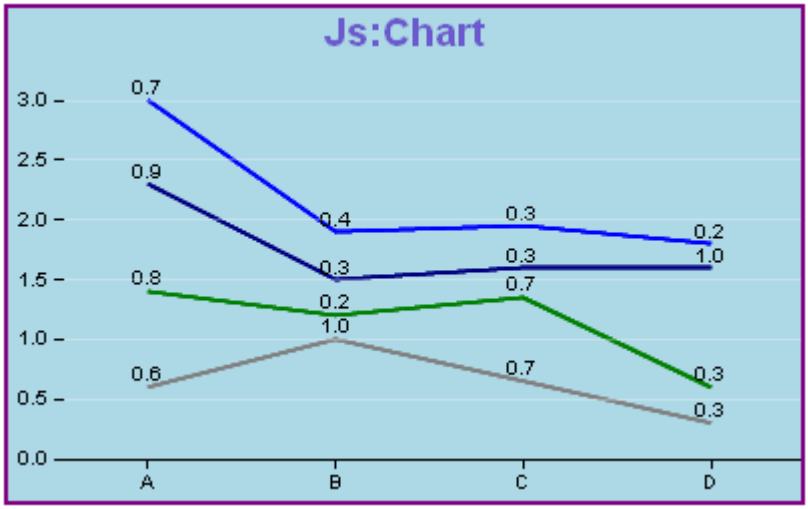
```
fill: {
   color: 'string' | JSON Object
}
```

PARAMETERS:

color: a color defined by a keyword or numerical specification string or a gradient defined by a string or JSON object. The default value is white. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string. See Gradient Definitions for the format of a linear or radial gradient.

EXAMPLE:

```
fill: {
         color: 'LightBlue'
         },
border: {
         width: 4,
         color: 'purple'
}
```



groupLabels

For all chart types except histogram, this property defines the chart's group labels.

```
groupLabels: 'string' | ['string'...'string']
```

PARAMETERS:

groupLabels: can be a string or an array of strings. A string will be split into an array along any space character. The default value is "ABCDEFGHIJKLMNOPQRSTUVWXYZ".split('').

NOTES:

- Use xaxisOrdinal:labels to format group labels.
- You can define date/time labels in this property and enable xaxisOrdinal:timeAxis to create a time-based ordinal axis. See the timeAxis property for more details and examples.

height/width

These properties define the width and height (in pixels) of the chart draw area.

```
width: number
height: number
```

PARAMETERS:

width: defines the width of the chart draw area (in pixels). The default value is 400.

height: defines the height of the chart draw area (in pixels). The default value is 250.

interaction

This property defines allowable user interaction with the chart. For bar, line, area, bubble, and scatter charts, mousedrag: 'pan' will reposition the risers/markers, axis labels, and data labels (if shown) when a user clicks and drags the mouse inside the chart frame. For 3D charts, mousedrag: 'rotate' will rotate the chart frame when a user clicks and drags the mouse around the chart frame. When an other slice is defined with the pieProperties: otherSlice property, the click: 'otherSliceDrillDown' option will drill-down into the other slice components (i.e., the drilled-down pie chart will show the slices that make up the other slice).

```
interaction: {
   click: 'string'
   mousedrag: 'string'
   dblclick: 'string'
},
```

PARAMETERS:

click: a string that defines interaction on mouse click: undefined or 'otherSliceDrillDown'. The default value is undefined.

mousedrag: a string that defines the interaction on mouse drag: 'select', 'pan', 'riserDrag' (for chartType: bar/blaProperties: seriesLayout: 'sideBySide' only), 'rotate' (for 3D charts only), or undefined. The default value is undefined.

dblclick: a string that defines the interaction on mouse double click: 'resetView' or undefined. The default value is undefined.

introAnimation

This property enables/disables and defines the duration of animation. When enabled, risers/markers are animated when the chart is drawn. Bar/Line/Area risers enter the chart from the base of the frame and become static when they reach their assigned values. Circular risers (e.g., bubble markers) are animated from the inside to the outer edge. Pie slices are drawn clockwise. Animation can be applied to all chart types except 3D charts and charts where 2.5 depth has been applied with the depth property.

```
introAnimation: {
   enabled: boolean,
   duration: number
},
```

PARAMETERS:

enabled: true = enable animation, false = disable animation. The default value is false

duration: a number that defines the duration of the animation. Larger numbers produce slower animation. The default value is 1000.

labelPadding

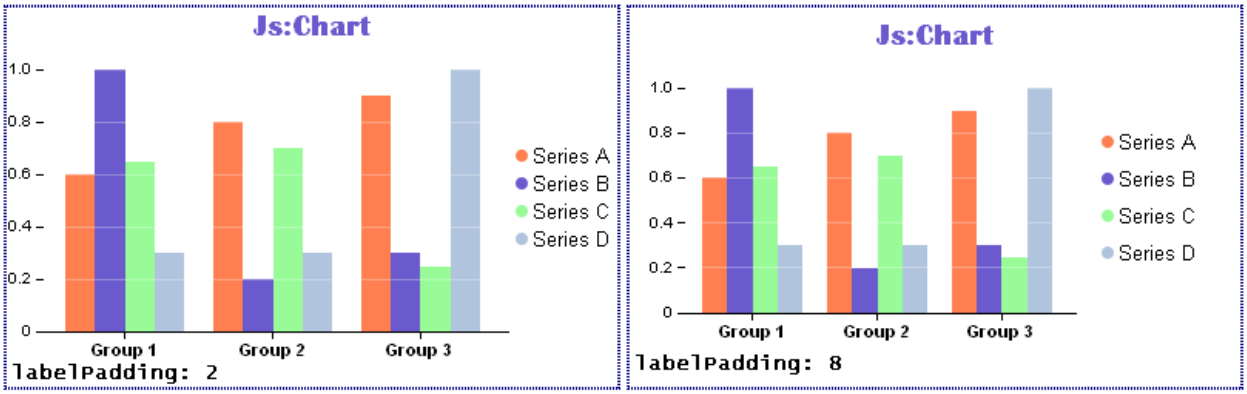
This property defines the amount of space to leave between labels/titles and the space between labels/titles and the chart frame.

```
labelPadding: Number
```

PARAMETERS:

labelPadding; label padding value. The default value is 5.

EXAMPLES:



mouseOverIndicator

This property defines a riser color and/or a marker to draw when the mouse hovers over a riser.

```
mouseOverIndicator: {
   enabled: boolean,
   color: 'string'  | JSON Object | undefined,
   marker: {
         color: 'string' | JSON Object,
         size: Number,
         shape: 'string',
         rotation: Number,
         position: 'string',
         border: {
               width: Number,
               color: 'string',
               dash: 'string'
         }
   }
}
```

PARAMETERS:

enabled: true/false enables/disables the mouse over indicator. The default value is false (disabled).

color: defines a color to apply when the mouse hovers over a riser. The color can be defined by a keyword or numerical specification string or a gradient defined by a string or JSON object. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string. See Gradient Definitions for the format of a linear or radial gradient. The default value is undefined.

marker: these properties define a marker to draw when the mouse hovers over a riser.

marker/color: defines the color of the marker. The color can be defined by a keyword or numerical specification string or a gradient defined by a string or

JSON object. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string. See Gradient Definitions for the format of a linear or radial gradient. The default value is 'lightblue'.

marker/size: a number that defines the size of the marker. The default value is 5.

marker/shape: a string that defines the shape of markers for a series: arrow, bar, circle, cross, diamond, fiveStar, hexagon, hourglass, house, pirateCross, plus, sixStar, square, thinPlus, tick, or triangle. Note that bar, cross, and tick markers require a border width and color. The default value is 'circle'.

marker/rotation: a number 0...360 that defines the angle (in degrees) of the marker. The default value is zero.

marker/position: a string defines the position of the marker. The default value is 'top'.

marker/border/width: a number defines the width of the marker border in pixels. The default value is 1.

marker/border/color: a color defined by a keyword or numerical specification string. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string. The default value is 'darkblue'.

marker/border/dash: a string that defines the border dash style. Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes. The default value is '' (a solid line).

riserBevel

This property applies a bevel to risers in a bar chart, markers in bubble/scatter charts, and slices in a pie chart.

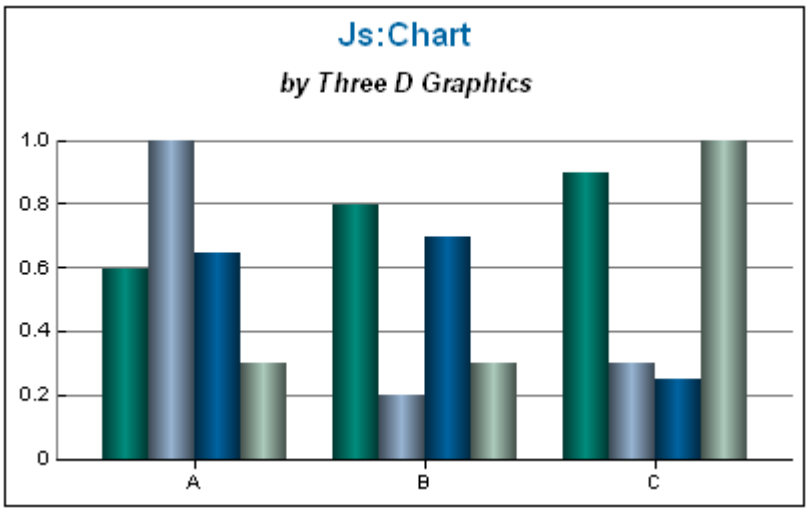
```
riserBevel: bevel: 'string'
```

PARAMETERS:

bevel: a string that defines the bevel type. For bar charts, use one of: bevel, cylinder, darken, darkenInverted, lighten, or lightenInverted. For pie charts, use one of: bevel, cylinder, or donut. The default value is undefined.

EXAMPLES:

```
riserBevel: 'cylinder'
```



riserCycleEndLightness

If there are more series than series colors defined, the charting library cycles through the defined colors. This property defines the lightness / darkness of the final riser color cycle.

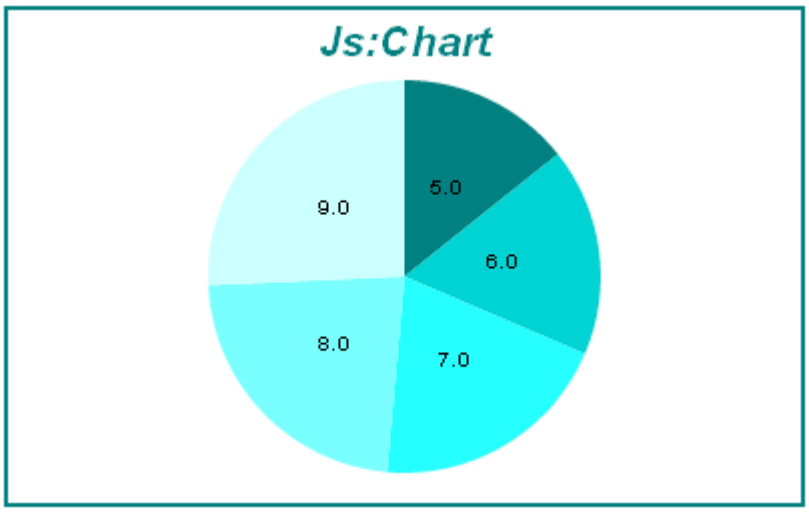
```
riserCycleEndLightness: Number
```

PARAMETERS:

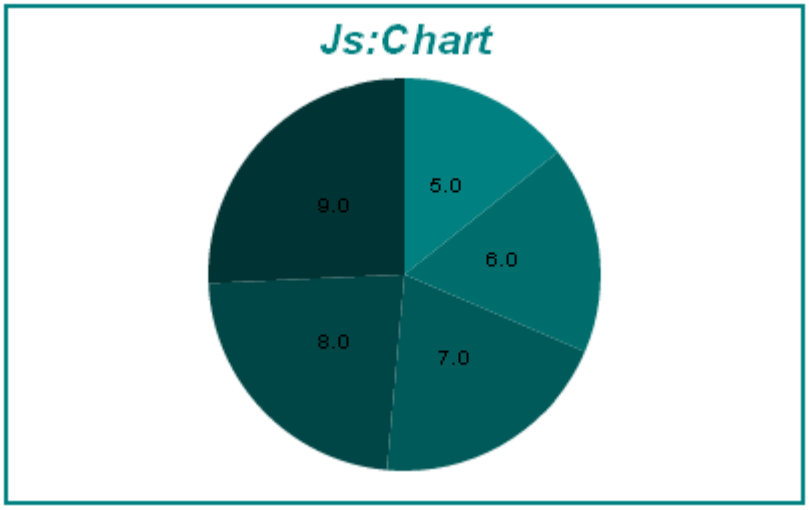
riserCycleEndLightness; a number in the range 0...1 controls the brightness of colors assigned to undefined series risers. Values less 0.5 darkens each series cycle. Values greater than 0.5 lightens each series cycle. The default value is 0.8.

EXAMPLES:

```
data: [[5,6,7,8,9]],
riserCycleEndLightness: 0.9,
chartType: 'pie',
series: [{series: 0, color: 'teal'}]
```




```
riserCycleEndLightness: 0.1,
```



riserDepthGap

In 3D chart and charts where 2.5D depth is applied to a chart with the depth property, this property adds space between risers that draw behind other risers.

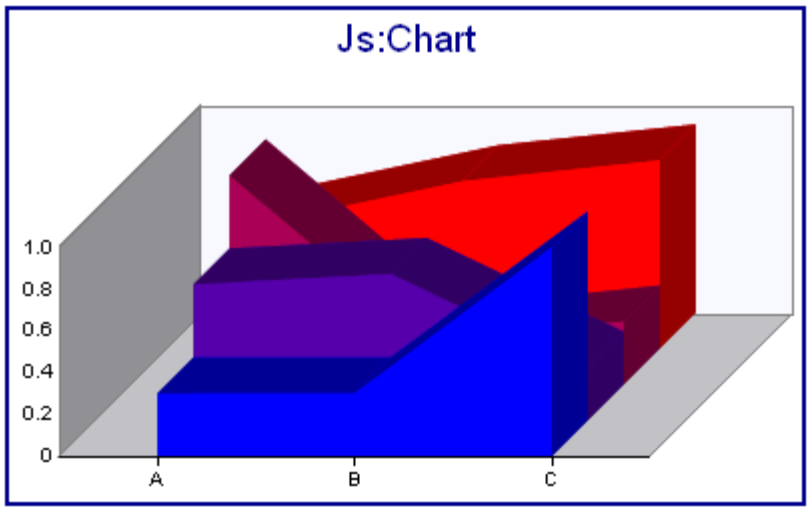
```
riserDepthGap: number
```

PARAMETERS:

riserDepthGap: a number in the range 0...1 (as a factor of the depth) that defines the margin between the risers. The default value is 0.2.

EXAMPLE:

```
chartType: 'area'
depth: 25,
riserDepthGap: 0,
chartFrame: {border: {width: 1, color: 'grey'}, fill: {color:
'ghostwhite'}},
dataLabels: {visible: false},
colorMode: 'byInterpolation',
colorModeColors: ['red', 'blue'],
blaProperties: {orientation: 'vertical'},
```



riserShadow

This property applies a shadow to chart risers/markers.

```
riserShadow: boolean | JSON object
```

PARAMETERS:

riserShadow: true/false enables/disables a default shadow on chart risers. The default value is false. You can also use a JSON object in the following format to define the shadow:

```
riserShadow:
{
   angle: Number,
   distance: Number,
   blur: Number,
}
```

riserShadow/angle: a number in the range 0...360 defines the position of the light source. 0 = 12 o'clock, 90 = 3 o'clock. The default value is 315.

riserShadow/distance: a number defines the distance to push the shadow away from parent shape, in the global coordinate space (pixels by default). The default value is 10.

riserShadow/blur: a number in the range 0...1 defines the hardness of the shadow. 0 = perfectly hard edge, 1 = perfectly soft edge. The default value is zero.

NOTES:

Other chart objects can be assigned shadows. See chartFrame: shadow and legend: shadow.

swapData

This property can be used to swap series and group orientation. When false (the default), values in a row in a data set are represented as series. When true, values in a column are represented as series.

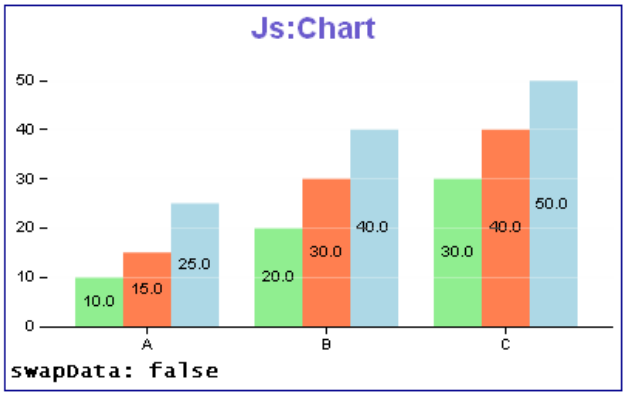
```
swapData: boolean
```

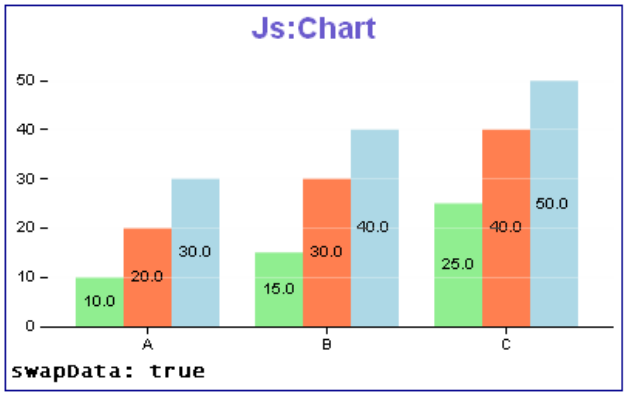
PARAMETERS:

swapData: true = swap series/group orientation, false (default) = do not swap series/group orientation

EXAMPLE:

```
data: [
          [10, 20, 30],
          [15, 30, 40],
          [25, 40, 50]
]
```





swapDataAndLabels

This property can be used to swap series and group orientation. It is the same as the swapData property except it also swaps the series and group labels. When false (the default), values in a row in a data set are represented as series. When true, values in a column are represented as series.

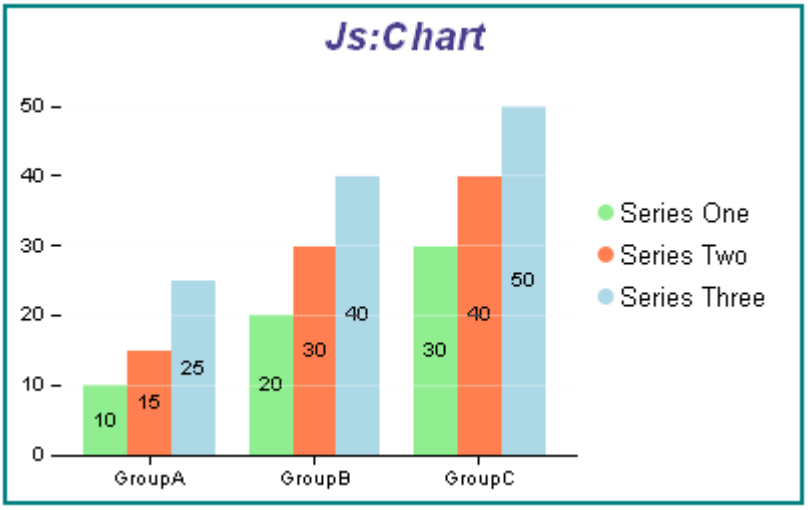
```
swapDataAndLabels: boolean
```

PARAMETERS:

swapData: true = swap series/group/ label orientation, false (default) = do not swap series/group/label orientation

EXAMPLES:

```
data: [  [10, 20, 30],
         [15, 30, 40],
         [25, 40, 50]],
swapDataAndLabels: false
```




```
data: [  [10, 20, 30],
         [15, 30, 40],
         [25, 40, 50]],
swapDataAndLabels: true
```

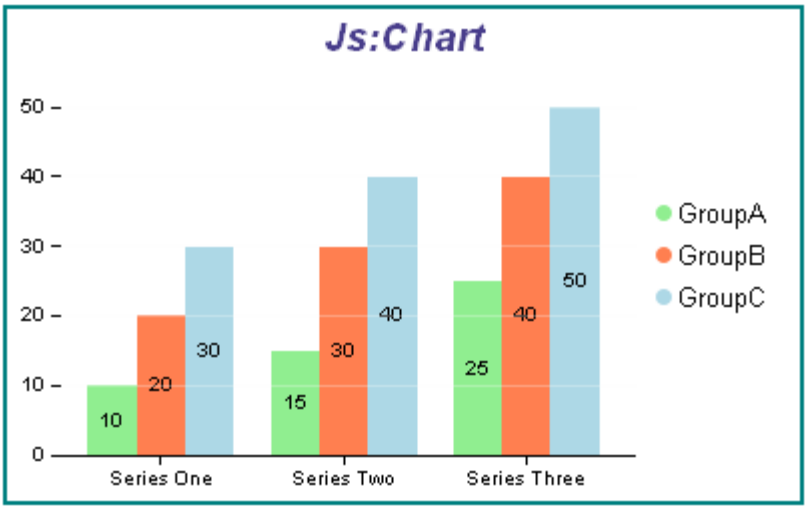


Chart Titles Properties

These properties control the format and visibility of the chart title, subtitle, and footnote. The following code segment shows the default values from the properties.js file:

```
title: {
      text: 'Chart Title',
      visible: true,
      align: 'center',  // One of 'left', 'center', 'right'
      font: 'bold 10pt Sans-Serif',
      color: 'black'
      tooltip: undefined
},
subtitle: {
      text: 'Chart Subtitle',
      visible: false,
      align: 'center',
      font: '10pt Sans-Serif',
      color: 'black'
      tooltip: undefined
},
footnote: {
      text: 'Chart Footnote',
      visible: false,
      align: 'right',
      font: '10pt Sans-Serif',
      color: 'black'
      tooltip: undefined
},
```

ALSO SEE:

legend: title

xaxisNumeric: title

xaxisOrdinal: title

yaxis: title

y2axis: title

zaxisOrdinal: title

title

These properties control the content, visibility, and format of the Chart Title.

```
title: {
   text: 'string',
   visible: boolean,
   align: 'string',
   font: 'string',
   color: 'string'
   tooltip: 'string' | function()
}
```

PARAMETERS:

text; a string that defines the Chart Title text. The default value is 'Chart Title'.

visible; true/false controls the visibility the Chart Title. The default value is true (visible).

align; a string that defines the alignment of the Chart Title: 'center', 'chartFrame' (center aligned with the chart frame), 'left', or 'right'. The default value is 'center'.

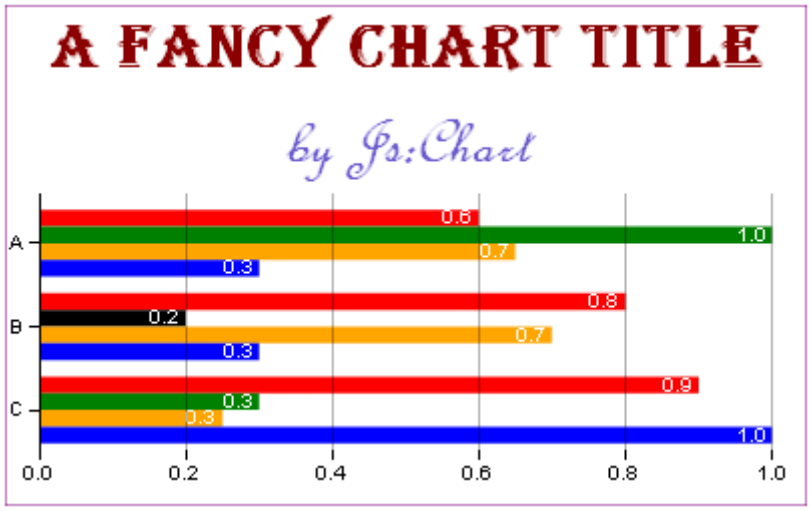
font; a string that defines the size and type face of the Chart Title. See http://www.w3.org/TR/CSS2/fonts.html#font-shorthand for the format of this string. The default values is 'bold 10pt Sans-Serif'.

color; a string that defines the color of the Chart Title. See http://www.w3.org/TR/css3-color/#colorunits for the format of this string. The default value is 'black'.

tooltip: a string or a function that returns a string to show when the mouse hovers over the title. The default value is undefined. A callback function is invoked with three arguments: value, series ID, and group ID. For non-riser objects, set the arguments to undefined. 'this' inside the callback function is set to the parent chart and provides full access to the property tree and API.

EXAMPLE:

```
title: {
   text: 'A FANCY CHART TITLE',align: 'center',
   font: 'Bold 24pt Algerian',color: 'darkred'
}
```



subtitle

These properties control the content, visibility, and format of the Chart Subtitle.

```
subtitle: {
   text: 'string',
   visible: boolean,
   align: 'string',
   font: 'string',
   color: 'string'
   tooltip: 'string' | function()
}
```

PARAMETERS:

text; a string that defines the Chart Subtitle text. The default value is 'Chart Subtitle'.

visible; true/false controls the visibility the Chart Subtitle. The default value is false.

align; a string that defines the alignment of the Chart Subtitle: 'center', 'chartFrame' (center aligned with the chart frame), 'left', or 'right'. The default value is 'center'.

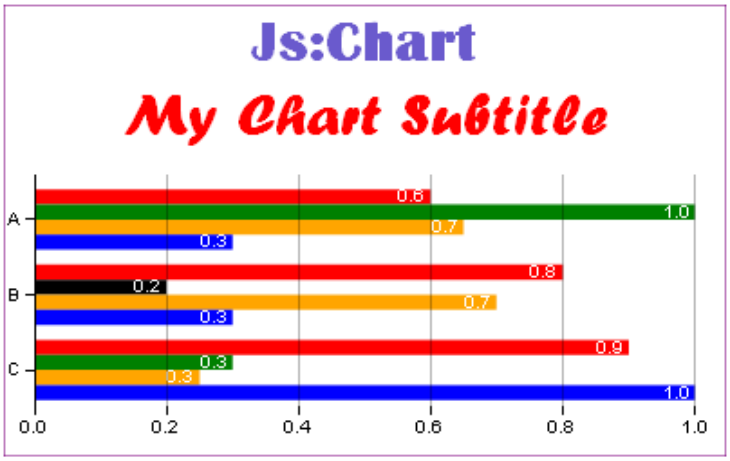
font; a string that defines the size and type face of the Chart Subtitle. See http://www.w3.org/TR/CSS2/fonts.html#font-shorthand for the format of this string. The default value is '10pt Sans-Serif'.

color; a string that defines the color of the Chart Subtitle. See http://www.w3.org/TR/css3-color/#colorunits for the format of this string. The default value is black.

tooltip: a string or a function that returns a string to show when the mouse hovers over the subtitle. The default value is undefined. A callback function is invoked with three arguments: value, series ID, and group ID. For non-riser objects, set the arguments to undefined. 'this' inside the callback function is set to the parent chart and provides full access to the property tree and API.

EXAMPLE:

```
subtitle: {
   text: 'My Chart Subtitle',visible: true,align: 'center',
   font: 'Bold 24pt Forte',color: 'red'
}
```



footnote

These properties control the content, visibility, and format of the Chart Footnote.

```
footnote: {
   text: 'string',
   visible: boolean,
   align: 'string',
   font: 'string',
   color: 'string'
   tooltip: 'string' | function()
}
```

PARAMETERS:

text; a string that defines the Chart Footnote text. The default value is 'Chart Footnote'.

visible; true/false controls the visibility the Chart Footnote. The default value is false.

align; a string that defines the alignment of the Chart Footnote: 'center', 'chartFrame' (center aligned with the chart frame), 'left', or 'right'. The default value is 'center'.

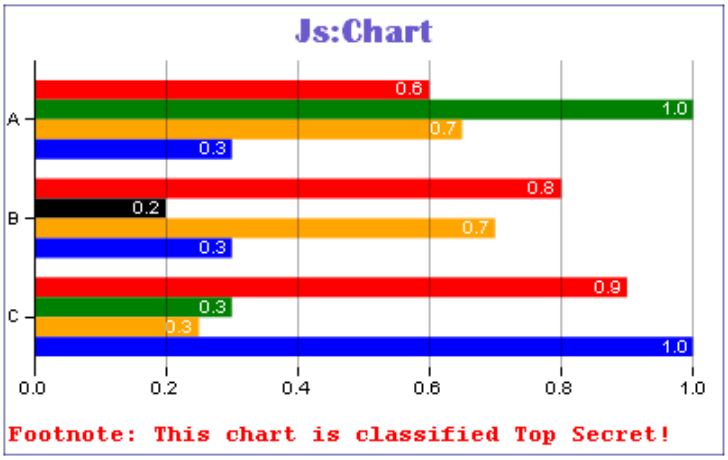
font; a string that defines the size and type face of the Chart Footnote. See http://www.w3.org/TR/CSS2/fonts.html#font-shorthand for the format of this string. The default value is '10pt Sans-Serif'.

color; a string that defines the color of the Chart Footnote. See http://www.w3.org/TR/css3-color/#colorunits for the format of this string. The default value is 'black'.

tooltip: a string or a function that returns a string to show when the mouse hovers over the footnote. A callback function is invoked with three arguments: value, series ID, and group ID. For non-riser objects, set the arguments to undefined. 'this' inside the callback function is set to the parent chart and provides full access to the property tree and API. The default value is undefined.

EXAMPLE:

```
footnote: {
   text: 'Footnote: This chart is classified Top Secret!',
   font: 'Bold 10pt Courier',visible: true,align: 'left',color: 'red'
}
```



Legend Properties

These properties control the visibility and format of the legend. This code segment shows the default values from properties.js:

```
legend: {
   visible: false,
   position: 'right',  // One of 'free', 'left', 'right', 'bottom'
   xy: {x:330, y:80},
   markerSize: 8,
   markerPosition: 'left',  // One of 'left', 'right', 'bottom'
   maxEntries: undefined,
   title: {
         visible: false,
         text: 'Legend Title',
         font: '10pt Sans-Serif',
         color: 'black'
   },
   labels: {
         font: '7.5pt Sans-Serif',
         color: 'black'
   },
   lineStyle: {
         width: 0,
         color: 'black',
         dash: ''
   },
   backgroundcolor: 'transparent',
   shadow: false
}
```

backgroundcolor

This property controls the background color of the legend area.

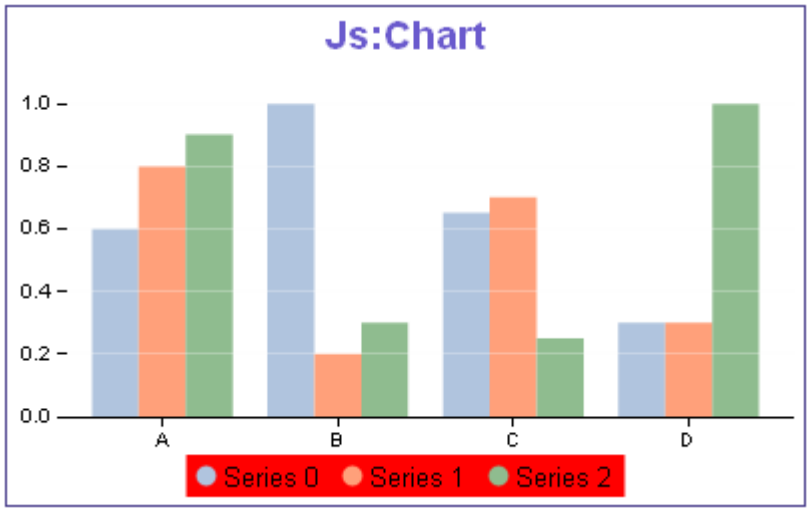
```
legend: {
   backgroundcolor: 'string' | JSON Object
}
```

PARAMETERS:

color; a color defined by a keyword or numerical specification string or a gradient defined by a string or JSON object. The default value is transparent. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string. See Gradient Definitions for the format of a linear or radial gradient.

EXAMPLE:

```
legend: {
   visible: true,
   position: 'bottom',
   backgroundcolor: 'red'
}
```



labels

This property controls the format of legend labels.

```
legend: {
   labels: {
         font: 'string',
         color: 'string'
         }
}
```

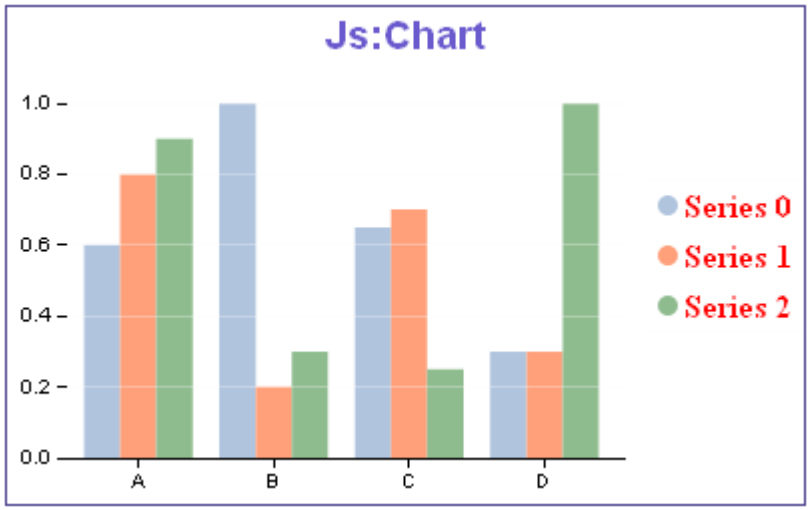
PARAMETERS:

font; a string that defines the size and type face of the Legend Labels. See http://www.w3.org/TR/CSS2/fonts.html#font-shorthand for the format of this string. The default value is '7.5pt Sans-Serif'.

color; a string that defines the color of the Legend Labels. See http://www.w3.org/TR/css3-color/#colorunits for the format of this string. The default value is 'black'.

EXAMPLE:

```
legend: {
   visible: true,
   labels: {
         font: 'Bold 12pt Times Roman',
         color: 'red'
   }
}
```



lineStyle

This property controls the width and color of a line that can be drawn around the legend area.

```
lineStyle: {
   width: number,
   color: 'string' | JSON Object,
   dash: 'string'
}
```

PARAMETERS:

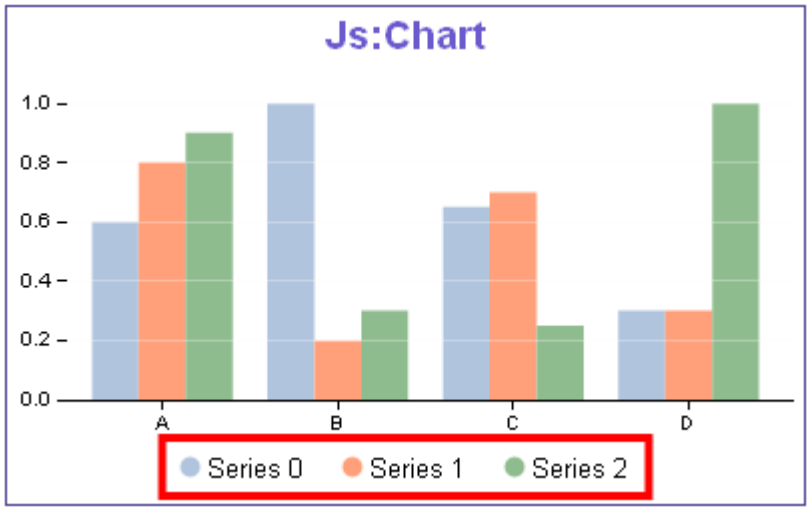
width; a number that defines the width of the line (Default = 0, no line)

color; a color defined by a keyword or numerical specification string or a gradient defined by a string or JSON object. The default value is black. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string. See Gradient Definitions for the format of a linear or radial gradient.

dash: A string that defines the border's dash style. The default value is '' (a solid line). Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes.

EXAMPLE:

```
legend: {
   visible: true,
   position: 'bottom',
   lineStyle": {'width': 4, 'color': 'red'}
}
```



markerPosition

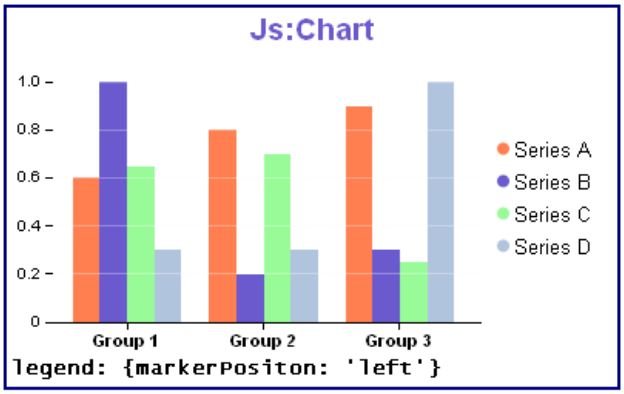
This property controls the location of the legend markers relative to the legend labels.

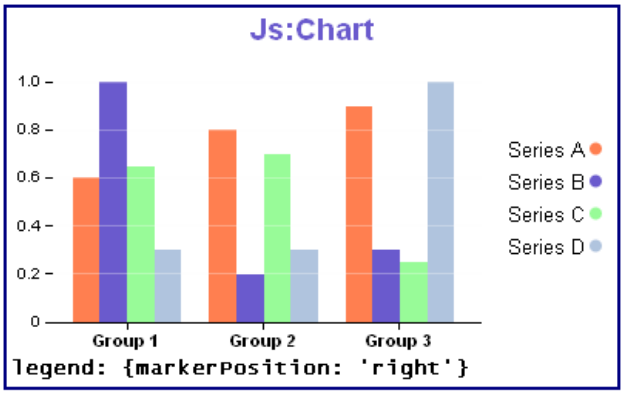
```
legend: {markerPosition: 'string'}
```

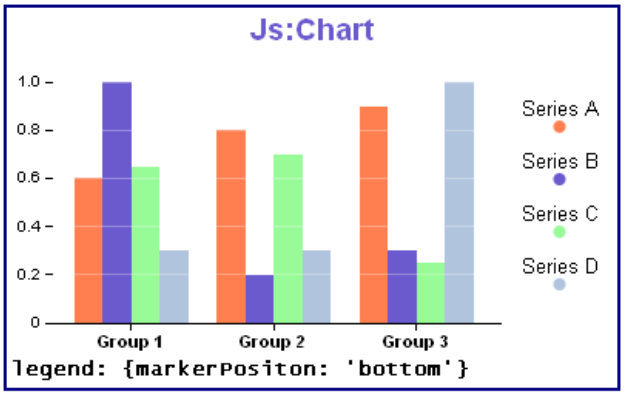
PARAMETERS:

markerPosition: a string that defines the location of legend markers relative to the legend label: 'left', 'right', or 'bottom'. The default value is 'left'.

EXAMPLE:







markerSize

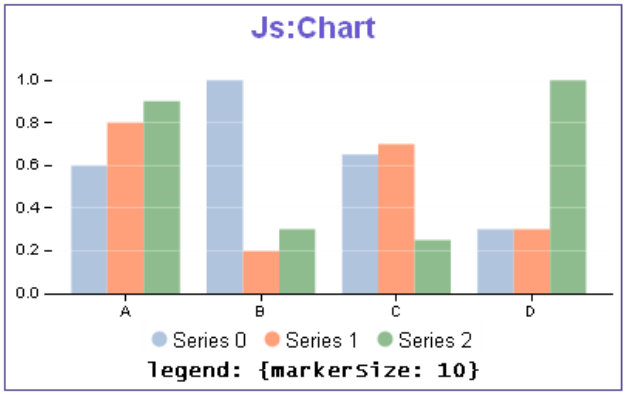
This property controls the size of legend markers.

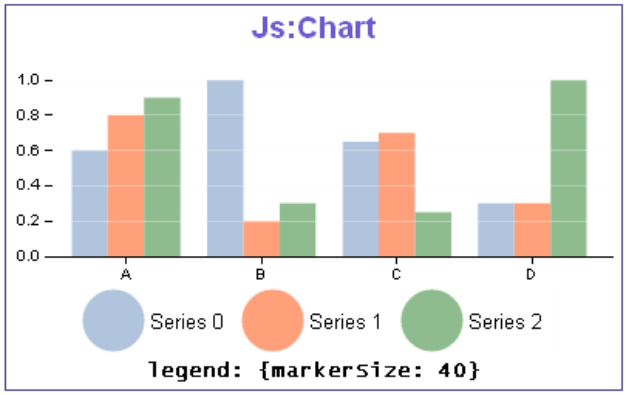
```
legend: {markerSize: Number}
```

PARAMETERS:

markerSize; defines the size of legend markers. Default value=8.

EXAMPLE:





maxEntries

This property controls the maximum number of series labels to draw in the legend area. When *maxEntries* is set to a number greater than zero and less than the number of series defined in the data set, a "more entries" indicator will be shown to indicate missing data in the legend area.

```
legend: {maxEntries: number}
```

PARAMETERS:

maxEntries: maximum number of series labels to draw in the legend. The default value is undefined.

position

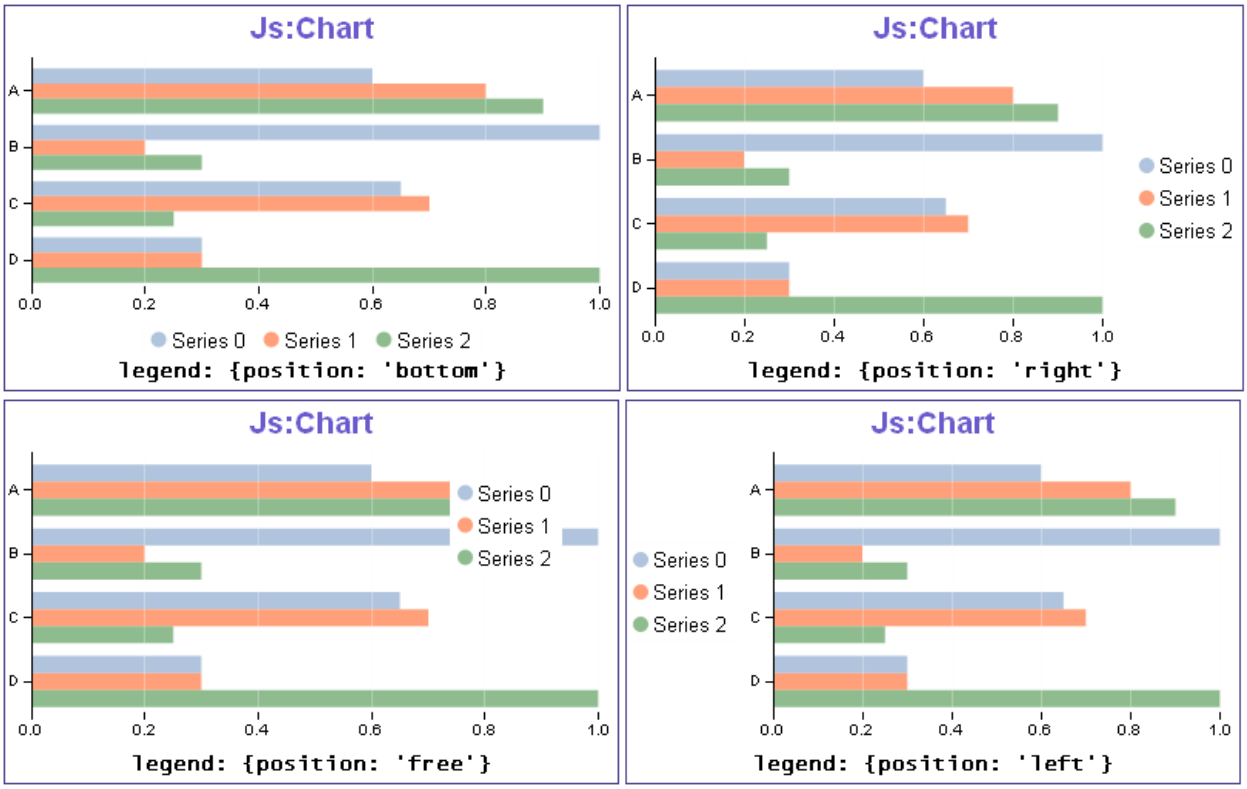
This property controls the location of the legend.

```
legend: {position: 'string'}
```

PARAMETERS:

position: a string that defines the location of the legend: 'free' (floating), 'left', 'right' (default), or 'bottom'.

EXAMPLE:



shadow

This property applies a shadow to the legend area.

```
legend: {
   shadow: 'boolean' | JSON Object
}
```

PARAMETERS:

shadow: true/false enables/disables a default shadow on the legend area. The default value is false. You can use a JSON object in the following format to define the shadow:

```
shadow:
{
   angle: Number,
   distance: Number,
   blur: Number,
}
```

shadow/angle: a number in the range 0...360 defines the position of the light source. 0 = 12 o'clock, 90 = 3 o'clock. The default value is 315.

shadow/distance: a number defines the distance to push the shadow away from parent shape, in the global coordinate space (pixels by default). The default value is 10.

shadow/blur: a number in the range 0...1 defines the hardness of the shadow. 0 = perfectly hard edge, 1 = perfectly soft edge. The default value is zero.

title

This property controls the visibility and format of the legend title.

```
legend: (
   title: {
         visible: boolean,
         text: 'string',
         font: 'string',
         color: 'string'
   }
}
```

PARAMETERS:

visible; true/false controls the visibility of the Legend Title. The default value is false.

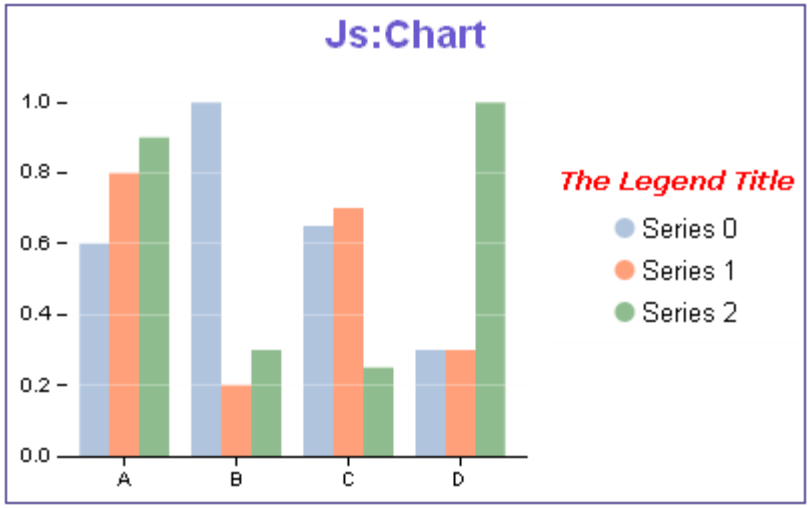
text; a string that defines the Legend Title text. The default value is 'Legend Title'.

font; a string that defines the size and type face of the Legend Title. See http://www.w3.org/TR/CSS2/fonts.html#font-shorthand for the format of this string. The default value is '10pt Sans-Serif'.

color; a string that defines the color of the Legend Title. See http://www.w3.org/TR/css3-color/#colorunits for the format of this string. The default value is 'black'.

EXAMPLE:

```
legend: {
   visible: true,
   title: {
         visible: true,
         text: 'The Legend Title',
         font: 'Bold Italic 10pt Verdana',
         color: 'red'
   }
}
```



visible

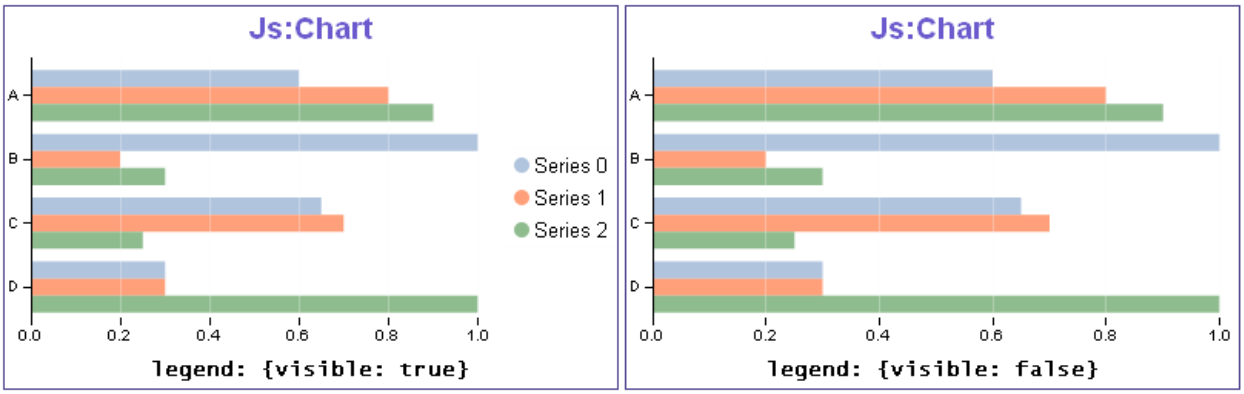
This property controls the visibility of the chart's legend area.

```
legend: {visible: boolean}
```

PARAMETERS:

visible: true = draw legend, false (default) = do not draw legend

EXAMPLE:



NOTES:

When legend position is set to "free", the legend may draw on top of other chart objects. Use the legend:xy property to define the location of the legend.

xy

When legend position is set to "free", this property controls the location of the legend.

```
legend: {xy: {x:Number, y:Number}}
```

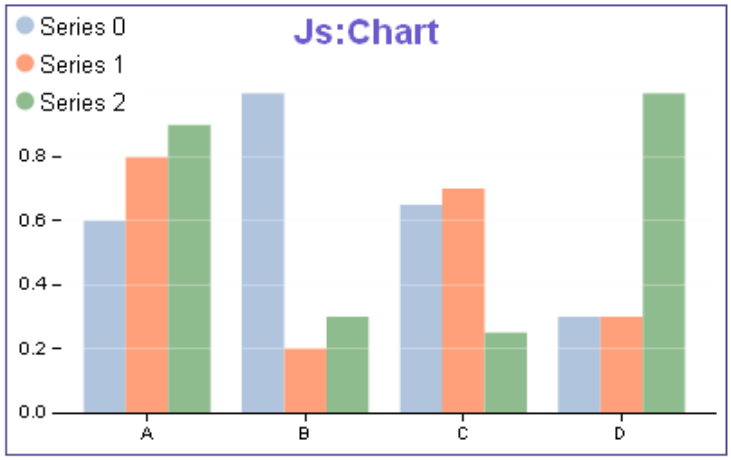
PARAMETERS:

x; Set to a value that defines the X/Horizontal location of the legend (default = 320)

y; Set to a value that defines the Y/Vertical location of the legend (default = 80)

EXAMPLE:

```
legend: {position: 'free',xy: {x:1, y:1}}
```



Ordinal Axes Properties (xaxisOrdinal / zaxisOrdinal)

These properties control the format and visibility of ordinal axes elements. The following code segment shows the default settings from properties.js.

```
xaxisOrdinal | zaxisOrdinal: {
   swapChartSide: false,
   invert: false,
   colorBand: [],
   title: {
         text: 'X Axis Title' | 'Z Axis Title',
         visible: false,
         font: '7.5pt Sans-Serif', // xaxisOrdinal
         font: '10pt Sans-Serif',  // zaxisOrdinal
         color: 'black'
   },
   labels: {
         visible: true,
         font: '7.5pt Sans-Serif',
         color: 'black',
         rotation: undefined       // xaxisOrdinal Only
   },
   bodyLineStyle: {
         width: 1,
         color: 'transparent',
         dash: ''
   },
   majorGrid: {
         visible: false,
         aboveRisers: true,
         lineStyle: {
               width: 1,
               color: 'rgba(255, 255, 255, 0.3)'
               dash: ''
         },
         ticks: {
               length: 5,
               visible: true,
               style: 'outer',
               lineStyle: {
                     width: 1,
                     color: 'black'
               }
         }
   }
   timeAxis: {
         enabled: false,
         startTime: undefined,
         stopTime: undefined,
         interval: undefined,
         stepSize: undefined,
         labelFormat: undefined
   }
},
```

Note that zaxisOrdinal only appears in 3D charts and heatmaps.

bodyLineStyle

The axis body line is perpendicular to the grid lines. It is drawn through all of the tick marks and on top of the chart frame (if visible, see the chartFrame property). These properties control the appearance of the Ordinal Axis body line.

```
xaxisOrdinal: {
   bodyLineStyle: {
         width: Number,
         color: 'string',
         dash: 'string'
   }
}
```

PARAMETERS:

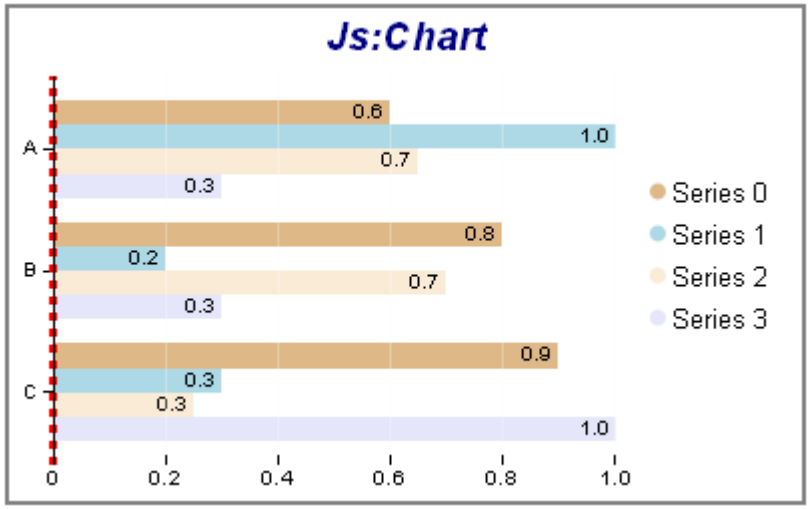
width; a number that defines the width of the line. The default value is 1.

color; a string that defines the color of the line. See http://www.w3.org/TR/css3-color/#colorunits for the format of this string. The default value is 'transparent'.

dash: a string that defines the dash style. The default value is '' (a solid line). Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes (e.g., dash: '1 1' draws a dotted line).

EXAMPLE:

```
xaxisOrdinal: {
   bodyLineStyle: {
         width: 4,
         color: 'red',
         dash: '4 4'
   }
}
```



colorBands

These properties create blocks of color that span the overall chart area

```
xaxisOrdinal | zaxisOrdinal: {
   colorBands: [
         {
               start: 'string' | number,
               stop: 'string' | number,
               color: 'string' | JSON Object
   ],
},
```

PARAMETERS:

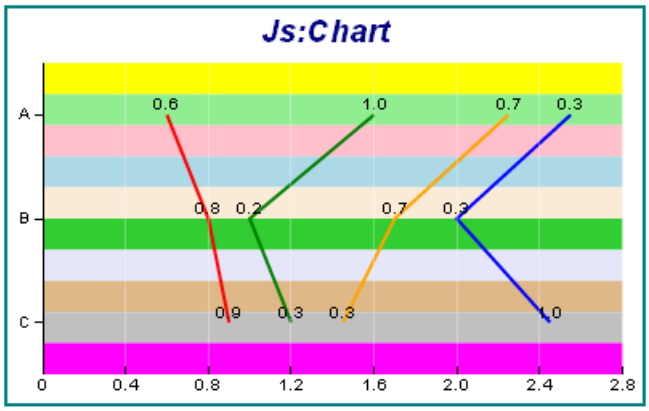
start: a number or string that defines where on the axis to start the color band. A number must be in the range 0...1 (e.g., .5 will start the color band in the center of the chart). A string must be a group label that is visible on the axis.

stop: a number or string that defines where on the axis to end the color band. A number must be in the range 0...1 (e.g., .5 will end the color band in the center of the chart). A string must be a group label that is visible on the axis.

color: a color defined by a keyword or numerical specification string or a gradient defined by a string or JSON object. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string. See Gradient Definitions for the format of a linear or radial gradient.

EXAMPLE:

```
xaxisOrdinal: {
   colorBands: [
         {start: 0,stop: .1,color: 'yellow'},
         {start: .1,stop: .2,color: 'lightgreen'},
         {start: .2,stop: .3,color: 'pink'},
         {start: .3,stop: .4,color: 'lightblue'},
         {start: .4,stop: .5,color: 'antiquewhite'},
         {start: .5,stop: .6,color: 'limegreen'},
         {start: .6,stop: .7,color: 'lavender'},
         {start: .7,stop: .8,color: 'burlywood'},
         {start: .8,stop: .9,color: 'silver'},
         {start: .9,stop: 1.0,color: 'fuchsia'},
         ],
   },
```



invert

This property controls the direction of the ordinal axis: ascending or descending.

```
xaxisOrdinal | zaxisOrdinal: {
   invert: boolean
}
```

PARAMETERS:

invert: true/false. true = draw descending axis, false = draw ascending axis. The default value is false.

labels

These properties control the visibility and format of Ordinal Axis Labels.

```
xaxisOrdinal | zaxisOrdinal: {
   labels: {
         visible: boolean,
         font: 'string',
         color: 'string'
         rotation: number
   }
}
```

PARAMETERS:

visible; true/false controls the visibility of the Ordinal Axis Labels. The default value is true.

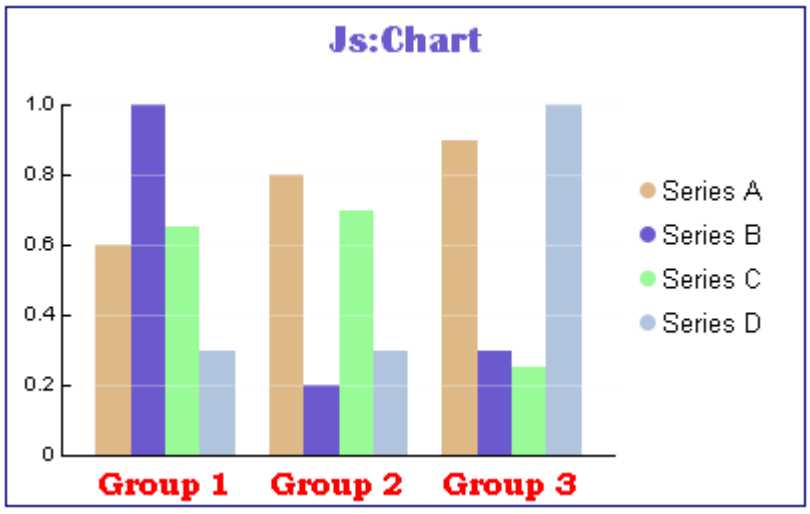
font; a string that defines the size and type face of Ordinal Axis Labels. See http://www.w3.org/TR/CSS2/fonts.html#font-shorthand for the format of this string. The default value is '7.5pt Sans-Serif'.

color; a string that defines the color of Ordinal Axis Labels. See http://www.w3.org/TR/css3-color/#colorunits for the format of this string. The default value is 'black'.

rotation: For xaxisOrdinal only in a vertical chart, defines the rotation of ordinal axis labels. 0 (none), 45 degrees, 90 degrees, 135 degrees, 180 degrees, 270 degrees, or undefined. Any value other than undefined will disable automatic layout (see chart: axisAutoLayout) of ordinal axis labels. The default value is undefined.

EXAMPLE:

```
blaProperties: {orientation: 'vertical'},
xaxisOrdinal: {
   labels: {visible: true,font: 'bold 12pt Bookman Old Style',
         color: 'red'}}
```



majorGrid

These properties control the visibility and appearance of major grid lines on the Ordinal Axis.

```
xaxisOrdinal: {
   majorGrid: {
         visible: boolean,
         aboveRisers: boolean,
         lineStyle: {
               width: Number,
               color: 'string',
               dash: 'string'
         }
   }
}
```

PARAMETERS:

visible: true = draw major grid lines/false = hide major grid lines. The default value is false.

aboveRisers: true = draw major grid lines above/on top of risers. false = draw major grid lines behind risers. The default value is true.

lineStyle: Defines the width and color of major grid lines.

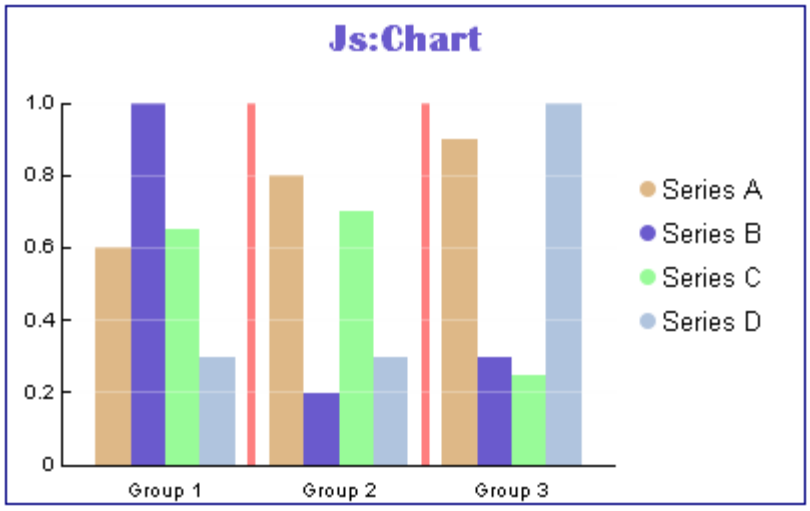
lineStyle/width: a number that defines the width of the line. The default value is 1.

lineStyle/color: a string that defines the color of the line. See http://www.w3.org/TR/css3-color/#colorunits for the format of this string. The default value is 'rgba(255, 255, 255, 0.3)'.

lineStyle/dash: a string that defines the dash style. The default value is '' (a solid line). Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes (e.g., dash: '1 1' draws a dotted line).

EXAMPLE:

```
xaxisOrdinal: {
   majorGrid: {visible: true, aboveRisers: true,
         lineStyle: {width: 4,color: 'red'}}
}
```



majorGrid: ticks

These properties control the visibility, style and format of Ordinal Axis tick marks.

```
xaxisOrdinal: {
   majorGrid: {
         ticks: {
               length: Number,
               visible: boolean,
               style: 'string',
               lineStyle: {width: Number, color: 'string'}
         }
   }
}
```

PARAMETERS:

length: a number that defines the length of tick marks. The default value is 5.

visible: true = draw tick marks/false = hide tick marks. The default value is true.

style: a string that defines the tick mark style: 'inner', 'outer', or 'span'. The default value is 'outer'.

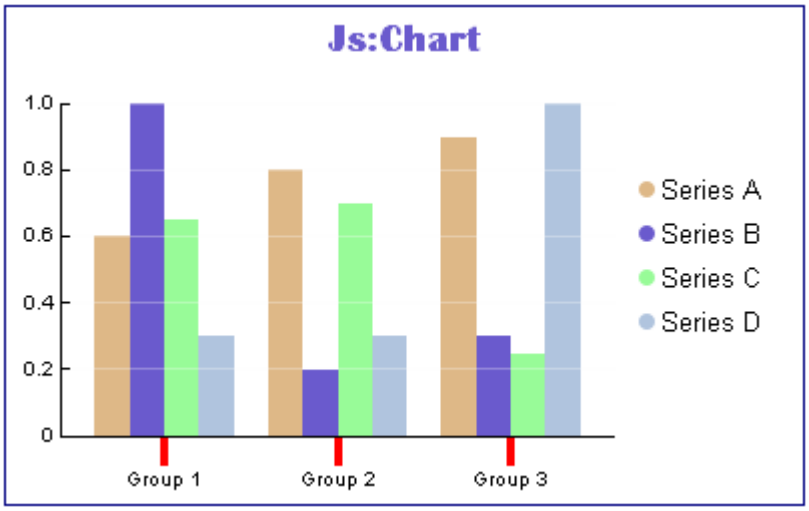
lineStyle: Defines the width and color of tick marks.

lineStyle/width: a number that defines the width of the tick marks. The default value is 1.

lineStyle/color: a string that defines the tick mark color. See http://www.w3.org/TR/css3-color/#colorunits for the format of this string. The default value is 'black'.

EXAMPLE:

```
xaxisOrdinal: {
   majorGrid: {
         ticks: {    length: 15,visible: true,style: 'outer',
                     lineStyle: {width: 4, color: 'red'}
               }
         }
}
```



swapChartSide

This property controls the location of the ordinal axis body line and labels. In the default configuration, the ordinal axis body line and labels appear on the left side of a horizontal Bar/Line/Area chart and the bottom side of a vertical Bar/Line/Area chart. This property can be used to reverse these default locations.

```
xaxisOrdinal | zaxisOrdinal: {
   swapChartSide: boolean
}
```

PARAMETERS:

swapChartSide: true = swap labels/body line locations. false = use the default axis locations. The default value is false.

timeAxis

These properties create data/time labels on the ordinal axis. The primary purpose of a time axis is to auto-generate group labels in units of time based on start/stop times and an optional interval. The library also supports 'sparse' data (i.e., the number of data values that define the chart do not match the number of group labels defined by the time axis). For example, the start/stop times are January/December (12 months) but the data set only includes six values. You can configure the time axis properties to show all 12 month/group labels with the six risers at given months.

```
xaxisOrdinal: {
timeAxis: {
   enabled: boolean,
   startTime: 'string',
   stopTime: 'string',
   interval: 'string',
   stepSize: Number,
   labelFormat: 'string'
   }
}
```

PARAMETERS:

enabled: true/false enables/disables the time axis. The default value is false (disabled).

startTime: a string identifying the start time. It can be almost any kind of string denoting time, mixing hours (xx:xx), days of the week, dates, months and years. Examples: "Mon", "1 July 20", "June 2001", "8:00", "6/10/01", "Thu, 4 Apr 1976 14:30", "2 0:00" (February 1), "1 1 0:00" (January 1). The default value is undefined.

stopTime: a string identifying the start time. As with startTime, the string can be almost any date/time format. The default value is undefined.

interval: a string that specifies the time interval: 'seconds', 'minutes', 'hours', 'days', 'weeks', 'months', 'quarters', or 'years'. This property defines the time gap between each group label. If undefined, the library will assign an interval based on the startTime, stopTime, and the number of groups. The default value is undefined.

stepSize: a number identifying how many group labels to skip between each label. If undefined (or 0), it's ignored and all labels are drawn. The default value is undefined.

labelFormat: a string that defines how a time/date number is converted into a label. If undefined, the library will we use a default formats based on the interval. See http://code.google.com/p/datejs/wiki/FormatSpecifiers for available format options. The default value is undefined.

NOTES:

You can also create a time axis by setting xaxisOrdinal: timeAxis: enabled: true and defining start and stop times in group labels. Example:

```
groupLabels: ["Monday", "Friday"]
```

If timeAxis: startTime is undefined and the first groupLabel is a date/time, it will be used as the start time. If timeAxis: stopTime is undefined and the last groupLabel is a date/time, it will be used as the stop time. However, timeAxis: startTime and stopTime properties take precedence over group labels

If timeAxis: enabled: true and all of group labels are dates, the library will plot the corresponding data points according to each group label date -- not according to their passed in order. This allows you to specify both sparse and unordered data.

title

These properties control the content, visibility, and format of an ordinal axis title. The zaxisOrdinal title is only available in a heatmap chart. The xaxisOrdinal title is not included in a radar chart. The zaxisOrdinal titles is not available in 3D charts.

```
xaxisOrdinal | zaxisOrdinal: {
   title: {
         text: 'string',
         visible: boolean,
         font: 'string',
         color: 'string'
   }
}
```

PARAMETERS:

text; a string that defines Ordinal Axis Title text. The default value for xaxisOrdinal is 'X Axis Title'. The default value for zaxisOrdinal is 'Z Axis Title'.

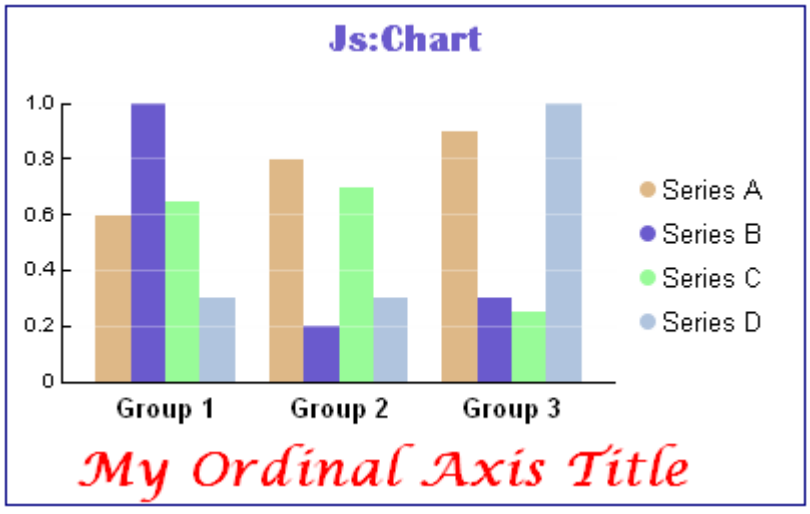
visible; true/false controls the visibility of the Ordinal Axis Title. The default value is false.

font; a string that defines the size and type face of the Ordinal Axis Title. See http://www.w3.org/TR/CSS2/fonts.html#font-shorthand for the format of this string. The default value for xaxisOrdinal is '7.5pt Sans-Serif'. The default value for zaxisOrdinal is '10pt Sans-Serif'.

color; a string that defines the color of the Ordinal Axis Title. See http://www.w3.org/TR/css3-color/#colorunits for the format of this string. The default value is 'black'.

EXAMPLE:

```
blaProperties: {orientation: 'vertical'},
xaxisOrdinal: {
   title: {
         text: 'My Ordinal Axis Title',
         visible: true,
         font: 'bold 18pt Lucida Calligraphy',
         color: 'red'
   }
}
```



Numeric Axes Properties (xaxisNumeric/yaxis/y2axis)

These properties control the appearance of the numeric axes. The numeric X-Axis is only present in Bubble, Polar, and Scatter charts. The Y-Axis appears in all chart types except Pie and Funnel Charts. The Y2-Axis is present in pareto charts and bar/line/area charts when a series is assigned to the axis with the yAxisAssignment property. This code segment shows the default settings from properties.js.

```
xaxisNumeric | yaxis: | y2axis: {
   min: undefined,
   max: undefined,
   intervalMode: undefined,
   intervalValue: undefined,
   numberFormat: '[>9]#,#;[<0]-#.#;[<=9]#.#;[=0]0',
   swapChartSide: false, // xaxisNumeric and yaxis ONLY
   invert: false,
   altFrameColor: undefined,
   colorBands: [],
   title: {
         text: 'X | Y | Y2 Axis Title',
         visible: false,
         font: '7.5pt Sans-Serif',
         color: 'black'
   },
   labels: {
         visible: true,
         font: '7.5pt Sans-Serif',
         color: 'black'
   },
   baseLineStyle: {
         width: 1,
         color: 'black',
         dash: '',
   },
   bodyLineStyle: {
         width: 1,
         color: 'transparent',
         dash: '',
   },
   majorGrid: {
         visible: true,
         aboveRisers: true,
         lineStyle: {
               width: 1,
               color: 'rgba(255, 255, 255, 0.3)',
               dash: ''
         },
         ticks: {
               length: 5,
               visible: true,
               style: 'inner',
               lineStyle: {width: 1,color: 'black'},
         },
   },
   minorGrid: {
         visible: false,
         count: undefined,
         lineStyle: {
               width: 1,
               color: 'black',
```

```
                dash: ''
        },
        ticks: {
                length: 5,
                visible: false,
                style: 'inner',
                lineStyle: {width: 1,color: 'black'}
        }
   }
},
```

altFrameColor

This property fills every other segment of a chart's numeric axis grid with a specified color.

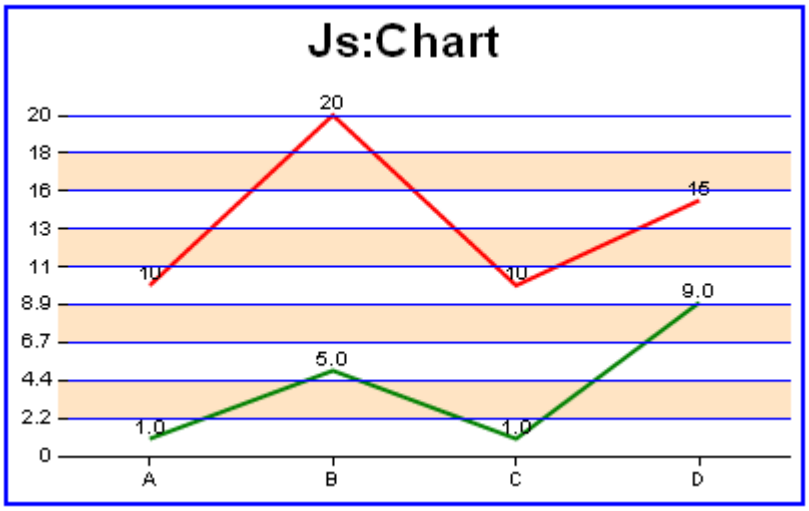
```
xaxisNumeric | yaxis | y2axis: {
   altFrameColor: 'string' | JSON Object
},
```

PARAMETERS:

color: a color defined by a keyword or numerical specification string or a gradient defined by a string or JSON object. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string. See Gradient Definitions for the format of a linear or radial gradient.

EXAMPLES:

```
yaxis: {
   intervalMode: 'count',intervalValue: 10,
   majorGrid: {lineStyle: {width: 1,color: 'blue'}},
   altFrameColor: 'bisque'},
```



baseLineStyle

The axis base line is parallel to the grid lines and normally passes through the zero value. These properties control the appearance of a numeric axis base line.

```
xaxisNumeric: | yaxis: | y2axis: {
   baseLineStyle: {
         width: Number,
         color: 'string'
         dash: 'string'
   }
}
```

PARAMETERS:

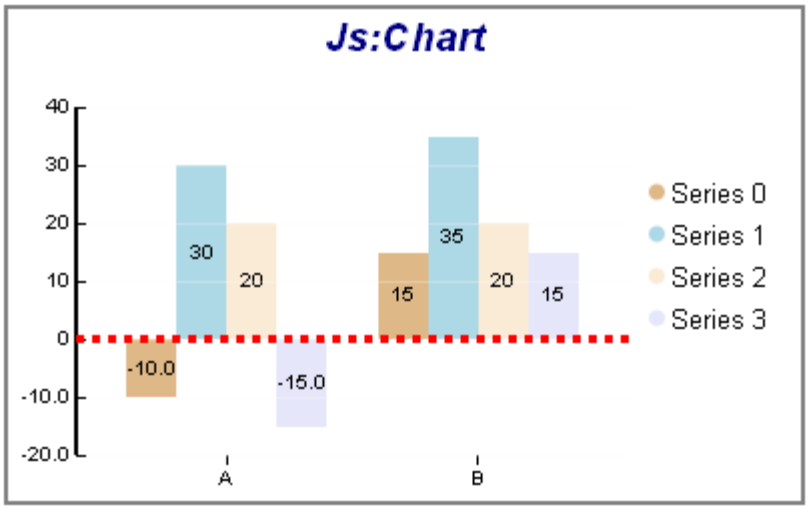
width; a number that defines the width of the line. The default value is 1.

color; a string that defines the color of the line. See http://www.w3.org/TR/css3-color/#colorunits for the format of this string. The default value is 'black'.

dash: a string that defines the dash style. The default value is '' (a solid line). Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes (e.g., dash: '1 1' draws a dotted line).

EXAMPLES:

```
blaProperties: {orientation: 'vertical'},
yaxis: {
   bodyLineStyle: {width: 2, color: 'black'},
   baseLineStyle: {width: 4, color: 'red', dash: '4 4'},
}
```



bodyLineStyle

The axis body line is perpendicular to the grid lines. It is drawn through all of the tick marks and on top of the chart frame (if visible, see the chartFrame property). These properties control the appearance of a numeric axis body line.

```
xaxisNumeric: | yaxis: | y2axis: {
   bodyLineStyle: {
         width: Number,
         color: 'string'
         dash: 'string'
   }
}
```

PARAMETERS:

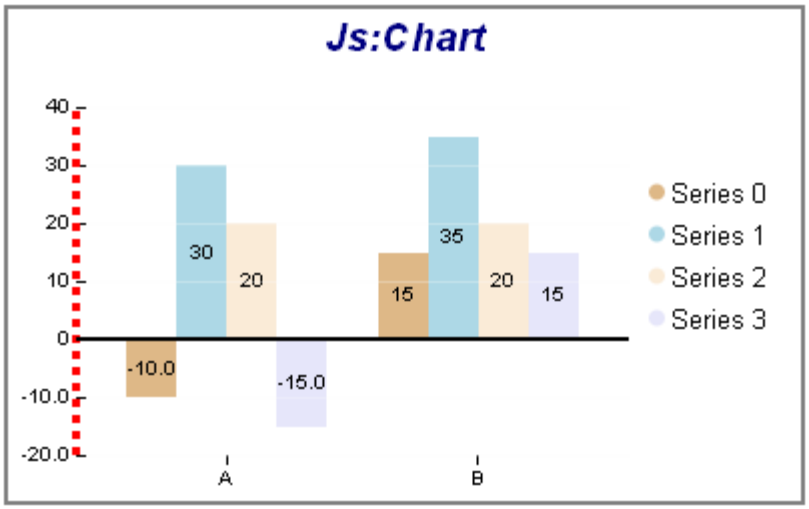
width; a number that defines the width of the line. The default value is 1.

color; a string that defines the color of the line. See http://www.w3.org/TR/css3-color/#colorunits for the format of this string. The default value is 'transparent'.

dash: a string that defines the dash style. The default value is '' (a solid line). Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes (e.g., dash: '1 1' draws a dotted line).

EXAMPLES:

```
blaProperties: {orientation: 'vertical'},
yaxis: {
   bodyLineStyle: {width: 4, color: 'red', dash: '4 4'},
   baseLineStyle: {width: 2, color: 'black'},
}
```



colorBands

These properties create blocks of color that span the overall chart area. For a gauge chart, yaxis:colorBands control the color of gauge band segments.

```
xaxisNumeric | yaxis | y2axis: {
   colorBands: [
         {
               start: 'string' | number,
               stop: 'string' | number,
               color: 'string' | JSON Object
   ],
},
```

PARAMETERS:

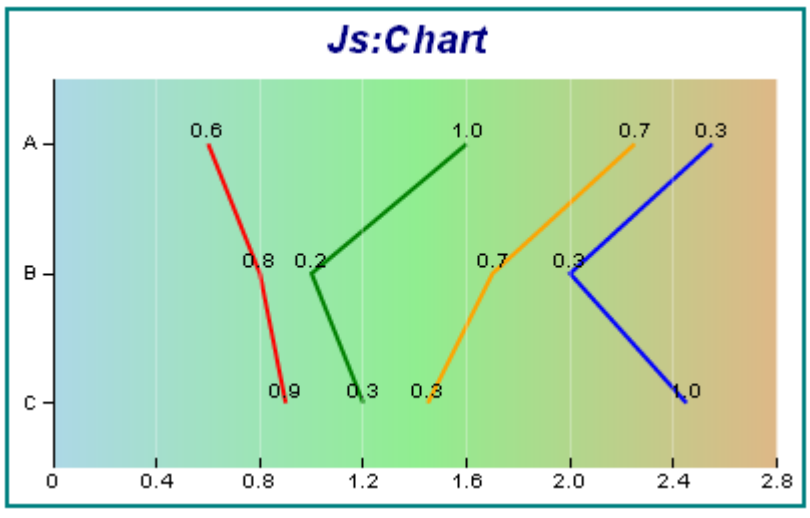
start: a number or string that defines where on the axis to start the color band. A number must be a value that appears on the axis. A string must represent a position percentage value (i.e., '0%' ... '100%').

stop: a number or string that defines where on the axis to end the color band. A number must be a value that appears on the axis. A string must represent a position percentage value (i.e., '0%' ... '100%').

color: a color defined by a keyword or numerical specification string or a gradient defined by a string or JSON object. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string. See Gradient Definitions for the format of a linear or radial gradient.

EXAMPLES:

```
yaxis: {
   colorBands: [
   {
         start: 0,
         stop: '100%',
         color: 'linear-gradient(0%,0,100%,0, 0 lightblue, 0.5
lightgreen, 1 burlywood)'
         },
   ],
},
```



intervalMode/Value

These properties control the number of major grid lines, ticks, and labels on a numeric axis.

```
xaxisNumeric: | yaxis: | y2axis: {
   intervalMode: 'string' | undefined
   intervalValue: Number | undefined
}
```

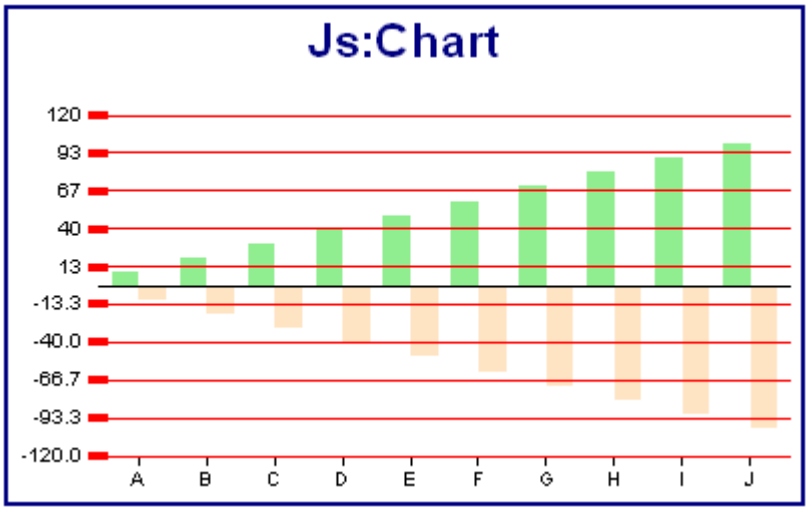
PARAMETERS:

intervalMode; a string that defines the interval mode: 'count', 'interval', or 'skip'. Set to undefined to automatically calculate the number and frequency of major grid lines. The default value is undefined.

intervalValue; if intervalMode is 'count', set this property to the number of major grid lines to draw. If intervalMode is 'interval', set this property to the interval at which major grid lines are drawn. If intervalMode is 'skip', set this property to the number of grid lines to skip. Set to undefined to automatically calculate the number and frequency of major grid lines. The default value is undefined.

EXAMPLE:

```
yaxis: {
   intervalMode: 'count', intervalValue: 10,
},
```



NOTES:

The first and last grid lines are always drawn regardless of the intervalMode or intervalValue.

invert

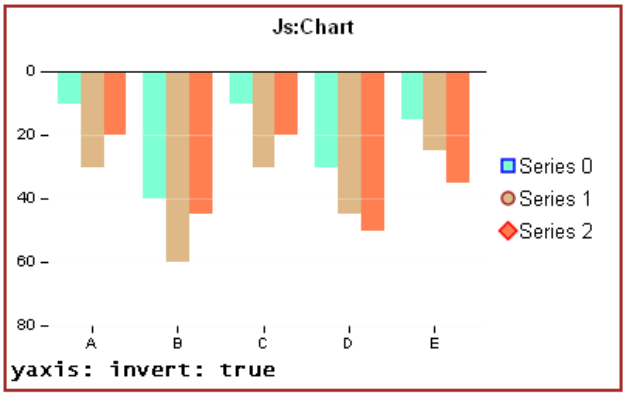
This property controls the direction of a numeric axis: ascending or descending.

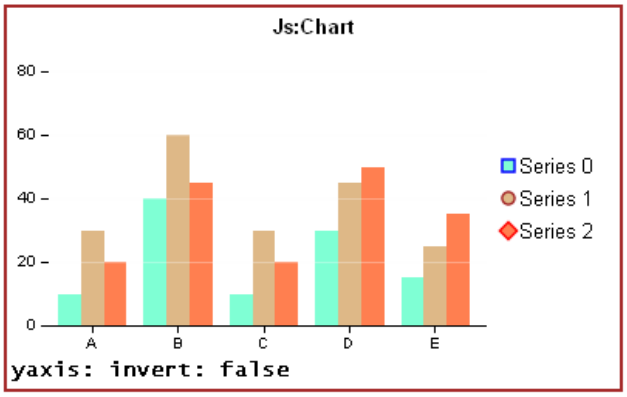
```
xaxisNumeric: | yaxis: | y2axis: {
   invert: boolean
}
```

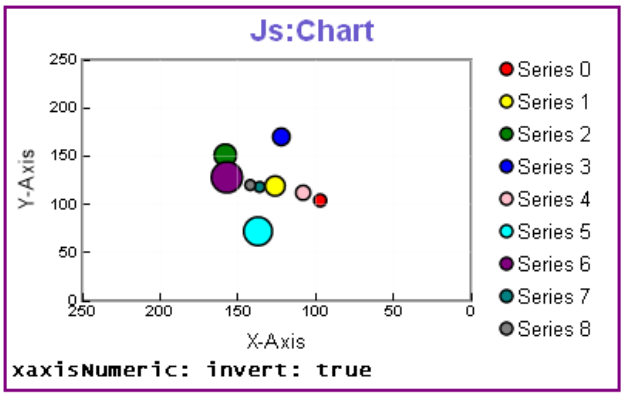
PARAMETERS:

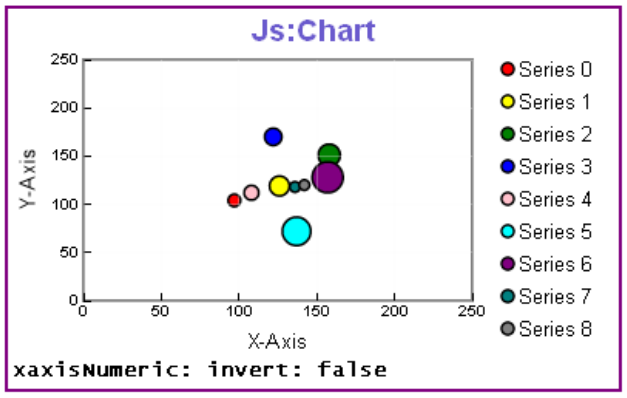
invert: true/false. true = draw descending axis, false = draw ascending axis. The default value is false.

EXAMPLES:









labels

These properties control the visibility and format of Numeric Axis Labels.

```
xaxisNumeric: | yaxis: | y2axis: {
   labels: {
         visible: boolean,
         font: 'string',
         color: 'string'
   }
}
```

PARAMETERS:

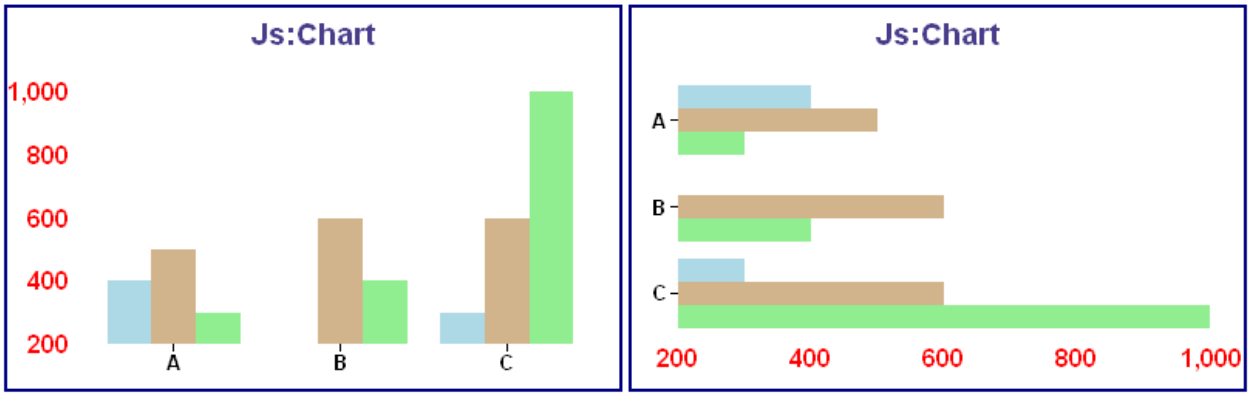
visible; true/false controls the visibility of the Numeric Axis labels. The default value is true.

font; a string that defines the size and type face of Numeric Axis labels. See http://www.w3.org/TR/CSS2/fonts.html#font-shorthand for the format of this string. The default value is '7.5pt Sans-Serif'.

color; a string that defines the color of Numeric Axis labels. See http://www.w3.org/TR/css3-color/#colorunits for the format of this string. The default value is 'black'.

EXAMPLES:

```
yaxis: {
   labels: {
         visible: true,
         font: 'bold 12pt Sans-Serif',
         color: 'red'
   }
}
```



majorGrid

These properties control the visibility and appearance of major grid lines on a numeric axis.

```
xaxisNumeric: | yaxis: | y2axis: {
   majorGrid: {
         visible: boolean,
         aboveRisers: boolean,
         lineStyle: {
               width: Number,
               color: 'string',
               dash: 'string'
         }
   }
}
```

PARAMETERS:

visible: true = draw major grid lines/false = hide major grid lines. The default value is true.

aboveRisers: true = draw major grid lines above/on top of risers. false = draw major grid lines behind risers. The default value is true.

lineStyle: Defines the width, color, and dash style of major grid lines.

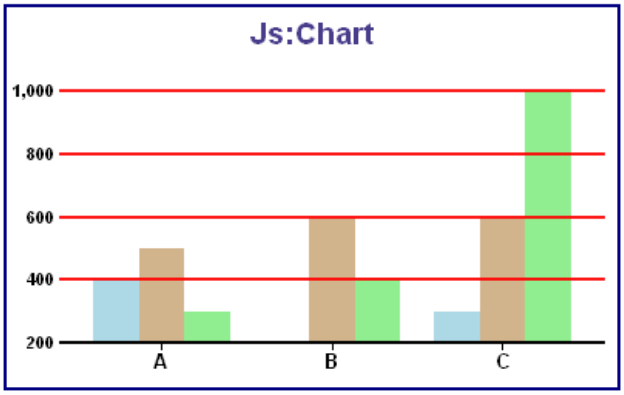
lineStyle/width: a number that defines the width of the line. The default value is 1.

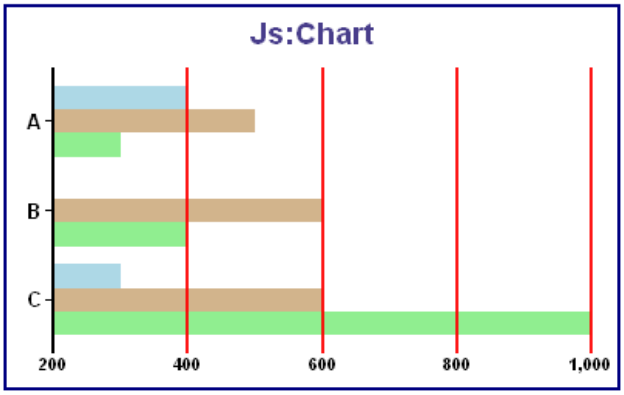
lineStyle/color: a string that defines the color of the line. See http://www.w3.org/TR/css3-color/#colorunits for the format of this string. The default value is 'rgba(255, 255, 255, 0.3)'.

lineStyle/dash: a string that defines the dash style. The default value is '' (a solid line). Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes (e.g., dash: '1 1' draws a dotted line).

EXAMPLES:

```
yaxis: {
   majorGrid: {
         visible: true,
         aboveRisers: true,
         lineStyle: {width: 4, color: 'red'}
   }
}
```





majorGrid: ticks

These properties control the visibility, style and format of major tick marks on a numeric axis.

```
xaxisNumeric: | yaxis: | y2axis: {
   majorGrid: {
         ticks: {
               length: Number,
               visible: boolean,
               style: 'string',
               lineStyle: {
                     width: Number,
                     color: 'string'
               }
         }
   }
}
```

PARAMETERS:

length: a number that defines the length of tick marks. The default value is 5.

visible: true = draw tick marks/false = hide tick marks. The default value is true.

style: a string that defines the tick mark style: 'inner', 'outer', or 'span'. The default value is 'inner'.

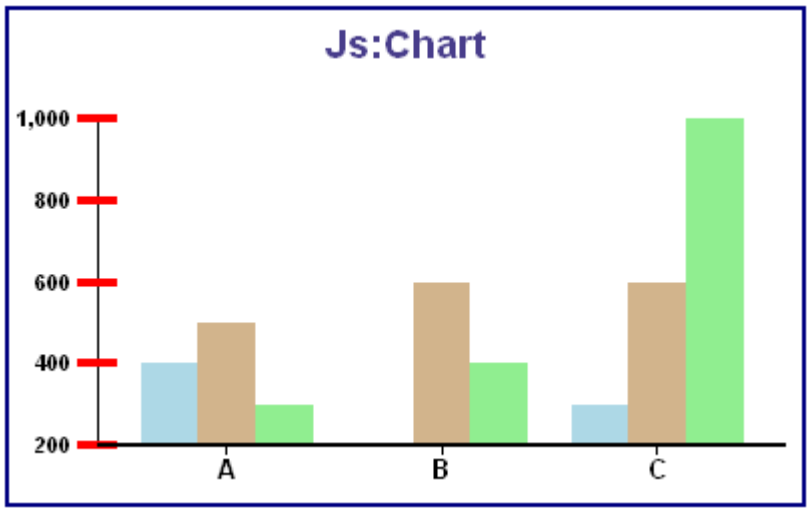
lineStyle: Defines the width and color of tick marks.

lineStyle/width: a number that defines the width of the tick marks. The default value is 1.

lineStyle/color: a string that defines the tick mark color. See http://www.w3.org/TR/css3-color/#colorunits for the format of this string. The default value is 'black'.

EXAMPLES:

```
yaxis: { majorGrid: {
         ticks: {    length: 10, visible: true,style: 'span',
                     lineStyle: {width: 4, color: 'red'}
   }     }
}
```



min/max

These properties define the minimum and maximum values to draw on a Numeric Axis.

```
xaxisNumeric: | yaxis: | y2axis: {
   min: Number
   max: Number
}
```

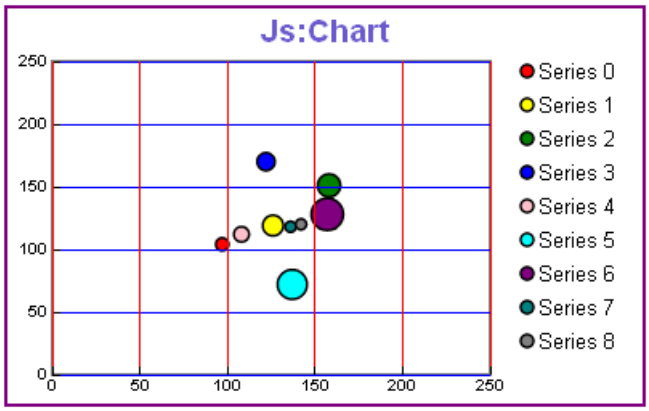
PARAMETERS:

min; minimum value to draw on a Numeric Axis. Set to undefined to automatically calculate the minimum value based on values in the data set. The default value is undefined.

max; maximum value to draw on a Numeric Axis. Set to undefined to automatically calculate the maximum value based on values in the data set. The default value is undefined.

EXAMPLE:

```
chartType: 'bubble',
yaxis: {
   min: 0,max: 250,
   majorGrid: {lineStyle: {width: 1, color: 'blue'}},
   },
xaxisNumeric: {
   min: 0,max: 250,
   majorGrid: {lineStyle: {width: 1, color: 'red'}},
   },
```



minorGrid

These properties control the visibility and appearance of minor grid lines on a numeric axis.

```
xaxisNumeric: | yaxis: | y2axis: {
   minorGrid: {
         visible: boolean,
         count: Number,
         lineStyle: {
               width: Number,
               color: 'string',
               dash: 'string'
         }
   }
}
```

PARAMETERS:

visible: true = draw minor grid lines/false = hide minor grid lines. The default value is false.

count: number of minor grid lines to draw between each major grid line. The default value is undefined.

lineStyle: Defines the width, color, and dash style of minor grid lines.

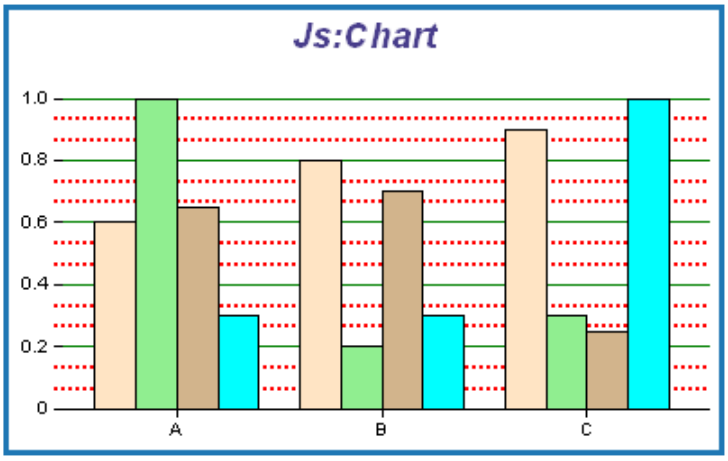
lineStyle/width: a number that defines the width of the line. The default value is 1.

lineStyle/color: a string that defines the color of the line. See http://www.w3.org/TR/css3-color/#colorunits for the format of this string. The default value is 'black'.

lineStyle/dash: a string that defines the dash style. The default value is '' (a solid line). Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes (e.g., dash: '1 1' draws a dotted line).

EXAMPLES:

```
yaxis: {
   minorGrid: {visible: true,count: 2,
         lineStyle: {width: 2,color: 'red',dash: '2 2'}
   }
},
```



minorGrid: ticks

These properties control the visibility, style and format of minor tick marks on a numeric axis.

```
xaxisNumeric: | yaxis: | y2axis: {
   minorGrid: {
         ticks: {
               length: Number,
               visible: boolean,
               style: 'string',
               lineStyle: {width: Number,color: 'string'}
         }
   }
}
```

PARAMETERS:

length: a number that defines the length of tick marks. The default value is 5.

visible: true = draw tick marks/false = hide tick marks. The default value is false.

style: a string that defines the tick mark style: 'inner', 'outer', or 'span'. The default value is 'inner'.

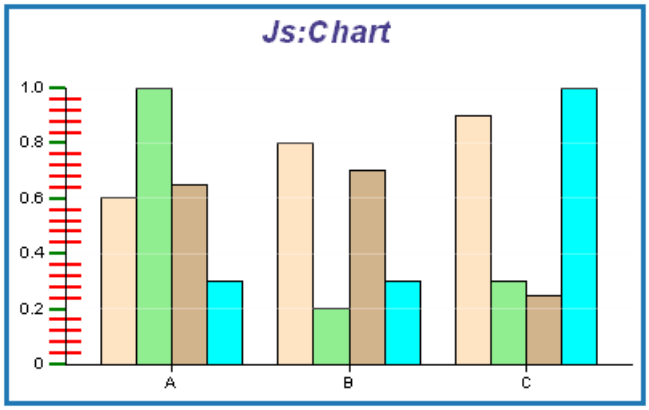
lineStyle: Defines the width and color of tick marks.

lineStyle/width: a number that defines the width of the tick marks. The default value is 1.

lineStyle/color: a string that defines the tick mark color. See http://www.w3.org/TR/css3-color/#colorunits for the format of this string. The default value is 'black'.

EXAMPLES:

```
yaxis: {
   majorGrid: {
         ticks: {length: 10,visible: true,style: 'outer',
               lineStyle: {width: 2,color: 'green'}}
   },
   minorGrid: {
         ticks: {length: 10,visible: true,style: 'span',
               lineStyle: {width: 2,color: 'red'}}
   }
},
```



numberFormat

This property defines the format of Numeric Axis labels.

```
xaxisNumeric: | yaxis: | y2axis: {
   numberFormat: JSON Object | 'string' | function()
}
```

PARAMETERS:

numberFormat: Use a string, JSON object, or function to define the format of numeric axis labels. See Properties Overview/Formatting Numbers in Section 1 for details and examples. The default value is '[>9]#,#;[<0]-#.#;[<=9]#.#;[=0]0'.

swapChartSide

This property controls the location of a numeric axis body line and labels. In the default configuration, the yaxis body line and labels appear on the bottom of a horizontal Bar/Line/Area chart and the left side of a vertical Bar/Line/Area chart. The Y2-Axis body line and labels appear on the top of a horizontal Bar/Line/Area chart and the right side of a vertical Bar/Line/Area chart. In Bubble and Scatter charts, the X-Axis normally appears on the bottom of the chart and the Y-Axis on the left side of the chart. This property can be used to reverse these default locations.

```
xaxisNumeric: | yaxis: {
   swapChartSide: boolean
}
```

PARAMETERS:

swapChartSide: true = swap labels/body line locations. false = use the default axis locations. The default value is false.

title

These properties control the content, visibility, format of numeric axis titles. The xaxisNumeric title is not available in polar charts.

```
xaxisNumeric: | yaxis: | y2axis: {
   title: {
         text: 'string',
         visible: boolean,
         font: 'string',
         color: 'string'
   },
}
```

PARAMETERS:

text; a string that defines Numeric Axis title text. The default value for xaxisNumeric is 'X Axis Title'. The default value for yaxis is 'Y Axis Title'. The default value for y2axis is 'Y2 Axis Title'.

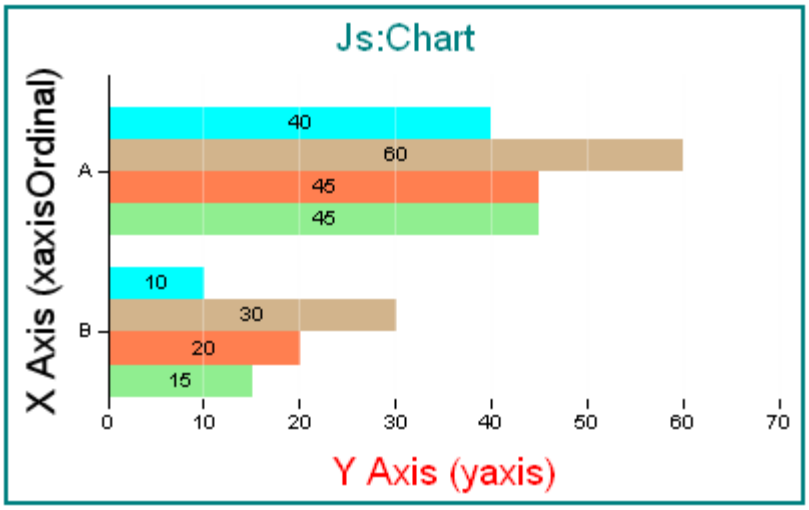
visible; true/false controls the visibility of the Numeric Axis title. The default value is false.

font; a string that defines the size and type face of the Numeric Axis title. See http://www.w3.org/TR/CSS2/fonts.html#font-shorthand for the format of this string. The default value is '7.5pt Sans-Serif'.

color; a string that defines the color of the Numeric Axis title. See http://www.w3.org/TR/css3-color/#colorunits for the format of this string. The default value is 'black'.

EXAMPLES:

```
xaxisOrdinal:{title: {
   visible: true, text: 'X Axis (xaxisOrdinal)',
   font: '14pt Sans-Serif', color: 'black'}},
yaxis:{title: {
         visible: true, text: 'Y Axis (yaxis)',
         font: '14pt Sans-Serif',
         color: 'red'}
         },
```



Series-Specific Properties

These properties can be used to control the appearance of individual series:

```
series: [
   series: Number,
   group: Number,
   label: 'string',
   color: 'string' | JSON Object,
   riserShape: 'string'
   border: {
         width: Number,
         color: 'string',
         dash: 'string'
   },
   marker: {
         visible: boolean,  // Line Charts Only
         color: 'string',
         size: Number,
         shape: 'string',
         rotation: Number,
         position: 'string'
         border: {
               width: Number,
               color: 'string',
               dash: 'string'
         }
   },
   showDataValues: boolean,
   explodeSlice: Number,
   deleteSlice: boolean,
   yAxisAssignment: Number,
   tooltip: 'string' | function()
]
```

The following code segment shows the default settings from properties.js:

```
series: [
   {series: 'all', color: 'blue', showDataValues: true, border:
{width: 2}, marker: {size: 8, border: {width: 1, color: 'black'}}},
   {series: 0, color: 'red'},
   {series: 1, color: 'green'},
   {series: 2, color: 'orange'}
]
```

border

These properties define a border for all risers, for all risers in an individual series, or for an individual riser identified by a series and group number. It can be used for area risers, bar risers, funnel segments, gauge needles, and pie slices.

```
series: [
   {
         series: Number,
         group: Number, // optional
         border: {
               width: Number,
               color: 'string',
               dash: 'string'
         },
   }
]
```

PARAMETERS:

series: zero-based series number. If the series does not exist in the chart, the property is ignored.

group: (optional) zero-based group number. If the group does not exist in the chart, the property is ignored. If a group number is not specified, the border is applied to all risers in the series.

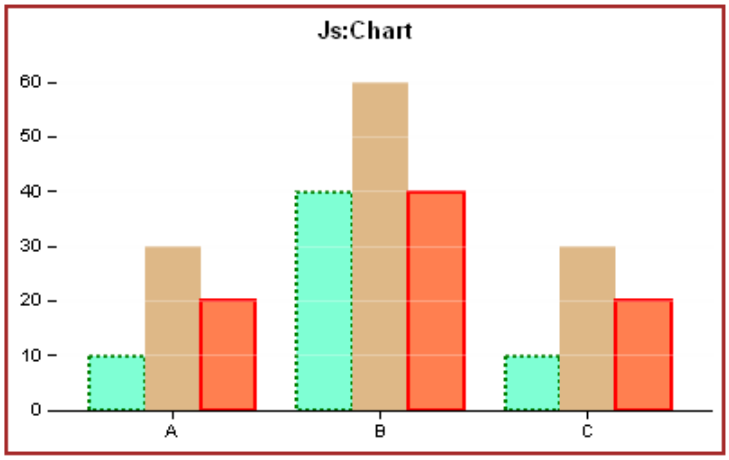
width: a number that defines the width of the border.

color: a color defined by a keyword or numerical specification string. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string.

dash: a string that defines the border dash style. Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes.

EXAMPLES:

```
blaProperties: {orientation: 'vertical'}
series: [
   {series: 0, color: 'aquamarine', border: {width: 2, color:
'green', dash: '2 2'}},
   {series: 1, color: 'burlywood'},
   {series: 2, color: 'coral', border: {width: 2, color: 'red'}},
   ]
```



color

These properties define a color for all risers, for all risers in an individual series, or for an individual riser identified by a series and group number. If a group number is not specified, the color is also applied to the series icon in the legend area.

```
series: [
   {
         series: Number,
         group: Number, // optional
         color: 'string' | JSON Object
   },
]
```

PARAMETERS:

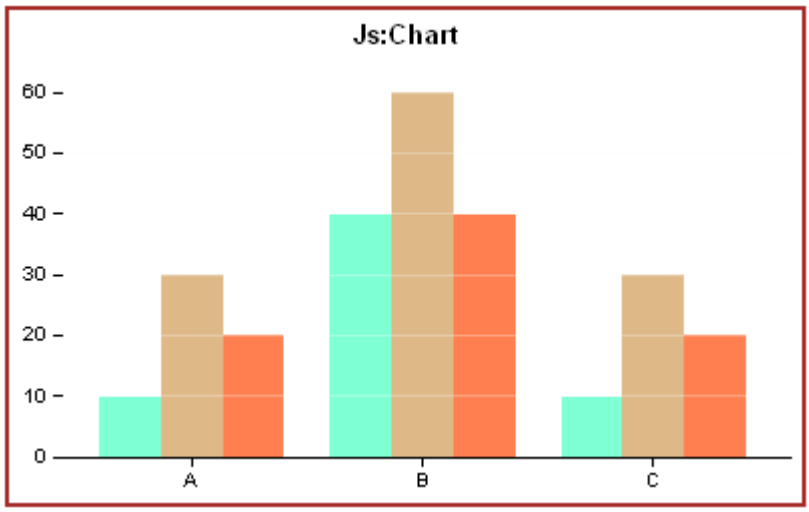
series: zero-based series number. If the series does not exist in the chart, the property is ignored.

group: optional zero-based group number. If the group does not exist in the chart, the property is ignored. If a group number is not specified, the color is applied to all risers in the series.

color: a color defined by a keyword or numerical specification string or a gradient defined by a string or JSON object. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string. See Gradient Definitions for the format of a linear or radial gradient.

EXAMPLES:

```
blaProperties: {orientation: 'vertical'}
series: [
   {series: 0, color: 'aquamarine'},
   {series: 1, color: 'burlywood'},
   {series: 2, color: 'coral'},
   ]
```



deleteSlice

This property deletes a wedge from a pie chart (effectively, assigns the wedge a transparent color).

```
series: [
   {
         series: Number | 'string,
         group: Number, // optional
         deleteSlice: boolean
   }
]
```

PARAMETERS:

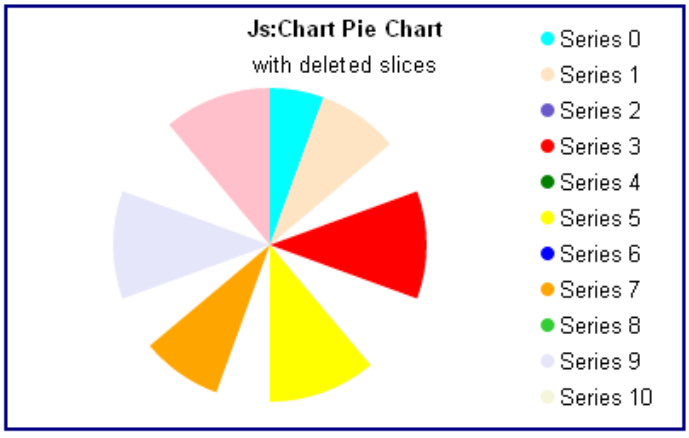
series: zero-based series number or 'all' to apply to all series. If the series does not exist in the chart, the property is ignored.

group: For multi-pie charts, optional zero-based group/pie number. If the group does not exist in the chart, the property is ignored. If a group number is not specified, the property is applied to all pies.

deleteSlice: true/false. true: delete the slice, false: restore the slice.

EXAMPLE:

```
border: {width: 2, color: 'navy'},
chartType: 'pie',
title: {text: 'Js:Chart Pie Chart'},
subtitle: {visible: true, text: 'with deleted slices'},
series: [
   {series: 0, color: 'cyan'},
   {series: 1, color: 'bisque'},
   {series: 2, color: 'slateblue', deleteSlice: true},
   {series: 3, color: 'red'},
   {series: 4, color: 'green', deleteSlice: true},
   {series: 5, color: 'yellow'},
   {series: 6, color: 'blue', deleteSlice: true},
   {series: 7, color: 'orange'},
   {series: 8, color: 'limegreen', deleteSlice: true},
   {series: 9, color: 'lavender'},
   {series: 10, color: 'beige', deleteSlice: true},
   {series: 11, color: 'pink'},
]
```



explodeSlice

This property defines the distance (in pixels) to push a slice away from the pie chart.

```
series: [
   {
         series: Number,
         group: Number, // optional
         explodeSlice: Number
   }
]
```

PARAMETERS:

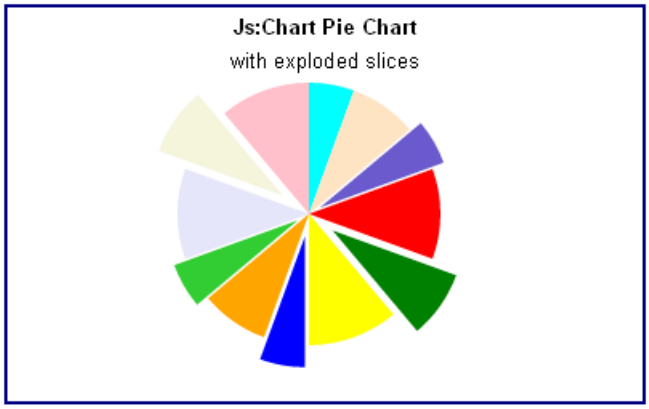
series: zero-based series number. If the series does not exist in the chart, the property is ignored.

group: For multi-pie charts, optional zero-based group/pie number. If the group does not exist in the chart, the property is ignored. If a group number is not specified, the property is applied to all pies.

explodeSlice: a number that defines the distance (in pixels) to push a slice away from the pie chart.

EXAMPLE:

```
border: {width: 2, color: 'navy'},
chartType: 'pie',
title: {text: 'Js:Chart Pie Chart'},
subtitle: {visible: true, text: 'with exploded slices'},
series: [
   {series: 0, color: 'cyan'},
   {series: 1, color: 'bisque'},
   {series: 2, color: 'slateblue', explodeSlice: 8},
   {series: 3, color: 'red'},
   {series: 4, color: 'green', explodeSlice: 18},
   {series: 5, color: 'yellow'},
   {series: 6, color: 'blue', explodeSlice: 14},
   {series: 7, color: 'orange'},
   {series: 8, color: 'limegreen', explodeSlice: 8},
   {series: 9, color: 'lavender'},
   {series: 10, color: 'beige', explodeSlice: 20},
   {series: 11, color: 'pink'},
]
```



label

This property assigns a label to an individual series that will be shown in the chart legend area. You can use the legend:labels property to define the format and color of the series label.

```
series: [
   {
         series: Number,
         label: 'string',
   }
]
```

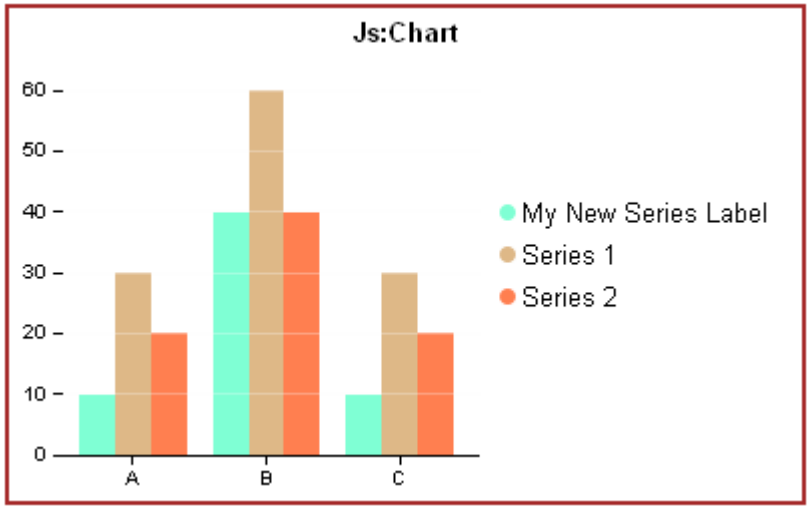
PARAMETERS:

series: zero-based series number. If the series does not exist in the chart, the property is ignored.

label: a string that defines the series label.

EXAMPLES:

```
blaProperties: {orientation: 'vertical'},
legend: {visible: true},
series: [
   {series: 0, color: 'aquamarine', label: 'My New Series Label'},
   {series: 1, color: 'burlywood'},
   {series: 2, color: 'coral'},
   ]
```



marker

These properties define the border color, size, shape and rotation of series markers.

```
series: [
   {
         series: Number,
         group: Number, // optional
         marker: {
               visible: boolean,  // Line Charts Only
               color: 'string',
               size: Number,
               shape: 'string',
               rotation: Number,
               position: 'string'
               border: {
                     width: Number,
                     color: 'string',
                     dash: 'string'
               }
         },
   }
]
```

PARAMETERS:

series: zero-based series number. If the series does not exist in the chart, the property is ignored.

group: (optional) zero-based group number. If the group does not exist in the chart, the property is ignored. If a group number is not specified, the marker definition is applied to all risers in the series.

visible: true/false controls the visibility of markers (in line charts only).

color: a color defined by a keyword or numerical specification string or a gradient defined by a string or JSON object. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string. See Gradient Definitions for the format of a linear or radial gradient.

size: a number that defines the size of the marker.

shape: a string that defines the shape of markers for a series: arrow, bar, circle, cross, diamond, fiveStar, hexagon, hourglass, house, pirateCross, plus, sixStar, square, thinPlus, tick, or triangle. Note that bar, cross, and tick markers require a border width and color.

rotation: a number 0...360 that defines the angle (in degrees) of the marker.

position: for bullet charts only, defines the position of the marker relative to data text: 'bottom', 'middle', 'top'. 'top' draws the data text label below the marker. 'bottom' draws the data text label above the marker. 'middle' draws the data text label according to the chart.dataLabels.position.

border/width: a number defines the width of the border in pixels.

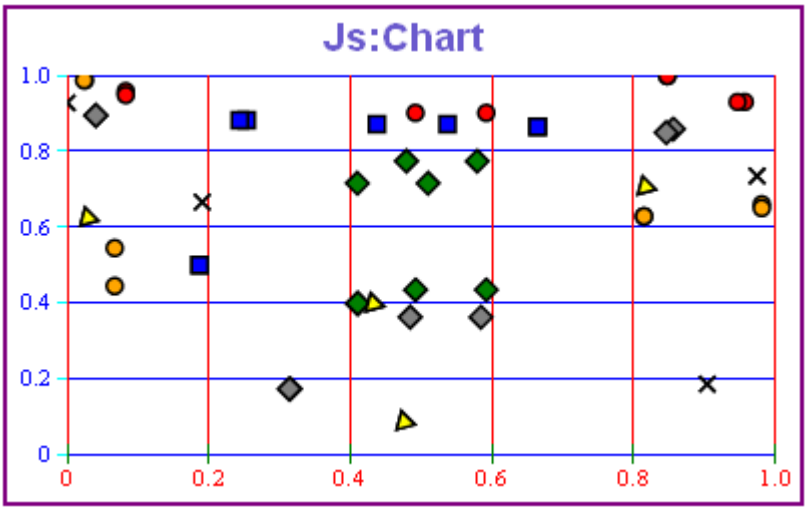
border/color: a color defined by a keyword or numerical specification string. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string.

border/dash: a string that defines the border dash style. Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes.

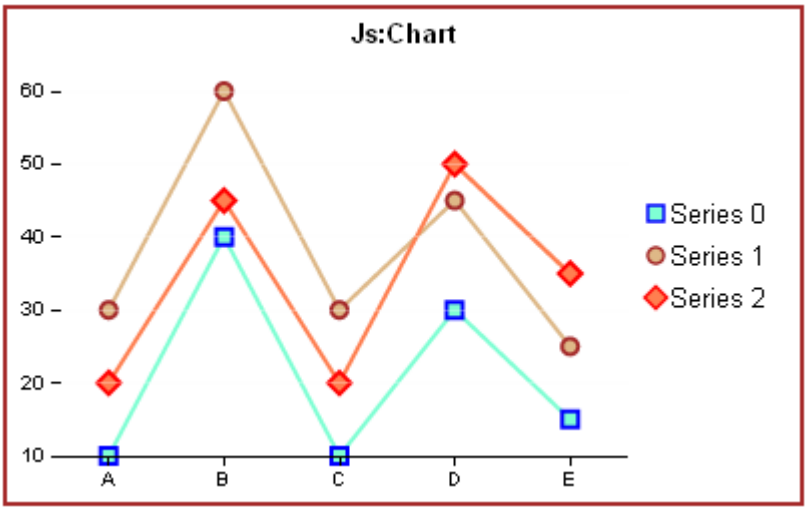
EXAMPLES:

```
chartType: 'scatter',
```

```
series: [
   {series: 0, color: 'red', marker:{shape: 'circle'}},
   {series: 1, color: 'green', marker:{shape: 'square'}},
   {series: 2, color: 'blue', marker:{shape: 'diamond'}},
   {series: 3, color: 'yellow', marker:{shape: 'triangle'}},
   {series: 4, color: 'purple', marker:{shape: 'cross'}},
   {series: 5, color: 'orange', marker:{shape: 'circle'}},
   {series: 6, color: 'grey', marker:{shape: 'square'}},
]
```

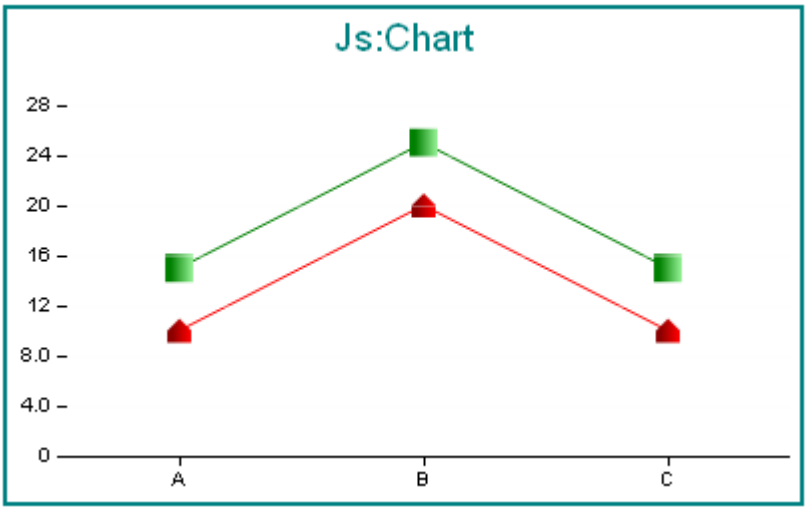



```
chartType: 'line',
blaProperties: {orientation: 'vertical', seriesLayout: 'absolute'},
legend: {visible: true},
series: [
   {series: 0, color: 'aquamarine', border: {width: 2}, marker:
{visible: true, shape: 'square', border: {color: 'blue'}}},
   {series: 1, color: 'burlywood', border: {width: 2}, marker:
{visible: true, shape: 'circle', border: {color: 'brown'})},
   {series: 2, color: 'coral', border: {width: 2}, marker: {visible:
true, shape: 'diamond', border: {color: 'red'}}},
]
```




```
chartType: 'line',
blaProperties: {orientation: 'vertical',seriesLayout: 'absolute'},
series: [
```

```
   {series: 0, color: 'red', marker: {visible: true, size: 12, shape:
'house', color: 'linear-gradient(0%,0%,100%,0%, 20% darkred, 95%
red)'}},
   {series: 1, color: 'green',  marker: {visible: true, size: 14,
shape: 'square', color: 'linear-gradient(0%,0%,100%,0%, 20% green,
95% lightgreen)'}}
]
```



riserShape

This property is used in conjunction with blaProperties:comboCharts to define a combination chart (i.e., a chart that can consist of a combination of bar risers, area risers, and line risers). When the chartType property is set to 'bar', 'line', or 'area', this property can be used to define the shape of risers (bar, line, or area) for each series. The comboCharts properties control the layout (stacked, absolute, percent, or sideBySide) of the risers for each series.

PARAMETERS:

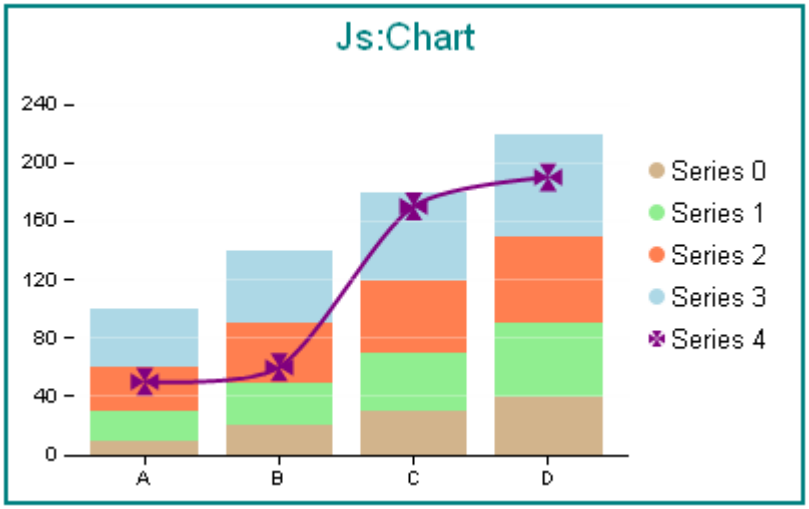
```
series: [
   {
         series: Number,
         riserShape: 'string'
   },
]
```

series: zero-based series number. If the series does not exist in the chart, the property is ignored.

riserShape: a string that defines the shape of the riser for the specified series: 'bar', 'line', or 'area'.

EXAMPLE:

```
legend: {visible: true},
blaProperties: {orientation: 'vertical',lineConnection:'curved',
   comboCharts: {barSeriesLayout: 'stacked', lineSeriesLayout:
'absolute',areaSeriesLayout: undefined}
},
series: [
{series: 0, color: 'tan', riserShape: 'bar'},
{series: 1, color: 'lightgreen', riserShape: 'bar'},
{series: 2, color: 'coral', riserShape: 'bar'},
{series: 3, color: 'lightblue', riserShape: 'bar'},
{series: 4, color: 'purple', riserShape: 'line', border: {width:
2},marker: {shape: 'pirateCross', size: 14, visible: true}}
]
```



showDataValues

This property enables/disables data text labels for individual series and groups.

```
series:
[
    (
         series: Number,
         group: Number,
         showDataValues: boolean,
   }
]
```

PARAMETERS:

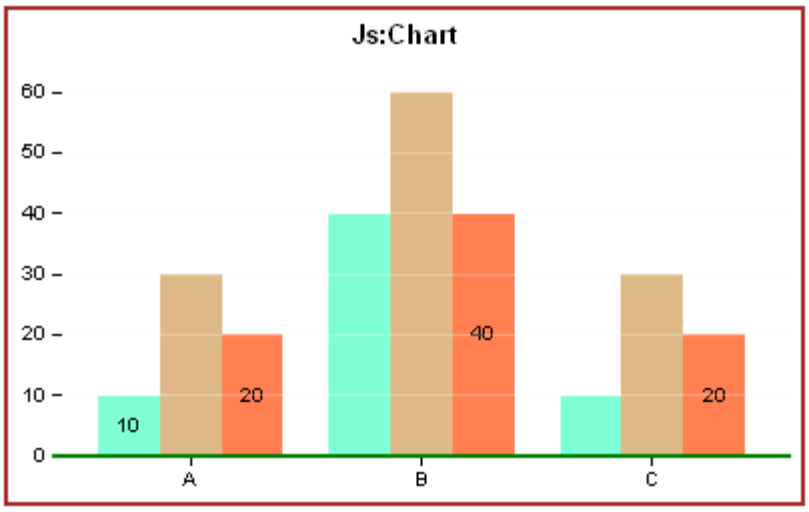
series: zero-based series number. If the series does not exist in the chart, the property is ignored.

group: (optional) zero-based group number. If the group does not exist in the chart, the property is ignored. If a group number is not specified, the marker definition is applied to all risers in the series.

showDataValues: When the dataLabels: visible property is true, use showDataValues false to turn off data labels for individual series/groups.

EXAMPLE:

```
blaProperties: {orientation: 'vertical'},
dataLabels: {visible: true},
series: [
   {series: 0, color: 'aquamarine'},
   {series: 0, group: 1, showDataValues: false},
   {series: 0, group: 2, showDataValues: false},
   {series: 1, color: 'burlywood', showDataValues: false},
   {series: 2, color: 'coral'}
]
```



tooltip

This property defines a tooltip for one or more risers. The tooltip string is shown when the mouse hovers over a riser. A tooltip can be assigned to: all risers/markers, a single riser/marker (identified by a series and group), or all risers/markers in a series.

```
series: [
   {
         series: Number | 'string',
         group: Number, // optional
         tooltip: 'string' | function
   }
]
```

PARAMETERS:

series: zero-based series number. If the series does not exist in the chart, the property is ignored. Use 'all' to define the tooltip for all risers.

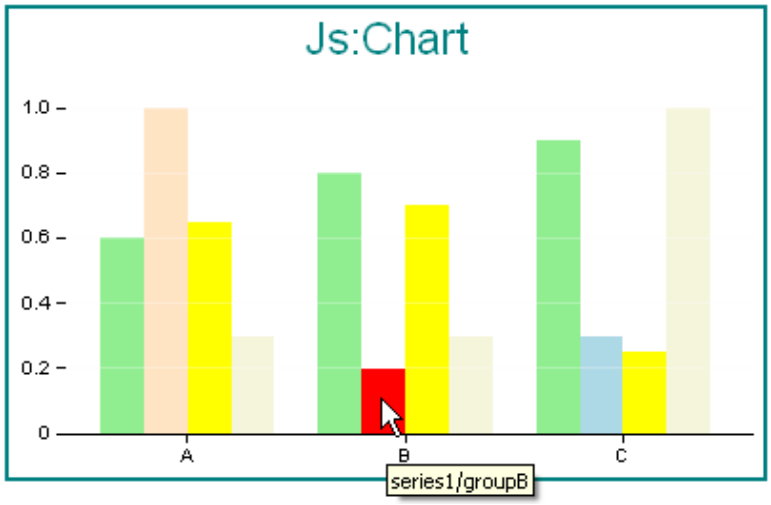
group: (optional) zero-based group number. If the group does not exist in the chart, the property is ignored. If a group number is not specified, the property is applied to all groups.

tooltip: a string or a function that returns a string to show when the mouse hovers over the riser. A callback function is invoked with three arguments: value, series ID, and group ID. Example:

```
chart.series[0].tooltip = function(value, series, group) {
    return this.title.text + ", value: " + value + ", s: " + series +
", g: " + g;
}
```

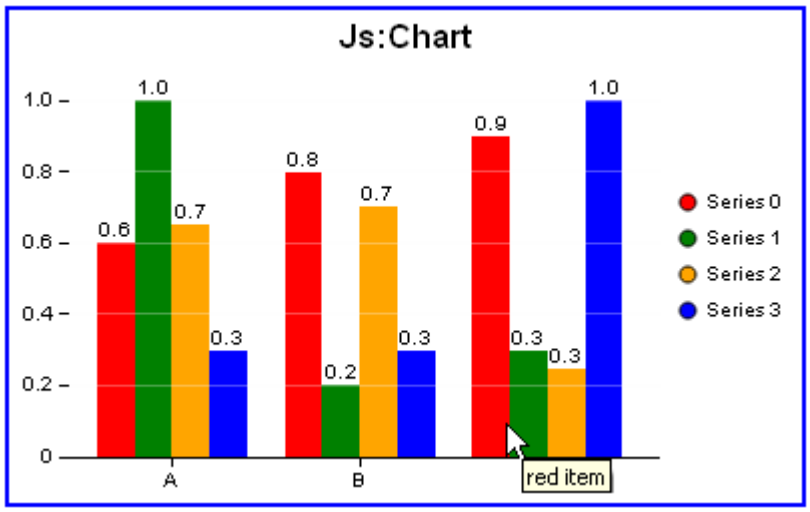
EXAMPLES:

```
series: [
{series: 0, color: 'lightgreen'},
{series: 1, group: 0, color: 'bisque'},
{series: 1, group: 1, color: 'red', tooltip: 'series1/groupB'},
{series: 1, group: 2, color: 'lightblue'},
{series: 2, color: 'yellow'},
{series: 3, color: 'beige'},
]
```




```
blaProperties: {orientation: 'vertical'},
legend: {visible: true},
```

```
series: [
{series: 0, color: 'red',  tooltip: 'red item'},
]
```



yAxisAssignment

This property assigns a series to an axis. When *yAxisAssignment* is set to 2, Y2-Axis labels are added to the chart. See the Numeric Axes for properties that control other y2axis objects.

```
series: [
   {
         series: Number,
         yAxisAssignment: number
   }
]
```

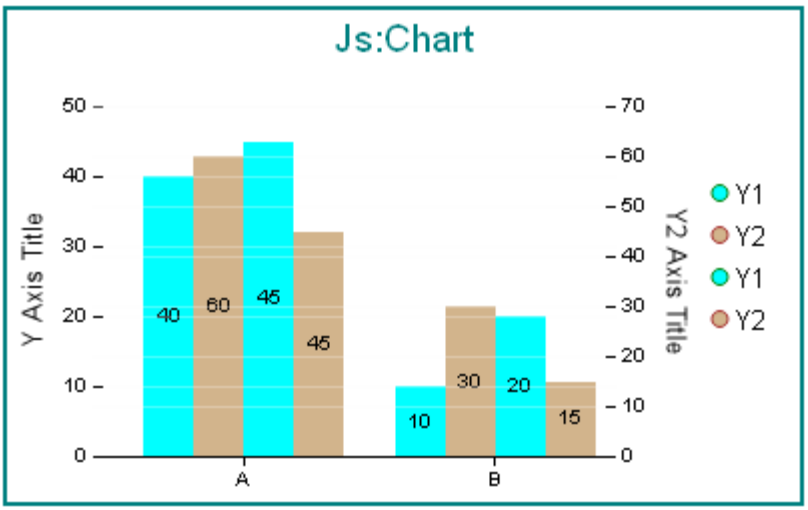
PARAMETERS:

series: zero-based series number. If the series does not exist in the chart, the property is ignored.

yAxisAssignment: 1 = assign series to Y-axis, 2 = assign series to Y2-Axis.

EXAMPLE:

```
blaProperties: {orientation: 'vertical'},
legend: {visible: true},
yaxis:{title: {visible: true}},
y2axis:{title: {visible: true}},
series:[
{series: 0, color: 'cyan', yAxisAssignment: 1, label: 'Y1', marker:
{border: {width: 1, color: 'green'}}},
{series: 1, color: 'tan', yAxisAssignment: 2, label: 'Y2', marker:
{border: {width: 1, color: 'brown'}}},
{series: 2, color: 'cyan', yAxisAssignment: 1, label: 'Y1', marker:
{border: {width: 1, color: 'green'}}},
{series: 3, color: 'tan', yAxisAssignment: 2, label: 'Y2', marker:
{border: {width: 1, color: 'brown'}}},
]
```



Section 4: Chart-Specific Properties

3D Chart Properties (threedProperties)

These properties control the general appearance of a area3D, bar3D, and surface3D charts.

```
threedProperties: {
   rotate: Number,
   tilt: Number,
   shadeSides: boolean
},
```

PARAMETERS:

rotate: a number 0...90 defines the rotation of the 3D cube. The default value is 40.

tilt: a number 0...90 defines the tilt of the 3D cube. The default value is 40.

shadeSides: true = shade 3D risers/ false = do not shade 3D risers. The default value is true.

Bar/Line/Area Chart Properties (blaProperties)

These properties control the basic layout of Bar / Line / Area Charts. The following code segment shows the default values from properties.js.

```
blaProperties: {
   seriesLayout: 'sideBySide',
   orientation: 'horizontal',
   lineConnection: 'linear',
   barGroupGapWidth: 0.2
   comboCharts: {
         barSeriesLayout: undefined,
         lineSeriesLayout: undefined,
         areaSeriesLayout: undefined,
   }
}
```

barGroupGapWidth

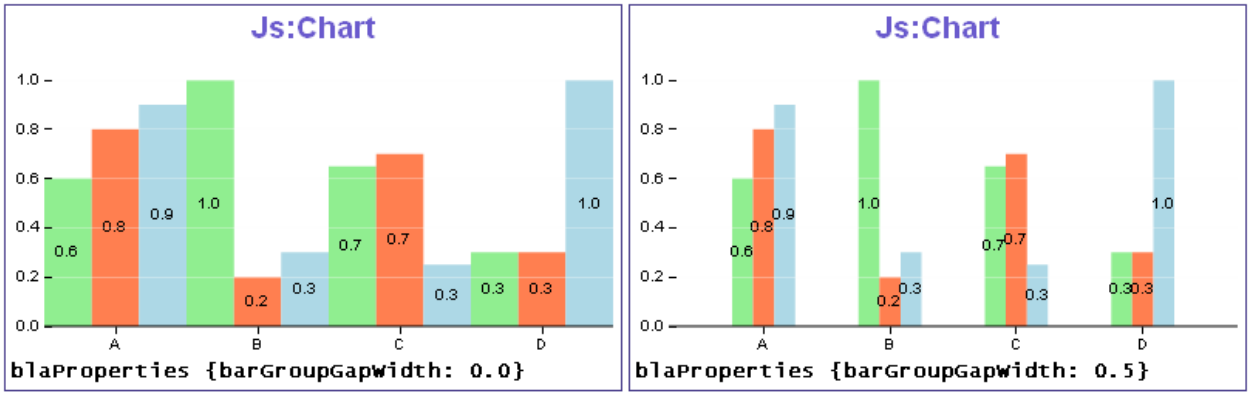
This property controls the width of the gap between groups of risers.

```
blaProperties: {
   barGroupGapWidth: Number
}
```

PARAMETERS:

barGroupGapWidth; 0.0 to 1.0 defines the width of the gap between groups of risers. The default value is 0.2.

EXAMPLE:



comboCharts

These properties are used in conjunction with the series#:riserShape property to define a combination chart (i.e., a chart that can consist of a combination of bar risers, area risers, and line risers). When the chartType property is set to 'bar', 'line', or 'area', the series#:riserShape property can be used to define the shape of risers (bar, line, or area) for each series. These properties control the layout (stacked, absolute, percent, or sideBySide) of the risers for each series.

```
blaProperties: {
   comboCharts: {
         barSeriesLayout: 'string' | undefined,
         lineSeriesLayout: 'string' | undefined,
         areaSeriesLayout: 'string' | undefined,
   }
}
```

PARAMETERS:

barSeriesLayout: a string or undefined. The default value is undefined. Use 'absolute', 'percent', 'sideBySide', or 'stacked' to define the layout of any series assigned to bar (series#:markerShape: 'bar'). undefined = use the series layout defined in blaProperties:seriesLayout.

lineSeriesLayout: a string or undefined. The default value is undefined. Use 'absolute', 'percent', or 'stacked' to define the layout of any series assigned to line (series#:markerShape: 'line'). undefined = use the series layout defined in blaProperties:seriesLayout.

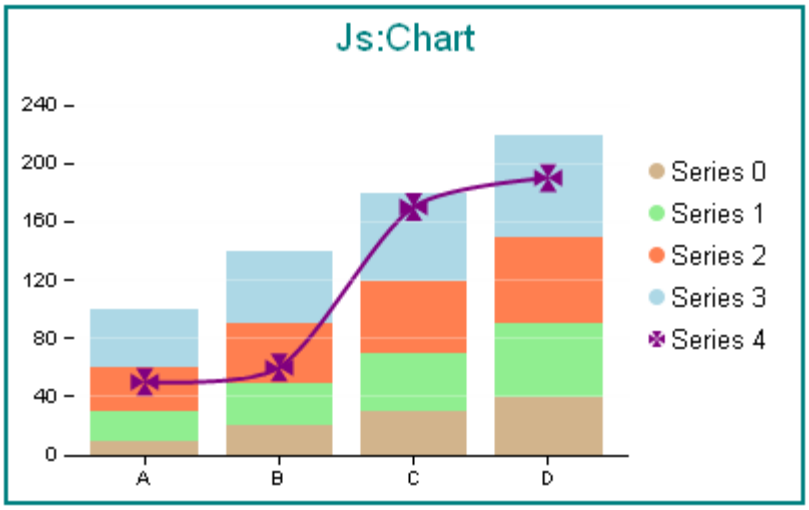
areaSeriesLayout: a string or undefined. The default value is undefined. Use 'absolute', 'percent', or 'stacked' to define the layout of any series assigned to area (series#:markerShape: 'area'). undefined = use the series layout defined in blaProperties:seriesLayout.

NOTES:

When a combination chart is defined with these properties and the series#:riserShape property, area risers are drawn first in the back, then bars, then lines in the front.

EXAMPLE:

```
data:
[[10,20,30,40],[20,30,40,50],[30,40,50,60],[40,50,60,70],[50,60,170,1
90]],
legend: {visible: true},
border: {width: 2, color: 'teal'},
title: {text: 'Js:Chart',font: '14pt Sans-Serif', color: 'teal'},
blaProperties: {orientation: 'vertical',lineConnection:'curved',
   comboCharts: {barSeriesLayout: 'stacked', lineSeriesLayout:
'absolute',areaSeriesLayout: undefined}
},
series: [
   {series: 0, color: 'tan', riserShape: 'bar'},
   {series: 1, color: 'lightgreen', riserShape: 'bar'},
   {series: 2, color: 'coral', riserShape: 'bar'},
   {series: 3, color: 'lightblue', riserShape: 'bar'},
   {series: 4, color: 'purple', riserShape: 'line',showDataValues:
true, border: {width: 2},marker: {shape: 'pirateCross', size: 14,
visible: true}}
]
```



lineConnection

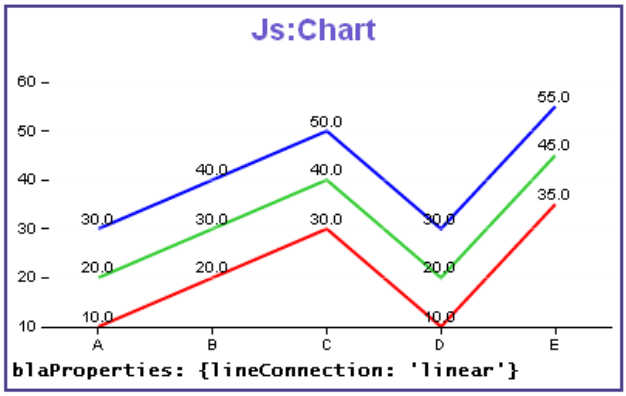
This property controls how data points are connected in line and area charts and in charts that use a combination of area and/or line risers using the comboCharts property and the series-specific riserShape property.

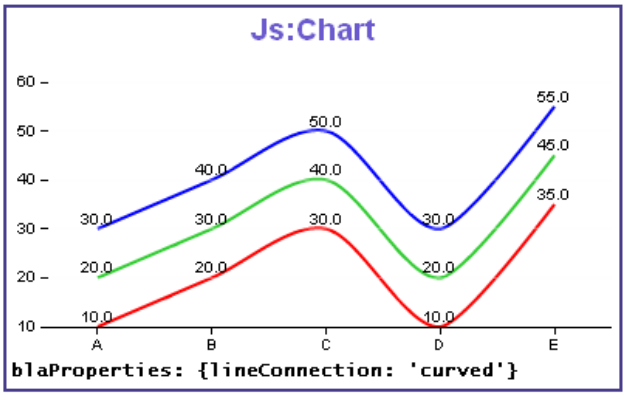
```
blaProperties: {
   lineConnection: 'string'
}
```

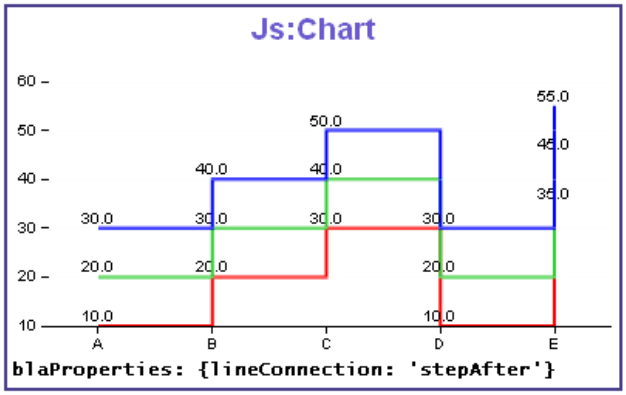
PARAMETERS:

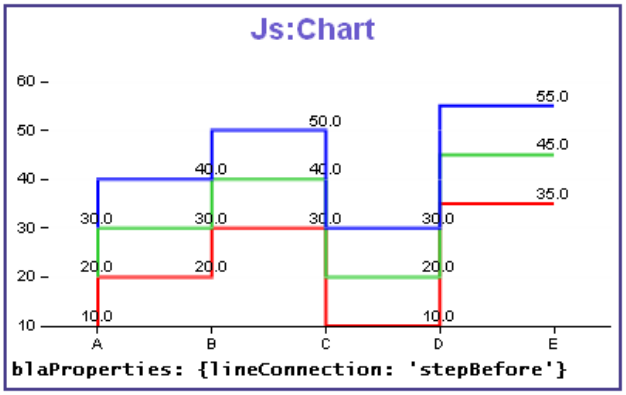
lineConnection; a string that defines the line connection: 'curved', 'linear', 'stepAfter', 'stepBefore', or 'stepBetween'. The default value is 'linear'.

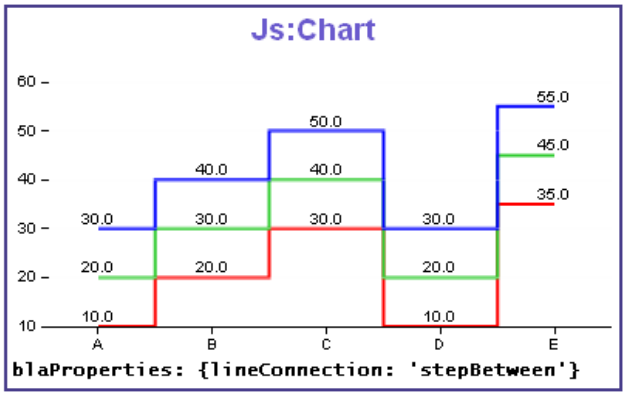
EXAMPLE:











orientation

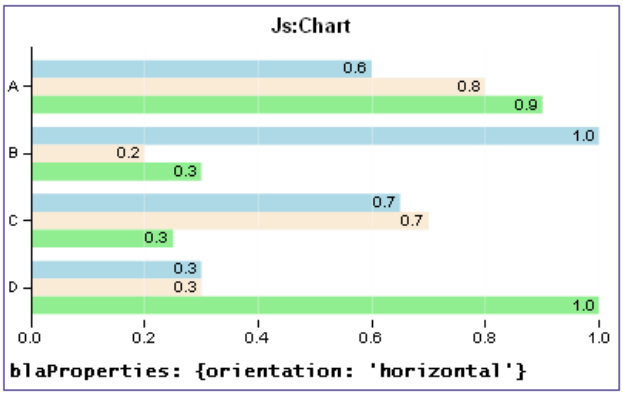
This property selects the orientation of a Bar, Line, or Area chart.

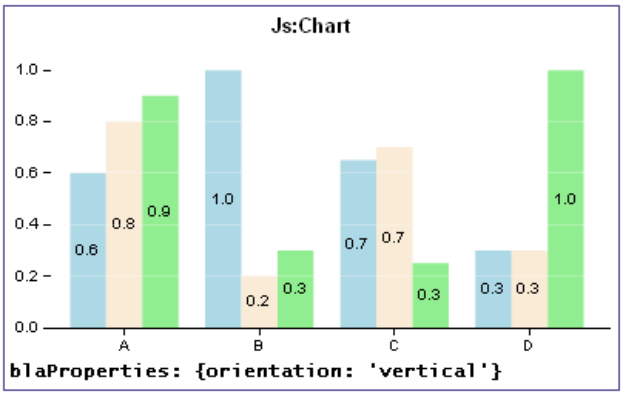
```
blaProperties: {
   orientation: 'string'
}
```

PARAMETERS:

orientation: 'horizontal' (default) or 'vertical'
'vertical' draws a vertical chart (X-axis on bottom, Y-Axis on left).
'horizontal' draws a horizontal chart (X-axis on left, Y-Axis on bottom)

EXAMPLE:





seriesLayout

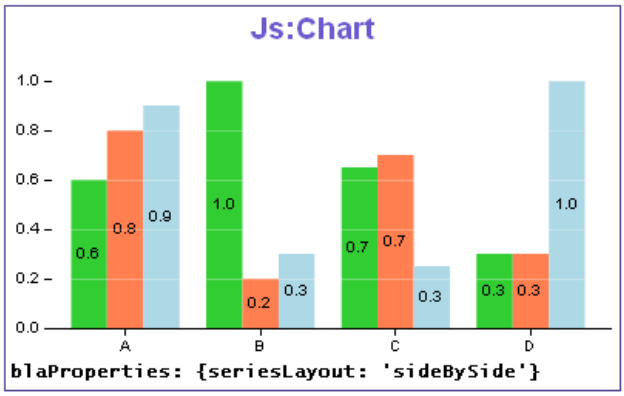
This property controls the arrangement of risers in a Bar, Line, or Area chart.

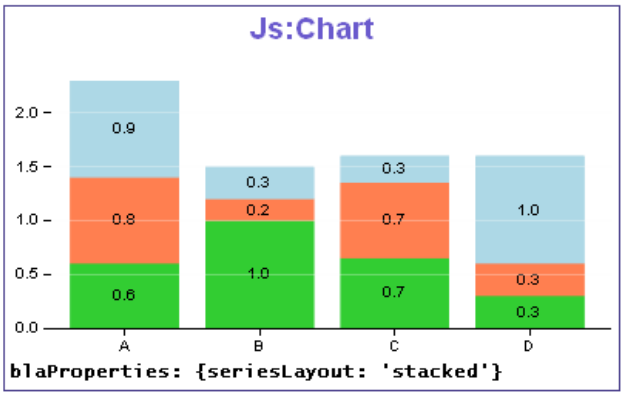
```
blaProperties: {
   seriesLayout: 'string'
}
```

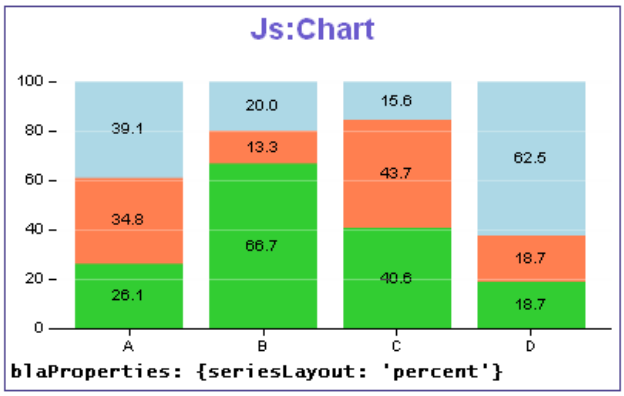
PARAMETERS:

seriesLayout; a string that defines the arrangement of risers: 'stacked', 'absolute', 'percent', or 'sideBySide' (bar charts only). The default value is 'sideBySide'.

EXAMPLES:







Box Plots (boxPlotProperties)

These properties control the general appearance of a box plot. The following code segment shows the default values from properties.js:

```
boxPlotProperties: {
   hatWidth: '100%',
   drawHatAsBox: false,
   medianLine: {
         width: 1,color: 'black',dash: ''
   },
   connectorLine: {
         width: 1,color: 'black',dash: ''
   }
}
```

connectorLine

These properties control the appearance of the connector line between the box plot and hats. The connector line is only visible when *drawHatAsBox* is false.

```
boxPlotProperties: {
   connectorLine: {
         width: Number,
         color: 'string',
         dash: 'string'
   }
},
```

PARAMETERS:

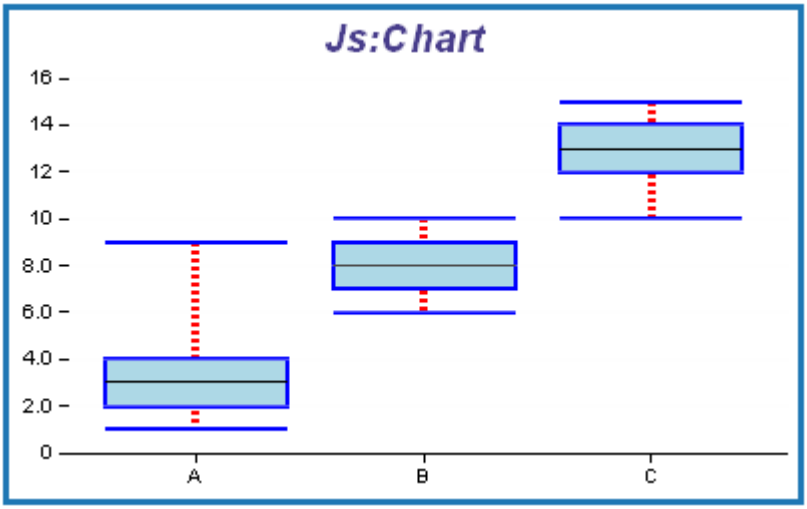
width: a number that defines the width of the line in pixels. The default value is 1.

color: a color defined by a keyword or numerical specification string. The default value is 'black'.

dash: a string that defines the line dash style. The default value is '' (a solid line). Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes.

EXAMPLE:

```
chartType: 'boxplot',
blaProperties: {orientation: 'vertical'},
boxPlotProperties: {
   connectorLine: {
         width: 4,
         color: 'red',
         dash: '2 2'
         }
   },
series: [
{series: 0, color: 'lightblue', border: {width: 2,color: 'blue'}},
],
```



drawHatAsBox

This property controls the appearance of the box plot hats (upper and lower).

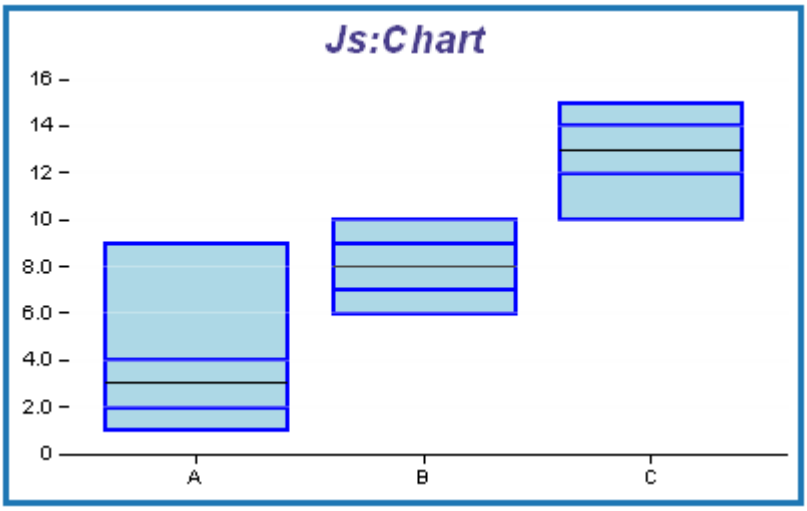
```
boxPlotProperties: {
   drawHatAsBox: boolean
},
```

PARAMETERS:

drawHatAsBox: true = draw box plot hats as boxes / false = draw normal hats. The default value is false.

EXAMPLE:

```
boxPlotProperties: {
   drawHatAsBox: true
},
```



hatWidth

This property controls the width of the hat that draws at the top and base of a box plot.

```
boxPlotProperties: {
   hatWidth: Number | 'string'
},
```

PARAMETERS:

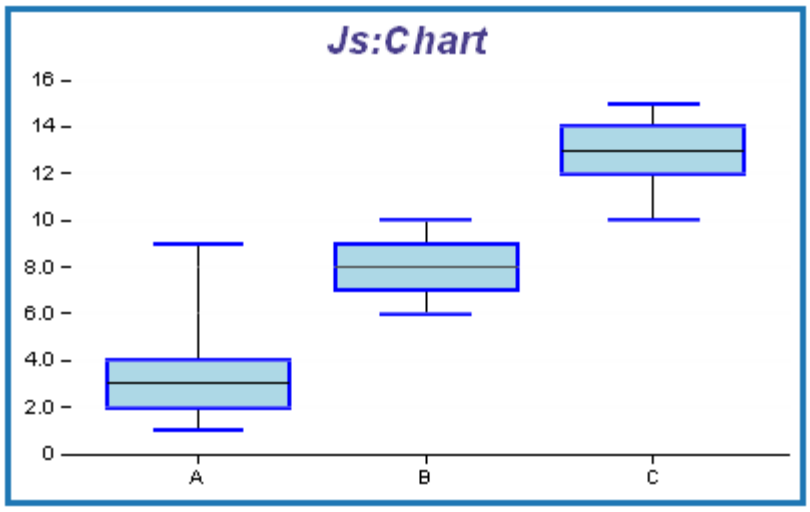
hatWidth: a string that expresses a percentage value ('0%'...'100%') or a number that defines the width of the hat in pixels. The default value is '100%' (the width of the box).

NOTES:

The series/group border properties control the color and format of the hats. You can set border:width to zero to remove the hats entirely (i.e., the same as hatWidth: '0%' or hatWidth: 0).

EXAMPLE:

```
data: [[[1,2,3,4,9],[6,7,8,9,10],[10,12,13,14,15]]],
chartType: 'boxplot',
blaProperties: {orientation: 'vertical'},
boxPlotProperties: {
   hatWidth: '50%',
   },
series: [
{series: 0, color: 'lightblue'},
],
```



medianLine

These properties control the appearance of the box plot median line.

```
boxPlotProperties: {
   medianLine: {
         width: Number,
         color: 'string',
         dash: 'string'
   }
},
```

PARAMETERS:

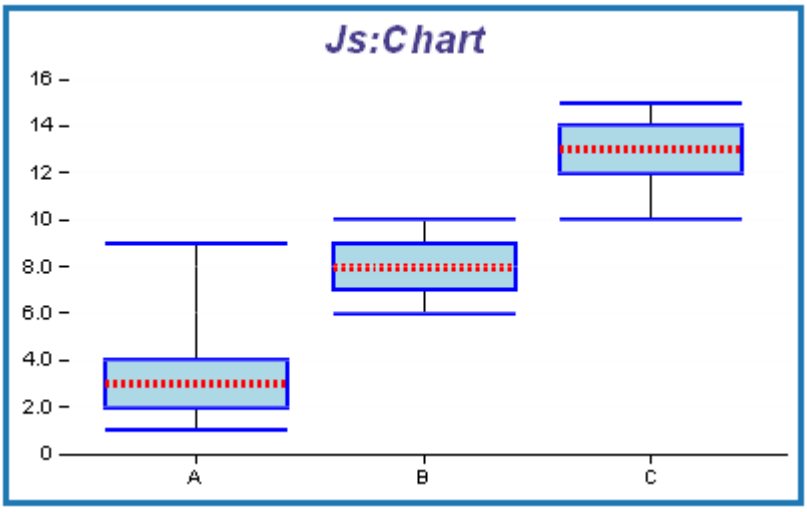
width: a number that defines the width of the line in pixels. The default value is 1.

color: a color defined by a keyword or numerical specification string. The default value is 'black'.

dash: a string that defines the line dash style. The default value is '' (a solid line). Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes.

EXAMPLE:

```
chartType: 'boxplot',
blaProperties: {orientation: 'vertical'},
boxPlotProperties: {
   medianLine: {
         width: 4,
         color: 'red',
         dash: '2 2'
         }
   },
series: [
{series: 0, color: 'lightblue', border: {width: 2,color: 'blue'}},
],
```



Bullet Charts (bulletProperties)

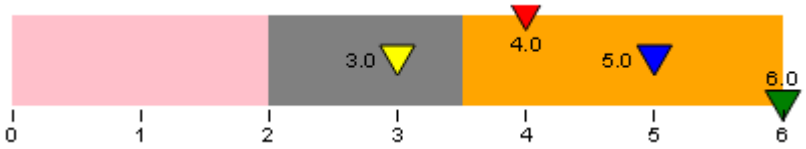
This property controls the general appearance of a bullet chart.

```
bulletProperties: {
   drawFirstValueAsBar: boolean
},
```

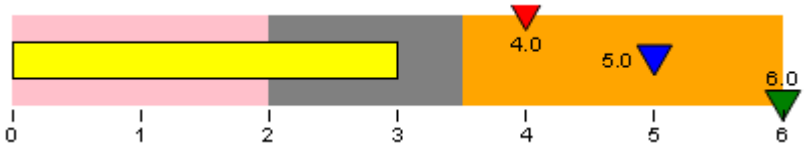
drawFirstValueAsBar: true = draw first data value as a bar, false = draw first value as a marker. The default value is true.

EXAMPLE:

```
data: [[3, 4, 5, 6]],
chartType: 'bullet',
bulletProperties: {drawFirstValueAsBar: false},
```




```
data: [[3, 4, 5, 6]],
chartType: 'bullet',
bulletProperties: {drawFirstValueAsBar: true},
```



Funnel Chart (funnelProperties)

These properties control the general layout of a funnel chart. The following code segment shows the default values.

```
funnelProperties: {
   topWidth: '90%',
   baseWidth: '20%',
   riserGap: 0,
   groupLabel: {
         visible: false,
         font: '10pt Sans-Serif',
         color: 'black'
   }
},
```

funnelProperties: {

topWidth: '90%', // Either a string percentage of overall width, or a number in pixels

riserGap: 0, // like topWidth (string percent or pixel number)

baseWidth

This property controls the width of the base of the funnel.

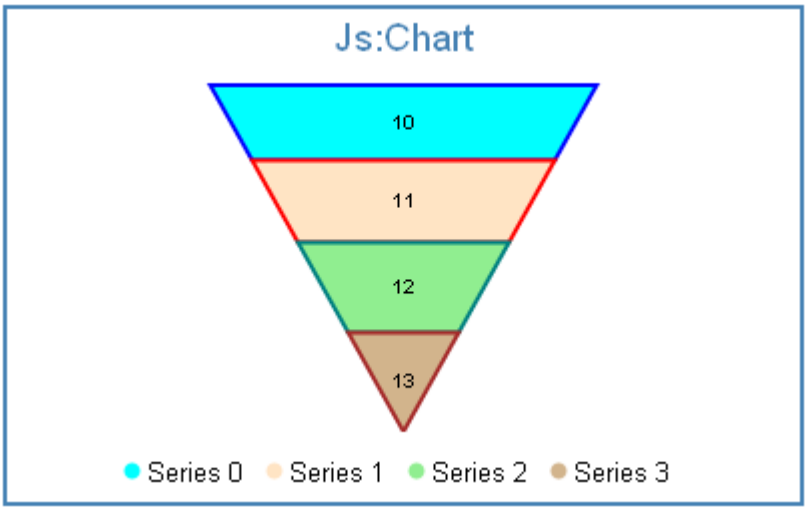
```
funnelProperties: {
   baseWidth: Number | 'string'
},
```

PARAMETERS:

baseWidth; a number or string that defines the width of the base of the funnel. A number defines the width in pixels. A string must represent a percentage value (i.e., '0%' ... '100%'). The default value is '20%'.

EXAMPLE:

```
data: [[10,11,12,13]],
chartType: 'funnel',
funnelProperties: {
   topWidth: '50%',
   baseWidth: 1
},
series: [
{series: 0, color: 'cyan', showDataValues: true, border: {width: 2,
color: 'blue'}},
{series: 1, color: 'bisque', showDataValues: true, border: {width: 2,
color: 'red'}},
{series: 2, color: 'lightgreen', showDataValues: true, border:
{width: 2, color: 'teal'}},
{series: 3, color: 'tan', showDataValues: true, border: {width: 2,
color: 'brown'}},
]
```



groupLabel

These properties control the visibility and format of the group label in a funnel chart.

```
funnelProperties: {
   groupLabel: {
         visible: boolean,
         font: 'string',
         color: 'string'
   }
},
```

PARAMETERS:

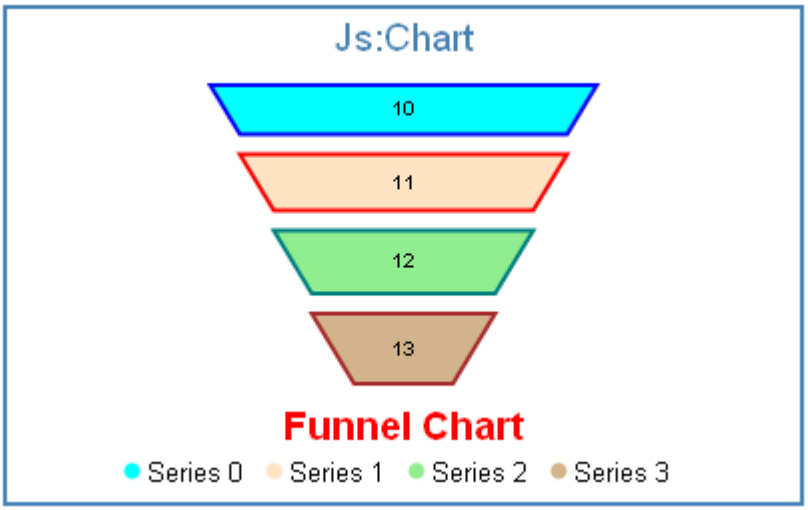
visible: true/false = show/hide group label. The default value is false.

font; a string that defines the size and type face of the label. See http://www.w3.org/TR/CSS2/fonts.html#font-shorthand for the format of this string. The default value is '10pt Sans-Serif'.

color: a color defined by a keyword or numerical specification string. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string. The default value is 'black'.

EXAMPLE:

```
data: [[10,11,12,13]],
chartType: 'funnel',
funnelProperties: {
   topWidth: '50%', baseWidth: 1,riserGap: 10
   groupLabel: {
         visible: true,font: 'bold 14pt Sans-Serif',color: 'red'
   }
},
groupLabels: ["Funnel Chart"],
series: [
{series: 0, color: 'cyan', border: {width: 2, color: 'blue'}},
{series: 1, color: 'bisque', border: {width: 2, color: 'red'}},
{series: 2, color: 'lightgreen', border: {width: 2, color: 'teal'}},
{series: 3, color: 'tan', border: {width: 2, color: 'brown'}},
]
```



riserGap

This property controls the empty space between layers in the funnel.

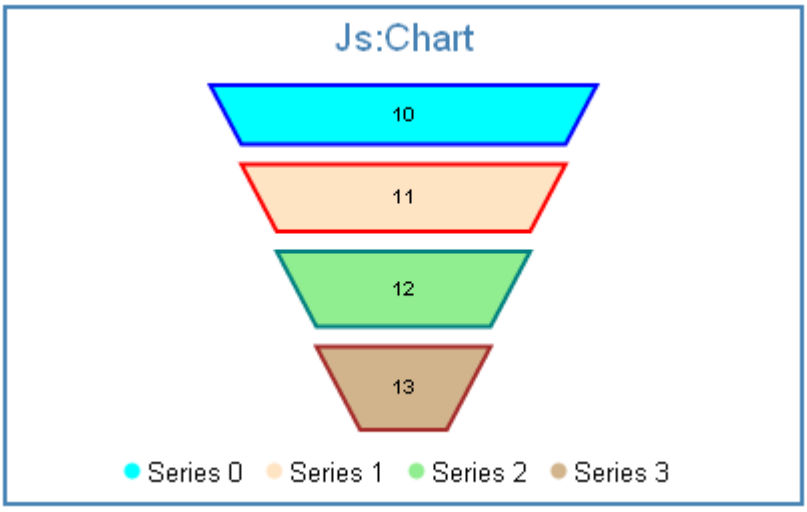
```
funnelProperties: {
   riserGap: Number | 'string'
},
```

PARAMETERS:

riserGap; a number or string that defines the width of the riser gap between funnel layers. A number defines the width of the gap in pixels. A string must represent a percentage value '0%' ... '100%' (of average riser height). For example, riserGap: '10%' will use 10% of each funnel layer as white space between each segment. The default value is zero.

EXAMPLE:

```
data: [[10,11,12,13]],
chartType: 'funnel',
funnelProperties: {
   topWidth: '50%',
   baseWidth: 1,
   riserGap: 10
},
series: [
{series: 0, color: 'cyan', border: {width: 2, color: 'blue'}},
{series: 1, color: 'bisque', border: {width: 2, color: 'red'}},
{series: 2, color: 'lightgreen', border: {width: 2, color: 'teal'}},
{series: 3, color: 'tan', border: {width: 2, color: 'brown'}},
]
```



topWidth

This property controls the width of the top of the funnel.

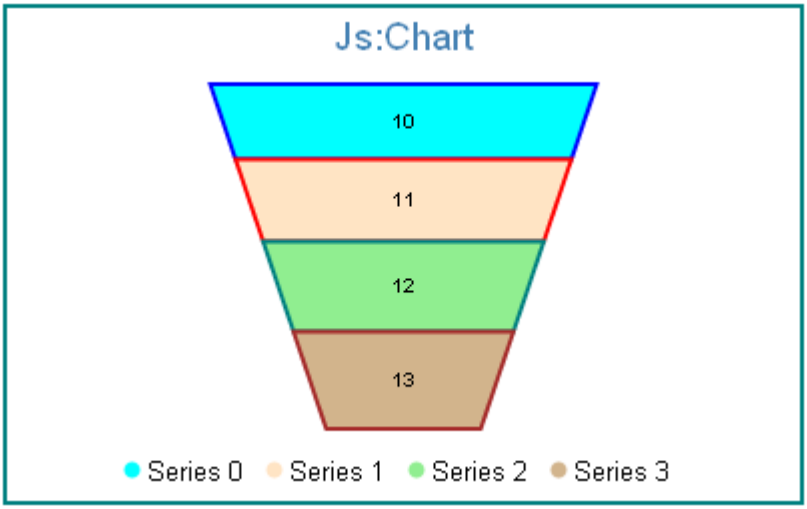
```
funnelProperties: {
   topWidth: Number | 'string'
},
```

PARAMETERS:

topWidth; a number or string that defines the width of the top of the funnel. A number defines the width in pixels. A string must represent a percentage value (i.e., '0%' ... '100%'). The default value is '90%'.

EXAMPLE:

```
data: [[10,11,12,13]],
chartType: 'funnel',
funnelProperties: {
   topWidth: '50%',
},
series: [
{series: 0, color: 'cyan', border: {width: 2, color: 'blue'}},
{series: 1, color: 'bisque', border: {width: 2, color: 'red'}},
{series: 2, color: 'lightgreen', border: {width: 2, color: 'teal'}},
{series: 3, color: 'tan', border: {width: 2, color: 'brown'}},
]
```



Gantt Charts (ganttProperties)

These properties define the general appearance and layout of a gantt chart.

```
ganttProperties: {
   startTime: 'string' | undefined,
   stopTime: 'string' | undefined,
   interval: 'string' | undefined,
   labelFormat: 'string' | undefined,
   durationValues: boolean,
   staggerRisers: boolean
},
```

startTime: a string identifying the start time. It can be almost any kind of string denoting time, mixing hours (xx:xx), days of the week, dates, months and years. Examples: "Mon", "1 July 20", "June 2001", "8:00", "6/10/01", "Thu, 4 Apr 1976 14:30", "2 0:00" (February 1), "1 1 0:00" (January 1). The default value is undefined.

stopTime: a string identifying the start time. As with startTime, the string can be almost any date/time format. The default value is undefined.

interval: a string that specifies the time interval: 'seconds', 'minutes', 'hours', 'days', 'weeks', 'months', 'quarters', or 'years'. This property defines the time gap between each group label. If undefined, the library will assign an interval based on the startTime, stopTime, and the number of groups. The default value is undefined.

labelFormat: a string that defines how a time/date number is converted into a label. If undefined, the library will we use a default formats based on the interval. See http://code.google.com/p/datejs/wiki/FormatSpecifiers for available format options. The default value is undefined.

durationValues: true/false controls the use of the second value in the data set that defines each riser. If false, the second data point is treated as a stop time. If true, the second data point is treated as a duration. The default value is false.

staggerRisers: If false, risers within same group are drawn in one column, stacked above (or beside) each other. In this mode, consecutive risers will be forced to have consecutive start / stop times. If true, risers within same group are drawn beside each other. In this mode, risers can have any arbitrary start / stop times. The default value is true.

Gauge Properties (gaugeProperties)

A gauge chart represents one or more values as needles or a single needle and markers on a circular surface. This chart type is typically used by executive dashboard applications.

The following properties control the basic layout of gauge charts. This code segment shows the default settings from properties.js.

```
gaugeProperties: {
   startAngle: 135,
   endAngle: 45,
   secondaryNeedlesAsMarkers: false,
   groupLabel: {
         visible: false,
         font: '10pt Sans-Serif',
         color: 'black'
   },
   fill: {
         color: 'transparent'
   },
   needleBase: {
         size: "6%",
         color: '#1f77b4',
         border: {
               width: 0,
               color: 'transparent',
               dash: ''
         }
   },
   axisWidth: "22%",
   axisTickLength: "30%",
   outerBorder: {
         width: '10%',
         fill: {
               color: '#1f77b4'
         },
         border: {
               width: 0,
               color: 'transparent',
               dash: ''
         }
   }
},
```

The numeric yaxis properties control the range and format of values, the format of grid lines, and color bands shown on the gauge axis.

axisTickLength

This property defines with length of axis tick marks.

```
gaugeProperties: {
   axisTickLength: Number | 'string'
},
```

PARAMETERS:

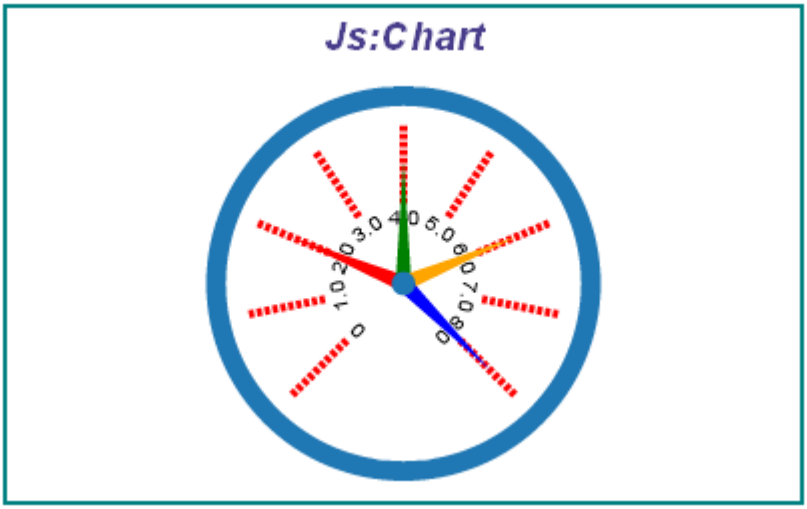
axisTickLength: a number or string that defines the length of tick marks. A number defines the length in pixels. A string must express a percent value that defines the width as a percentage of the overall gauge. The default value is '30%'.

NOTES:

Use yaxis:majorGrid to format the tick marks.

EXAMPLE:

```
gaugeProperties: {
   axisTickLength: '50%',
},
yaxis: {
   majorGrid: {
         lineStyle: {
         width: 4,
               color: 'red',
               dash: '2 2'
         },
   }
},
```



axisWidth

This property defines the width of the gauge axis.

```
gaugeProperties: {
   axisWidth: number | 'string'
},
```

PARAMETERS:

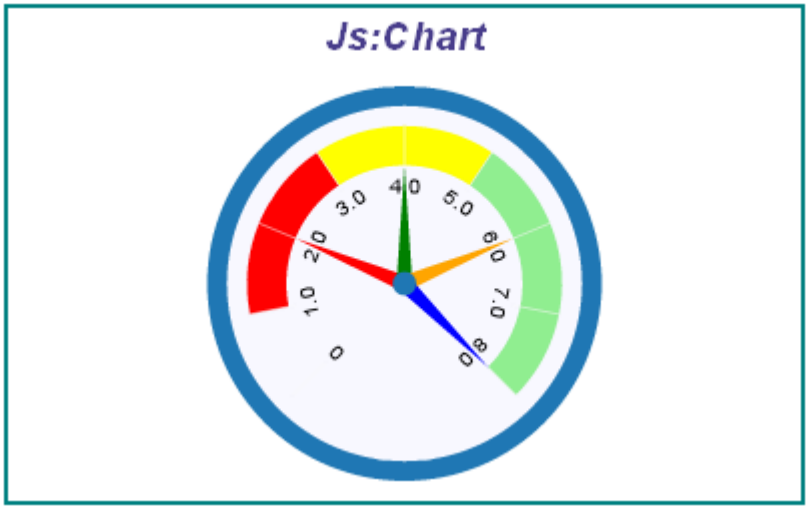
axisWidth: a number or string that defines the axis width. A number defines the width in pixels. A string must express a percent value that defines the width as a percentage of the overall gauge. The default value is '22%'.

NOTES:

Use yaxis:colorBands (as shown in the examples) to apply color to the gauge axis.

EXAMPLE:

```
chartType: 'gauge',
gaugeProperties: {
   axisWidth: '25%',
},
yaxis: {
   colorBands: [
         {start: 1,stop: 3,color: 'red'},
         {start: 3,stop: 5,color: 'yellow'},
         {start: 5,stop: 8,color: 'lightgreen'},
   ],
},
```



fill

This property controls the fill color of the interior of the gauge face.

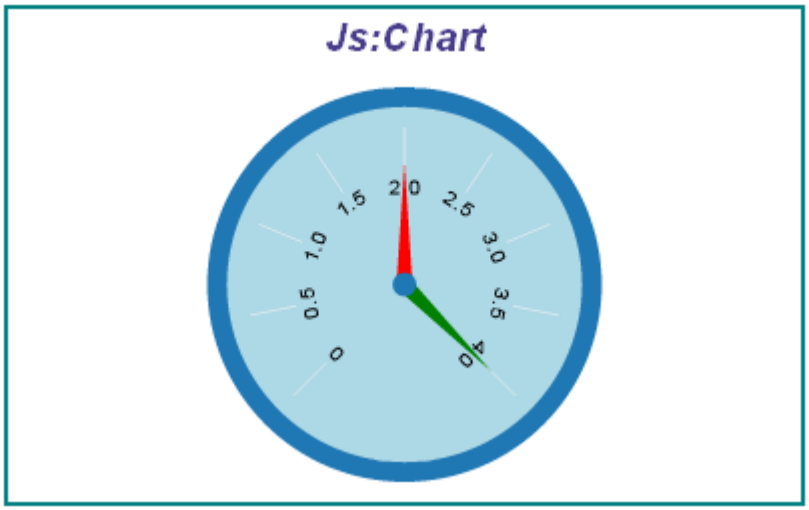
```
gaugeProperties: {
   fill: {
         color: 'string' | JSON Object
   },
},
```

PARAMETERS:

color: a color defined by a keyword or numerical specification string or a gradient defined by a string or JSON object. The default value is transparent. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string. See Gradient Definitions for the format of a linear or radial gradient.

EXAMPLE:

```
chartType: 'gauge',
gaugeProperties: {
   fill: {color: 'lightBlue'}
},
```



groupLabel

These properties control the visibility and format of the group label in a gauge chart.

```
gaugeProperties: {
   groupLabel: {
         visible: false,
         font: '10pt Sans-Serif',
         color: 'black'
   },
```

PARAMETERS:

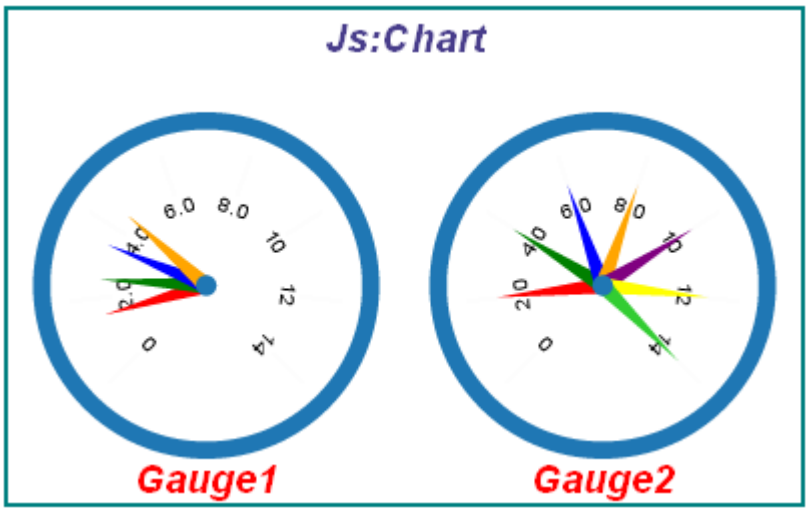
visible: true/false = show/hide group label. The default value is false.

font; a string that defines the size and type face of the label. See http://www.w3.org/TR/CSS2/fonts.html#font-shorthand for the format of this string. The default value is '10pt Sans-Serif'.

color: a color defined by a keyword or numerical specification string. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string. The default value is 'black'.

EXAMPLE:

```
groupLabels: "Gauge1 Gauge2",
chartType: 'gauge',
gaugeProperties: {
   secondaryNeedlesAsMarkers: false,
   groupLabel: {
         visible: true,
         font: 'bold italic 14pt Sans-Serif',
         color: 'red'
   },
}
```



needleBase

These properties control the format of the base of the gauge needle.

```
gaugeProperties: {
   needleBase: {
         size: number | 'string',
         color: 'string' | JSON object,
         border: {
               width: number,
               color: 'string',
               dash: 'string'
         }
   },
},
```

PARAMETERS:

size: a number or string that defines the radius of the base of the gauge needle. A number defines the radius in pixels. A string must express a percent value that defines the radius as a percentage of the overall gauge. The default value is '6%'.

color: a color defined by a keyword or numerical specification string or a gradient defined by a string or JSON object. The default value is '#1f77b4'. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string. See Gradient Definitions for the format of a linear or radial gradient.

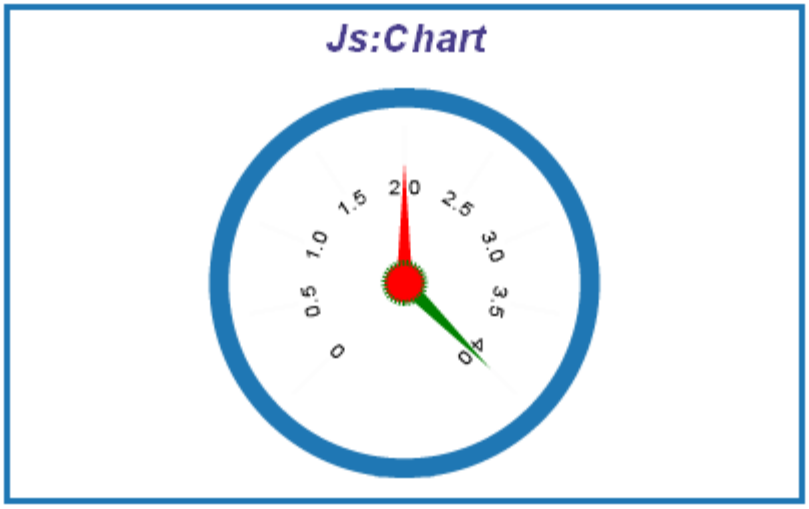
border/width: a number that defines the width of the border around the needle base. The default value is zero (no border).

border/color: a color defined by a keyword or numerical specification string. The default value is 'transparent' (no border).

border/dash: a string that defines the border dash style. The default value is '' (a solid line). Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes.

EXAMPLE:

```
gaugeProperties: {
   needleBase: {size: 10,color: 'red',
         border: {width: 3,color: 'green',dash: '1 1'}}
},
```



outerBorder

These properties control the format of a gauge's outer border.

```
gaugeProperties: {
   outerBorder: {
         width: number | 'string'
         fill: {
               color: 'string' | JSON Object
         },
         border: {
               width: number,
               color: 'string',
               dash: 'string'
         }
   }
},
```

PARAMETERS:

width: a number or a percent string. A number defines the width (in pixels) of the outer border ring. A percent string defines the width of the outer border as a percentage of the gauge size. The default value is '10%'.

fill/color: a color defined by a keyword or numerical specification string or a gradient defined by a string or JSON object. The default value is '#1f77b4'. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string. See Gradient Definitions for the format of a linear or radial gradient.

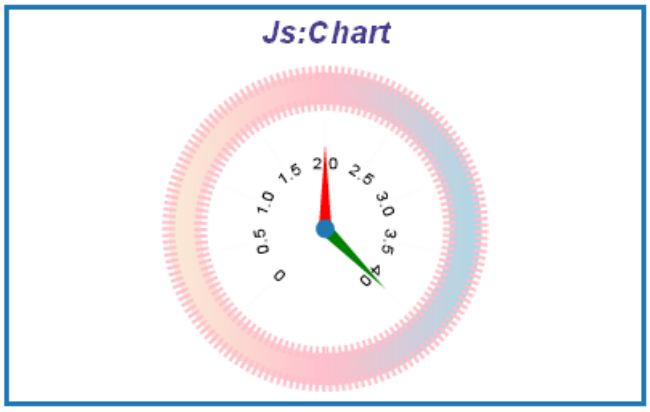
border/width: a number that defines the width (in pixels) of the border of the outerborder. The default value is zero.

border/color: a color defined by a keyword or numerical specification string. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string. The default value is transparent.

border/dash: a string that defines the border dash style. The default value is '' (a solid line). Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes.

EXAMPLE:

```
gaugeProperties: {
   outerBorder: {width: 20, fill: {color: 'linear-
gradient(0%,0,100%,0, 0 antiquewhite, 0.5 pink, 1 lightblue)'},
   border: {width: 8,color: 'pink',dash: '2 2'}}
},
```



secondaryNeedlesAsMarkers

When a gauge includes multiple needles, use the property to draw secondary needles as markers.

```
gaugeProperties: {
   secondaryNeedlesAsMarkers: boolean,
},
```

PARAMETERS:

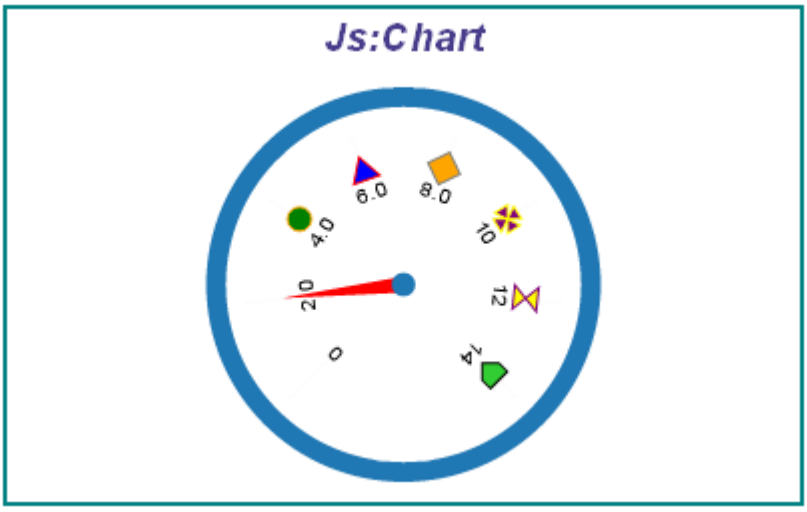
secondaryNeedlesAsMarkers: true = show secondary needles as markers, false = draw all values as multiple needles. The default value is false (multiple needles).

NOTES:

When secondaryNeedlesAsMarkers is false, use the series:marker property to control the size and format of markers.

EXAMPLE:

```
chartType: 'gauge',
gaugeProperties: {
   secondaryNeedlesAsMarkers: true
},
series: [
{series: 0, color: 'red',marker: {shape: 'square', size: 12, border:
{width: 1, color: 'black'}}},
{series: 1, color: 'green',marker: {shape: 'circle', size: 12,
border: {width: 1, color: 'orange'}}},
{series: 2, color: 'blue',marker: {shape: 'triangle', size: 12,
border: {width: 1, color: 'red'}}},
{series: 3, color: 'orange',marker: {shape: 'diamond', size: 12,
border: {width: 1, color: 'grey'}}},
{series: 4, color: 'purple',marker: {shape: 'pirateCross', size: 12,
border: {width: 1, color: 'yellow'}}},
{series: 5, color: 'yellow',marker: {shape: 'hourglass', size: 12,
border: {width: 1, color: 'purple'}}},
{series: 6, color: 'limegreen',marker: {shape: 'house', size: 12,
border: {width: 1, color: 'black'}}},
]
```



startAngle/endAngle

These properties control the start/end angle of the gauge value bands. Angles are defined in degrees. Positive angles draw clockwise, negative angles draw counter clockwise. Zero will start or end the band at the 3 o'clock position. 90 will start or end the band at the 6 o'clock position. -90 specifies the 12 o'clock position.

```
gaugeProperties: {
   startAngle: number,
   endAngle: number
},
```

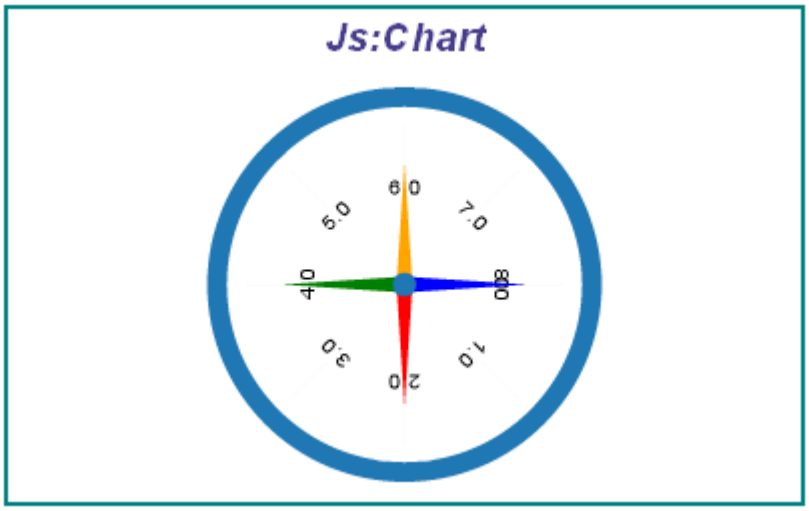
PARAMETERS:

startAngle: a number that defines the angle (in degrees) to start the gauge value band. The default value is 135.

endAngle: a number that defines the angle (in degrees) to end the gauge band value. The default value is 45.

EXAMPLE:

```
chartType: 'gauge',
gaugeProperties: {
   startAngle: 0,
   endAngle: 0
},
```



Heatmap Charts (heatmapProperties)

These properties control the color of cells in a heat map:

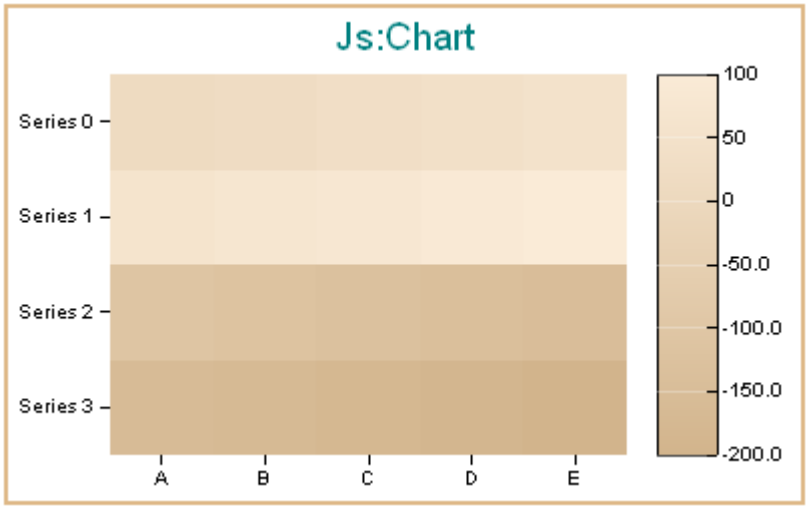
```
heatmapProperties: {
   dataColors: ['string',...,'string']
},
```

PARAMETERS:

dataColors: an array of color string that will be used to fill in the table according to each data value. The default colors are: ['#FFFFCC', '#A1DAB4', '#41B6C4','#2C7FB8','#253494'].

EXAMPLE:

```
data: [
[10,20,30,40,50],[60,70,80,90,100],
[-110,-120,-130,-140,-150],[-160,-170,-180,-190,-200]
],
chartType: 'heatmap',
heatmapProperties: {dataColors: ['tan', 'antiquewhite']},
legend: {visible: true},
```



Histogram Charts (histogramProperties)

A histogram is a graphical representation that shows a visual impression of the distribution of data. These properties control the distribution of data.

```
histogramProperties: {
   binCount: Number | undefined,
   binSize: Number | Array | undefined
   startBinValue: Number | undefined
   },
```

PARAMETERS:

binCount: number of group labels to draw or undefined (automatic). The default value is undefined.

binSize: a number, an array of numbers (for variable bin sizes), or undefined (automatic). If *binSize* is an array, *binCount* is ignored and assumed to be the length of the *binSize* array. The default value is undefined.

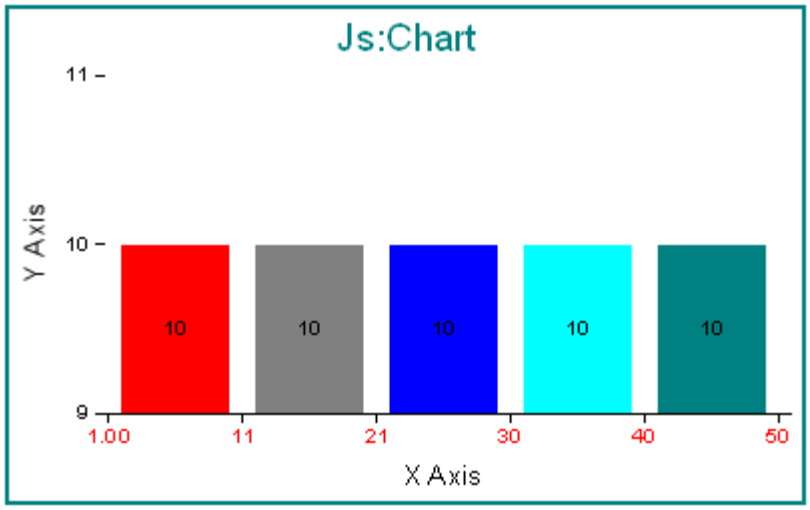
startBinValue: a number to use as the first bin or undefined (automatic). The default value is undefined.

NOTES:

- Use blaProperties: orientation to draw the histogram in vertical or horizontal format.
- blaProperties: barGroupGapWidth can be used to modify the width and space between histogram risers.
- The series color and border properties can be used to format histogram risers.
- The axis:invert property can be used to reverse the order of the axis.
- Use yaxis properties to control the format of Y-axis title and labels.
- Use xaxisOrdinal properties to control the format of X-axis title and labels. Ordinal axis labels are calculated. They are not defined by the groupLabels property.
- In a vertical histogram, the Y-axis is on the left side of the chart and the ordinal X-axis is drawn on the bottom of the chart. In a horizontal histogram, the ordinal X-axis is on the left side of the chart and the Y-axis axis is drawn on the bottom of the chart.

EXAMPLE:

```
data = [[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,
   20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,
   36,37,38,39,40,41,42,43,44,45,46,47,48,49,50]],
chartType: 'histogram',
blaProperties: {orientation: 'vertical'},
histogramProperties: {
   binCount: 5,
},
series: [
{series: 0, group: 0, color: 'red'},
{series: 0, group: 1, color: 'grey'},
{series: 0, group: 2, color: 'blue'},
{series: 0, group: 3, color: 'cyan'},
{series: 0, group: 4, color: 'teal'},
]
```



Pie Chart Properties (pieProperties)

These properties control the basic layout of pie charts. The following code segment shows the default values:

```
pieProperties: {
   holeSize: 0,
   rotation: 0,
   skew: 0,
   label: {
         visible: false,
         font: '10pt Sans-Serif',
         color: 'black'
   },
   totalLabel: {
         visible: false,
         font: '10pt Sans-Serif',
         color: 'black'
   },
   feelerLine: {
         visible: true,
         width: 1,
         color: 'black',
         dash: ''
   },
   otherSlice: {
         threshold: undefined,
         legendLabel: 'Other',
         color: 'grey',
         border: {
               width: 1,
               color: 'transparent',
               dash: ''
         },
         showDataValues: true,
         marker: {
               shape: 'circle',
               border: {width: 0,color: 'transparent',dash: ''}
         }
   }
}
```

Also see the series-specific properties to:

- border: draw borders around pie slices
- explodeSlice: explode a slice from a pie
- deleteSlice: delete a slice from a pie
- For charts with multiple pies, use the chartsPerRow property to defined the number of pie charts to draw in a horizontal row.
- Set the dataLabels: position property to 'outside' to draw data text labels outside pie slices with feeler lines.
- Set the dataLabels: displayMode property to "%" to show slice values as a percent of the total pie.

feelerLine

When dataLabels: position is set to 'outside', this property controls the appearance of feeler lines between the pie chart and data text labels.

```
pieProperties: {
   feelerLine: {
         visible: boolean,
         width: number,
         color: 'string',
         dash: 'string'
   },
}
```

visible; true/false controls the visibility of the feeler lines. The default value is true.

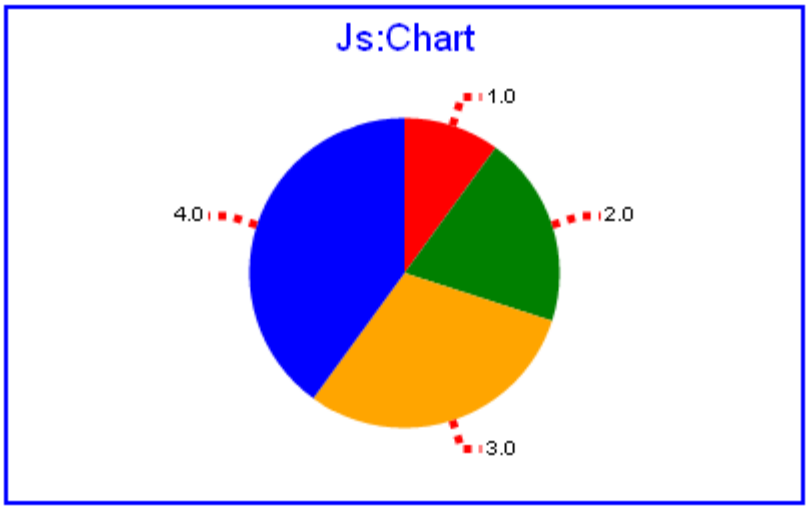
width; a number that defines the width of the feeler lines. The default value is 1.

color; a string that defines the color of the feeler lines. See http://www.w3.org/TR/css3-color/#colorunits for the format of this string. The default value is 'black'.

dash; a string that defines the dash style. The default value is '' (a solid line). Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes (e.g., dash: '1 1' draws a dotted line).

EXAMPLE:

```
pieProperties: {
   feelerLine: {visible: true,width: 4,color: 'red',dash: '4 4'}
},
dataLabels: {position: 'outside', visible: true}
```



holesize

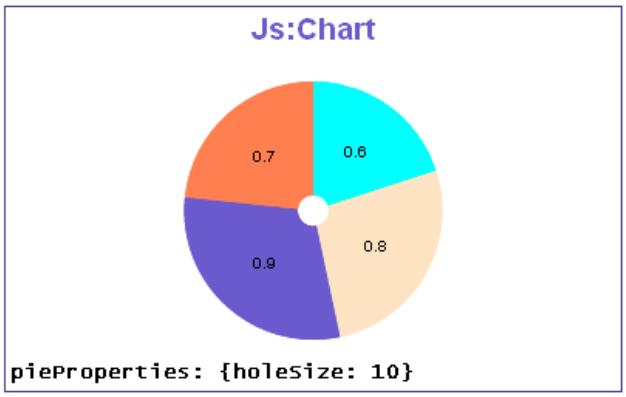
This property determines the relative size of the ring (inner circle) in a ring pie.

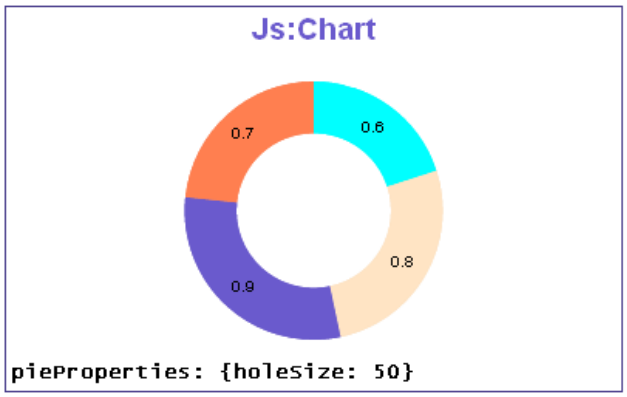
```
pieProperties: {
   holesize: Number
}
```

PARAMETERS:

holesize: 0...100 defines the relative size of the inner circle of the pie. The default value is zero.

EXAMPLE:





label

This property controls the visibility and format of the pie chart label.

```
pieProperties: {
   label: {
         visible: boolean,
         font: 'string',
         color: 'string'
   }
}
```

PARAMETERS:

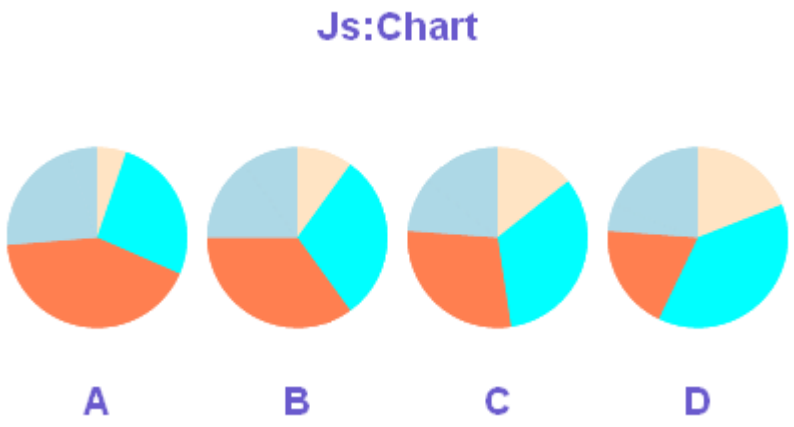
visible; true/false controls the visibility of the pie chart label. The default value is false.

font; a string that defines the size and type face of the pie chart label. See http://www.w3.org/TR/CSS2/fonts.html#font-shorthand for the format of this string. The default value is '10pt Sans-Serif'.

color; a string that defines the color of the pie chart label. See http://www.w3.org/TR/css3-color/#colorunits for the format of this string. The default value is 'black'.

EXAMPLE:

```
pieProperties: {
   label: {
         visible: true,
         font: 'Bold 14pt Sans-Serif',
         color: 'DarkSlateBlue'
   }
}
```



otherSlice

This property controls the visibility and format of pie slices that are below a specified value.

```
pieProperties: {
   otherSlice: {
         threshold: Number | 'string' | undefined,
         legendLabel: 'string',
         color: 'string' | JSON Object,
         border: {
               width: Number,
               color: 'string',
               dash: 'string'
         }
         showDataValues: boolean,
         marker: {
               shape: 'string',
               border: {
                     width: Number,
                     color: 'string',
                     dash: 'string'
               }
         }
   }
}
```

threshold: a number (absolute data value), a percent string, a 'top n' string, or undefined (disable other slice). If threshold is a number, any values below this threshold are combined into the other slice. If threshold is a percent string (e.g., '2%'), any slices whose overall contribution to the pie is less than this value are merged into the other slice. If the string begins with 'top' followed by a number (e.g., 'top 2'), slices with the highest "n" values will be drawn as individual slices. All other slices will be grouped into the other slice. If threshold is undefined or null, the other slice is disabled. The default value is undefined.

legendLabel: A string identifying the label to show in the legend for the other slice. The default value is 'Other'.

color: a color defined by a keyword or numerical specification string or a gradient defined by a string or JSON object controls the color of the other slice. The default value is grey. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string. See Gradient Definitions for the format of a linear or radial gradient.

border/width: The width of the other slice border in pixels. The default value is 1.

border/color: a color defined by a keyword or numerical specification string controls the color of the other slice border. The default value is 'transparent'.

border/dash: a string that defines the border's dash style. The default value is '' (a solid line). Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes (e.g., dash: '1 1' draws a dotted line).

showDataValues: true/false = show/hide a data text label for the other slice. The default value is true.

marker/shape: a string that defines the shape of the marker to show in the legend area for the other slice: arrow, bar, circle, cross, diamond, fiveStar, hexagon, hourglass, house, pirateCross, plus, sixStar, square, thinPlus, tick, or triangle. The default value is 'circle'. Note that bar, cross, and tick markers require a border width and color.

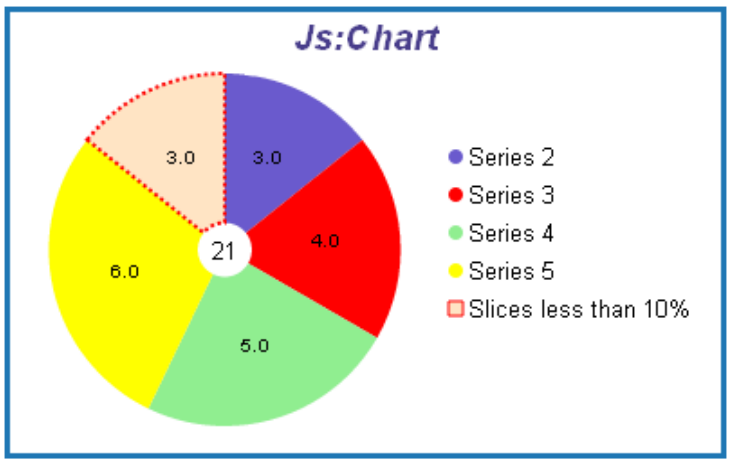
marker/border/width: The width of the marker border in pixels. The default value is 1.

marker/border/color: a color defined by a keyword or numerical specification string or a gradient defined by a string or JSON object. The default value is 'transparent'.

marker/border/dash: a string that defines the border's dash style. The default value is '' (a solid line). Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes (e.g., dash: '1 1' draws a dotted line).

EXAMPLE:

```
data = [[1,2,3,4,5,6]],
chartType: 'pie',
pieProperties: {
   holeSize: 15,
   totalLabel: {visible: true},
   otherSlice: {
         threshold: '10%',
         legendLabel: 'Slices less than 10%',
         color: 'bisque',
         border: {width: 2,color: 'red',dash: '2 2'},
         showDataValues: true,
         marker: {
               shape: 'square',
               border: {width: 1,color: 'red',dash: ''}
         }
   }
},
legend: {visible: true},
series: [
   {series: 0, color: 'cyan'},
   {series: 1, color: 'pink'},
   {series: 2, color: 'slateblue'},
   {series: 3, color: 'red'},
   {series: 4, color: 'lightgreen'},
   {series: 5, color: 'yellow'},
   ]
```



rotation

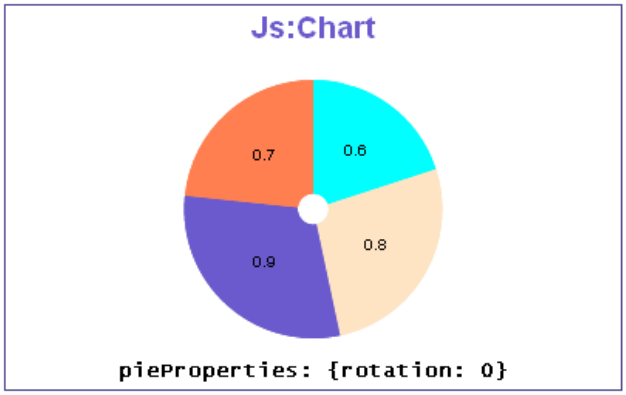
This property rotates a pie chart a specified number of degrees.

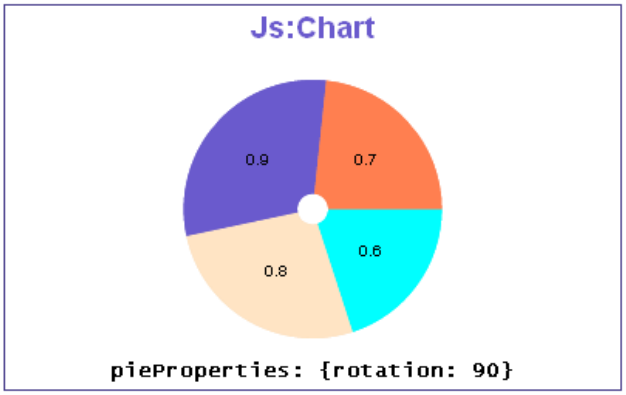
```
pieProperties: {
   rotation: Number
}
```

PARAMETERS:

rotation: 0...359. The default value is zero.

EXAMPLE:





skew

When depth is applied to a pie chart, this property controls the tile of the pie chart.

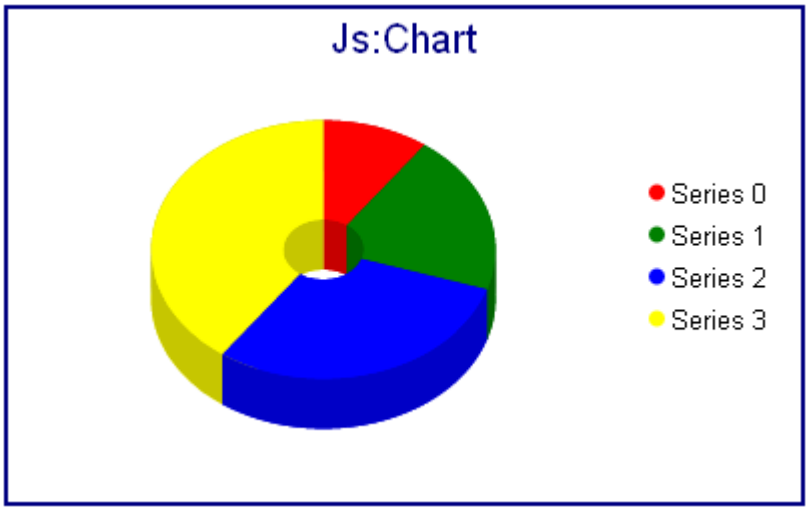
```
pieProperties: {
   skew: Number
}
```

PARAMETERS:

skew: 0...100. The default value is zero.

EXAMPLE:

```
depth: 25,
pieProperties: {holeSize: 20, skew: 25},
```



totalLabel

This property controls the visibility and format of a pie chart's total label (i.e., the total value displayed in the center of the chart).

```
pieProperties: {
   totalLabel: {
         visible: boolean,
         font: 'string',
         color: 'string'
   }
}
```

PARAMETERS:

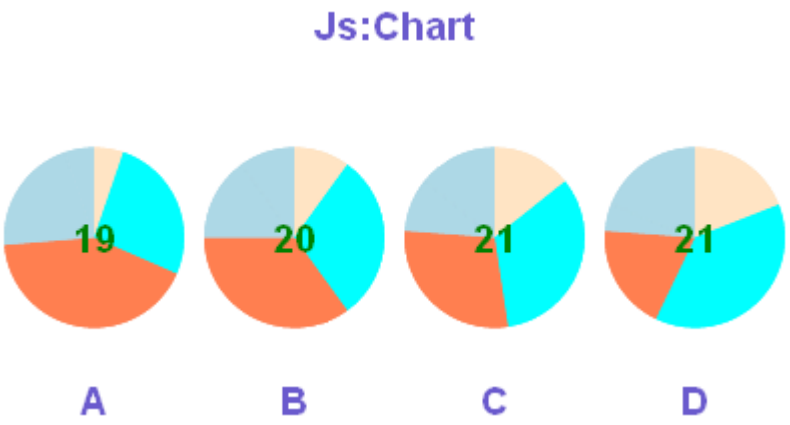
visible; true/false controls the visibility of the total label. The default value is false.

font; a string that defines the size and type face of the total label. See http://www.w3.org/TR/CSS2/fonts.html#font-shorthand for the format of this string. The default value is '10pt Sans-Serif'.

color; a string that defines the color of the total label. See http://www.w3.org/TR/css3-color/#colorunits for the format of this string. The default value is 'black'.

EXAMPLE:

```
pieProperties: {
   totalLabel: {
         visible: true,
         font: 'Bold 14pt Sans-Serif',
         color: 'Green'
   }
}
```



Polar Charts (polarProperties)

These properties control the general appearance of polar and radar charts.

```
polarProperties: {
   straightGridLines: boolean,
   extrudeAxisLabels: boolean,
   drawAsArea: boolean
},
```

PARAMETERS:

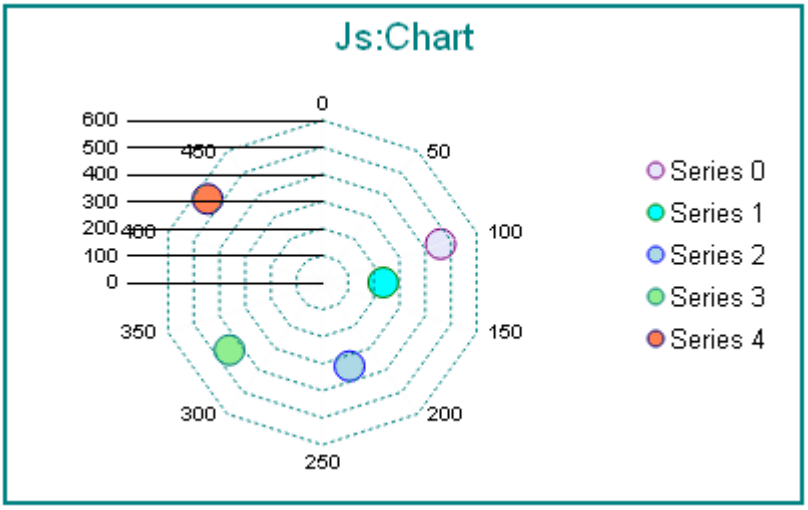
straightGridLines: true = draw straight grid lines with yaxis:majorGrid, false = draw round grid lines with yaxis: majorGrid. The default value is false.

extrudeAxisLabels: true = draw Y-axis labels extruded from the circular grid, false = draw Y-axis labels within the circular grid. The default value is false.

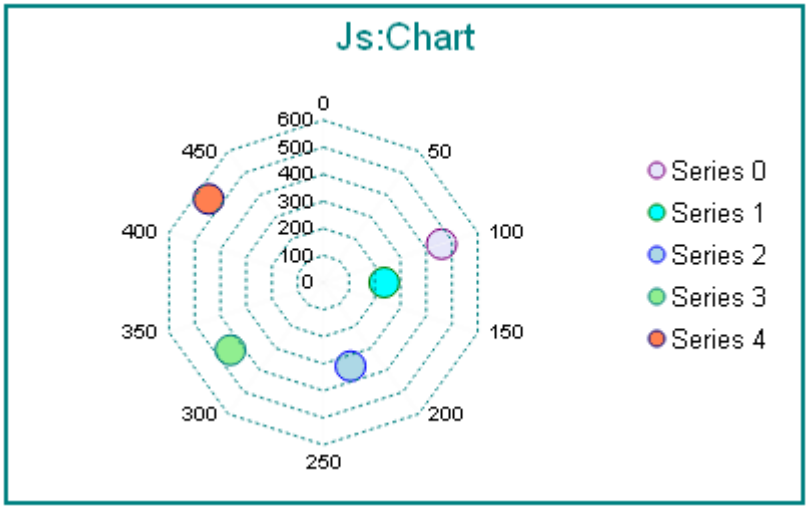
drawAsArea: for radar charts only, true = draw a circular area chart, false = draw a circular line chart. The default value is false.

EXAMPLES:

```
data: [[[100,460]],[[125,225]],[[225,325]],[[325,425]],[[425,525]]],
chartType: 'polar',
legend: {visible: true},
polarProperties: {
   straightGridLines: true,
   extrudeAxisLabels: true
},
yaxis: {
   majorGrid: {
   lineStyle: {width: 1,color: 'teal',dash: '2 2'},
   }
},
series: [
{series: 0, color: 'lavender', marker:{size: 15, shape: 'circle',
border: {width: 1, color: 'purple'}}},
{series: 1, color: 'cyan', marker:{size: 15, shape: 'circle',
border: {width: 1, color: 'green'}}},
{series: 2, color: 'lightblue', marker:{size: 15, shape: 'circle',
border: {width: 1, color: 'blue'}}},
{series: 3, color: 'lightgreen', marker:{size: 15, shape: 'circle',
border: {width: 1, color: 'teal'}}},
{series: 4, color: 'coral', marker:{size: 15, shape: 'circle',
border: {width: 1, color: 'navy'}}},
],
```




```
polarProperties: {
   straightGridLines: true,
   extrudeAxisLabels: false
},
```



NOTES:

- A polar chart is a circular scatter chart.
- Like scatter charts, a polar chart requires two values to draw each marker.
- yaxis properties control the circular grid around polar chart markers. These properties also control the appearance of axis labels and gridlines.
- xaxisNumeric properties control the appearance of labels and values on the X-axis (around the outside edge of the circular grid) and X-axis gridlines.

Stock Charts (stockProperties)

A high/low/open/close stock chart requires an array of 4 numbers for each riser: [high, low, open, close]. The first two numbers (high/low) define the high/low line that draws up and down from the box. The second two numbers (open/close) draw the open/close box riser. If either high or low is null, the high-low line is not drawn. If either open or close is null, the open-close box is not drawn. These properties control the general appearance of a stock chart.

```
stockProperties: {
   startTime: 'string' | undefined,
   stopTime: 'string' | undefined,
   interval: 'string' | undefined,
   labelFormat: 'string' | undefined,
   upRiserColor: 'string' | JSON Object,
   downRiserColor: 'string' | JSON Object,
   hiLowLine: {
         width: Number,
         color: 'string' | JSON Object,
         dash: 'string'
   }
},
```

PARAMETERS:

startTime: a string identifying the start time. It can be almost any kind of string denoting time, mixing hours (xx:xx), days of the week, dates, months and years. Examples: "Mon", "1 July 20", "June 2001", "8:00", "6/10/01", "Thu, 4 Apr 1976 14:30", "2 0:00" (February 1), "1 1 0:00" (January 1). The default value is undefined.

stopTime: a string identifying the start time. As with startTime, the string can be almost any date/time format. The default value is undefined.

interval: a string that specifies the time interval: 'seconds', 'minutes', 'hours', 'days', 'weeks', 'months', 'quarters', or 'years'. This property defines the time gap between each group label. If undefined, the library will assign an interval based on the startTime, stopTime, and the number of groups. The default value is undefined.

labelFormat: a string that defines how a time/date number is converted into a label. If undefined, the library will we use a default formats based on the interval. See http://code.google.com/p/datejs/wiki/FormatSpecifiers for available format options. The default value is undefined.

upRiserColor: a color defined by a keyword or numerical specification string or a gradient defined by a string or JSON object. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string. See Gradient Definitions for the format of a linear or radial gradient. The default value is '#77b39a'.

downRiserColor: a color defined by a keyword or numerical specification string or a gradient defined by a string or JSON object. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string. See Gradient Definitions for the format of a linear or radial gradient. The default value is '#e2675b'.

hiLowLine/width: a number that defines the width of the line in pixels. The default value is 1.

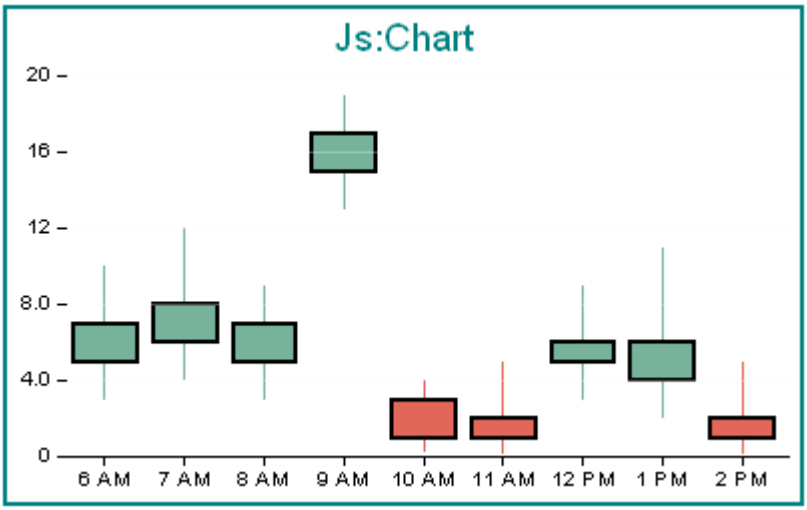
hiLowLine/color: a color defined by a keyword or numerical specification string. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string. If a color is not specified, the high-low line for each riser will use the color

assigned to the upRiserColor and downRiserColor. This property colors the high low line for all risers (up and down). The default value is undefined.

hiLowLine/dash: a string that defines the line's dash style. The default value is '' (a solid line). Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes (e.g., dash: '1 1' draws a dotted line).

EXAMPLE:

```
data: [
[[10,3,5,7],[12,4,6,8],[9,3,5,7],[19,13,15,17],[4,.2,3,1],[5,.1,2,1],
[9,3,5,6],[11,2,4,6],[5,.1,2,1]]
],
chartType: 'stock',
blaProperties: {orientation: 'vertical'},
stockProperties: {
   startTime: '6A',
   stopTime: '2P',
   interval: 'hours',
   labelFormat: 'h tt'
   },
```



NOTES:

- A high/low/open/close stock chart requires an array of 4 numbers for each riser: [high, low, open, close].
- The first two numbers (high/low) define the high/low line that draws up and down from the box. The second two numbers (open/close) draw the open/close box riser.
- If either high or low is null, the high-low line is not drawn.
- If either open or close is null, the open-close box is not drawn.
- Use stockProperties to control the general appearance of a stock chart

Waterfall Charts (waterfallProperties)

Waterfall charts graphically illustrate the cumulative effect of sequentially introducing positive or negative values to an initial value. The initial and final (or total) values are represented by whole columns drawn from the ordinal axis baseline. Intermediate positive and negative values are drawn as floating columns.

The following properties control the general layout of waterfall charts. This code segment shows the default settings from properties.js.

```
waterfallProperties: {
   appendTotalRiser: true,
   positiveRiserColor: '#77b39a',
   negativeRiserColor: '#e2675b',
   zeroRiserColor: '#7593bd',
   otherRiserColor: '#aaaaaa',
   subtotalRisers: [],
   otherRisers: [],
   connectorLine: {
         width: 1,
         color: 'black',
         dash: ''
   }
},
```

appendTotalRiser

This property controls the appearance of a total riser as the last riser in a waterfall chart.

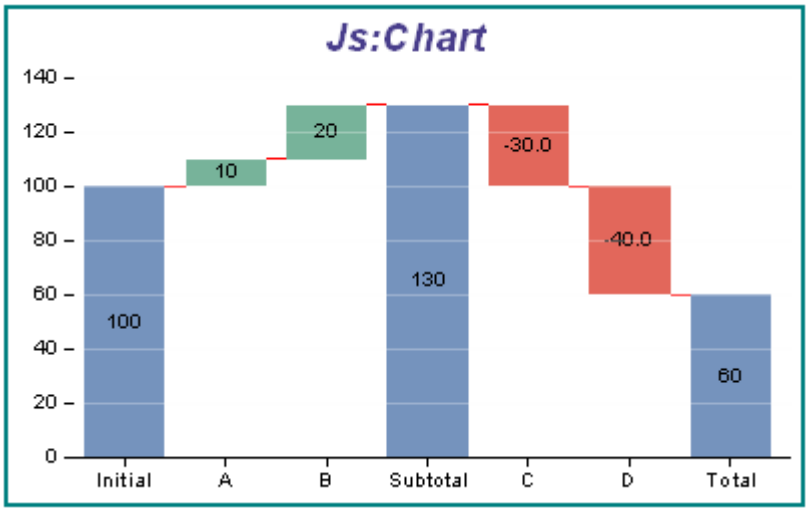
```
waterfallProperties: {
   appendTotalRiser: boolean,
},
```

PARAMETERS:

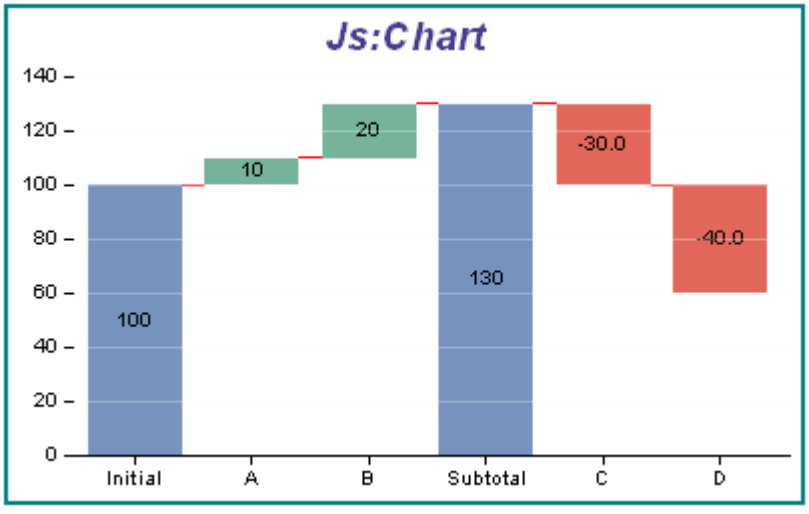
appendTotalRiser: true = draw total riser, false = do not draw total riser. The default value is true.

EXAMPLES:

```
data: [[100,10, 20, 130, -30, -40]],
chartType: 'waterfall',
blaProperties: {orientation: 'vertical'},
groupLabels: ['Initial', 'A', 'B', 'Subtotal', 'C',  'D','Total'],
waterfallProperties: {appendTotalRiser: true}
```




```
waterfallProperties: {appendTotalRiser: false}
```



connectorLine

These properties control the appearance of connector lines between risers in a waterfall chart:

```
waterfallProperties: {
   connectorLine: {
         width: number,
         color: 'string',
         dash: 'string'
   }
},
```

PARAMETERS:

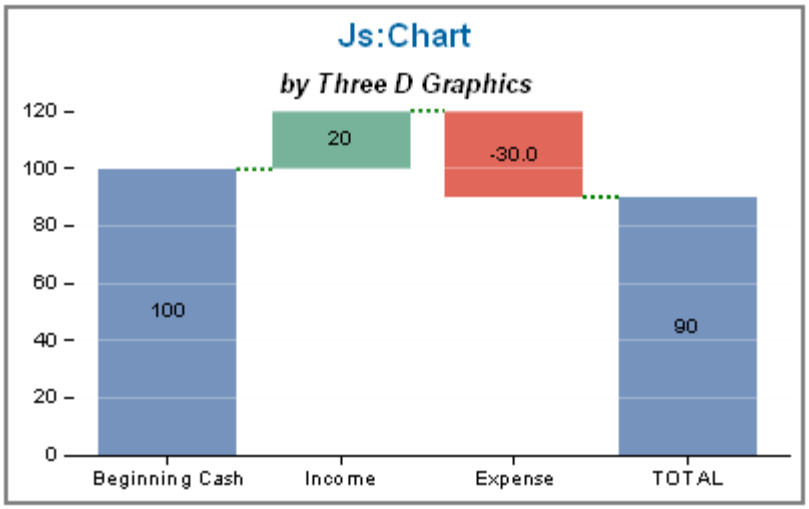
width: The width of the connector line in pixels. The default value is 1.

color: a color defined by a keyword or numerical specification string. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string. The default value is black.

dash: a string that defines the line's dash style. The default value is '' (a solid line). Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes (e.g., dash: '1 1' draws a dotted line).

EXAMPLE:

```
chartType: 'waterfall',
blaProperties: {orientation: 'vertical'},
waterfallProperties: {
   connectorLine: {width: 2, color: 'green', dash: '2 2'},
},
groupLabels: ["Beginning Cash", "Income", "Expense","TOTAL"]
```



negativeRiserColor

This property controls the color of risers that represent negative values in a waterfall chart.

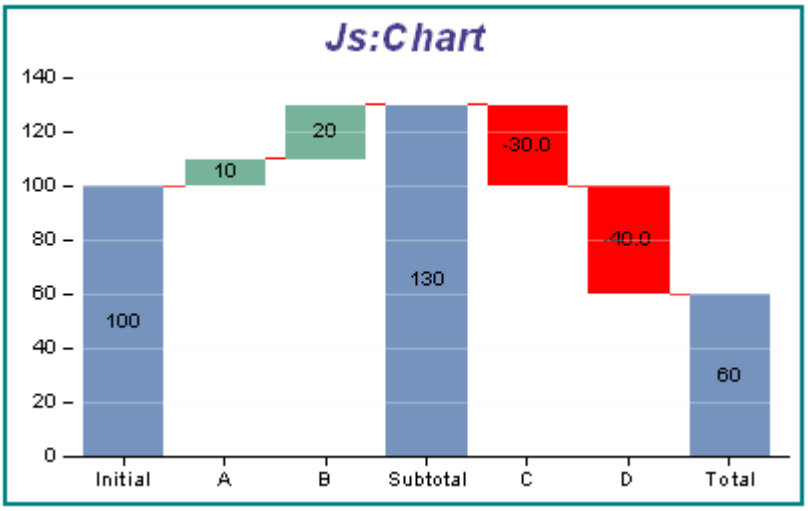
```
waterfallProperties: {
   negativeRiserColor: 'string' | JSON Object
},
```

PARAMETERS:

negativeRiserColor: a color defined by a keyword or numerical specification string or a gradient defined by a string or JSON object. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string. See Gradient Definitions for the format of a linear or radial gradient. The default value is #e2675b.

EXAMPLES:

```
data: [[100,10, 20, 130, -30, -40]],
chartType: 'waterfall',
blaProperties: {orientation: 'vertical'},
groupLabels: ['Initial', 'A', 'B', 'Subtotal', 'C',  'D','Total'],
waterfallProperties: {
   negativeRiserColor: 'red'
}
```



NOTES:

You can also use series-specific properties to format these risers individually. Example:

```
series: [
   {series: 0, group: 4, color: 'coral'},
   {series: 0, group: 5, color: 'pink'},
]
```

otherRiserColor

This property controls the color of risers that represent "other" values (if any) in a waterfall chart. Other values are defined in the otherRisers[] array.

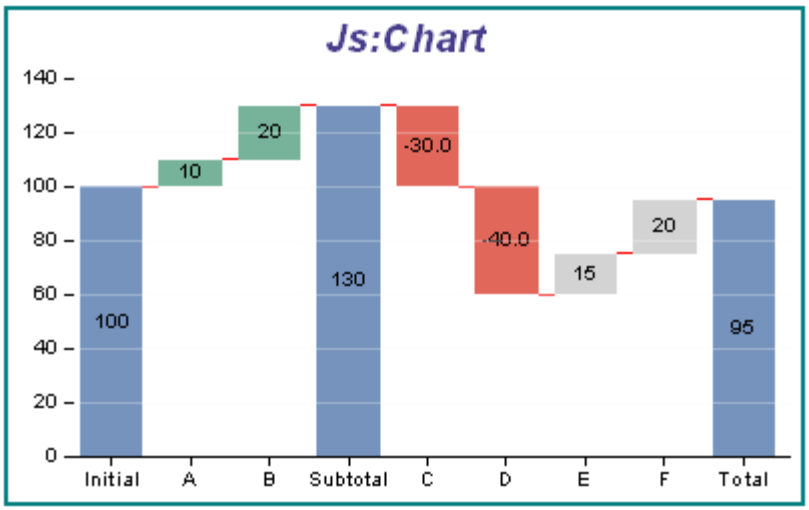
```
waterfallProperties: {
   otherRiserColor: 'string' | JSON object
},
```

PARAMETERS:

otherRiserColor: a color defined by a keyword or numerical specification string or a gradient defined by a string or JSON object. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string. See Gradient Definitions for the format of a linear or radial gradient. The default value is '#aaaaaa' (grey).

EXAMPLE:

```
data: [[100,10, 20, 130, -30, -40, 15, 20]],
chartType: 'waterfall',
blaProperties: {orientation: 'vertical'},
groupLabels: ['Initial', 'A', 'B', 'Subtotal', 'C',  'D', 'E',
'F','Total'],
waterfallProperties: {
   otherRiserColor: 'lightgrey',
   otherRisers: ['E', 'F'],
}
```



NOTES:

You can also use series-specific properties to format these risers individually. Example:

```
series: [
   {series: 0, group: 6, color: 'lightgreen'},
   {series: 0, group: 7, color: 'cyan'},
]
```

otherRisers[]

This property defines one or more groups/values to be interpreted as "other" values. Risers representing these values will be assigned the color defined by the otherRiserColor property.

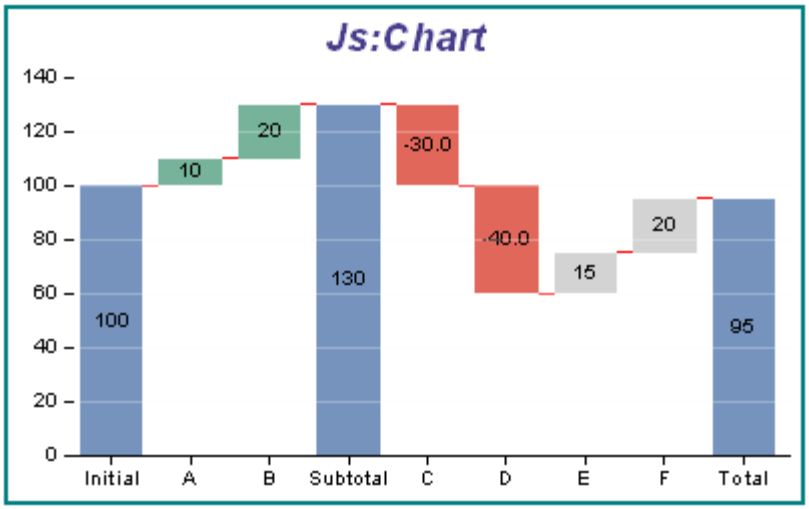
```
waterfallProperties: {
   otherRisers: [number | 'string', ... number | 'string'],
},
```

PARAMETERS:

otherRisers: An array of numbers (matching group indices), or strings (matching group names).

EXAMPLE:

```
data: [[100,10, 20, 130, -30, -40, 15, 20]],
chartType: 'waterfall',
blaProperties: {orientation: 'vertical'},
groupLabels: ['Initial', 'A', 'B', 'Subtotal', 'C',  'D', 'E',
'F','Total'],
waterfallProperties: {
   otherRiserColor: 'lightgrey',
   otherRisers: ['E', 'F'],
}
```



positiveRiserColor

This property controls the color of risers that represent positive values in a waterfall chart.

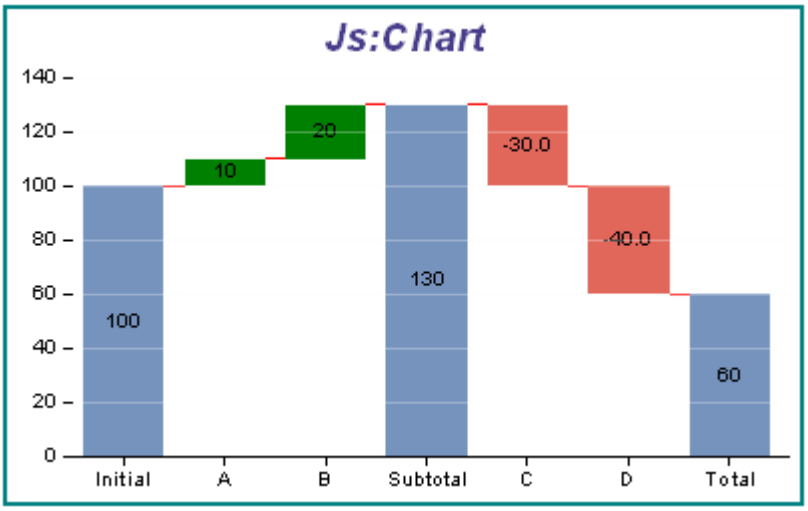
```
waterfallProperties: {
   positiveRiserColor: 'string' | JSON object,
},
```

PARAMETERS:

positiveRiserColor: a color defined by a keyword or numerical specification string or a gradient defined by a string or JSON object. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string. See Gradient Definitions for the format of a linear or radial gradient The default value is '#77b39a'.

EXAMPLES:

```
data: [[100,10, 20, 130, -30, -40]],
chartType: 'waterfall',
blaProperties: {orientation: 'vertical'},
groupLabels: ['Initial', 'A', 'B', 'Subtotal', 'C',  'D','Total'],
waterfallProperties: {
   positiveRiserColor: 'green'
}
```



NOTES:

You can also use series-specific properties to format these risers individually. Example:

```
series: [
   {series: 0, group: 1, color: 'limegreen'},
   {series: 0, group: 2, color: 'lightgreen'},
]
```

subtotalRisers[]

This property defines one or more groups/values to be interpreted as "subtotal" values.

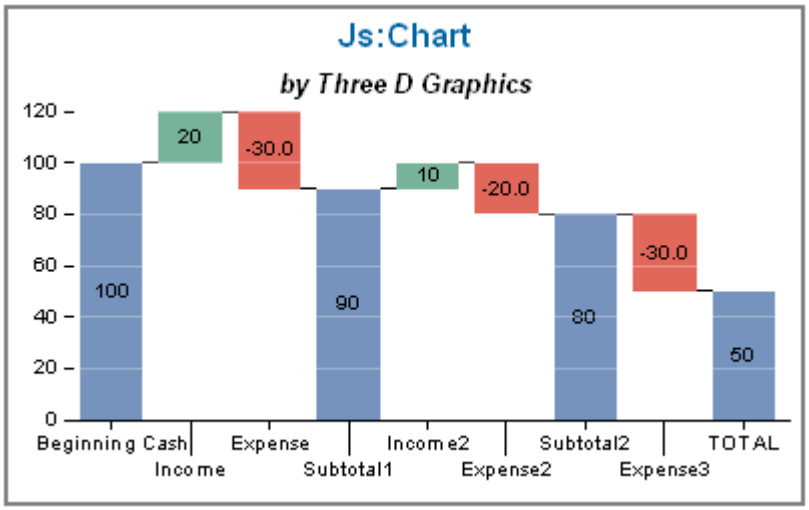
```
waterfallProperties: {
   subtotalRisers: []
},
```

PARAMETERS:

subtotalRisers: An array of numbers (matching group indices), or strings (matching group names).

EXAMPLES:

```
data: [[100,20,-30,90,10,-20,80,-30]],
chartType: 'waterfall',
blaProperties: {orientation: 'vertical'},
waterfallProperties: {
   subtotalRisers: ['Subtotal1','Subtotal2'],
},
groupLabels: ["Beginning Cash", "Income", "Expense",
"Subtotal1","Income2","Expense2","Subtotal2","Expense3","TOTAL"]
```



zeroRiserColor

This property controls the color of risers that represent the initial value, subtotal values (if any) and the total value in a waterfall chart.

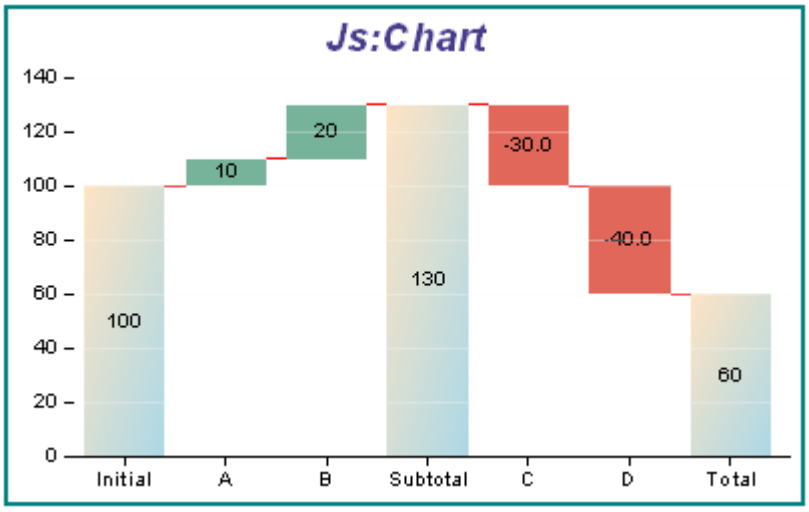
```
waterfallProperties: {
   zeroRiserColor: 'string' | JSON object
},
```

PARAMETERS:

zeroRiserColor: a color defined by a keyword or numerical specification string or a gradient defined by a string or JSON object. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string. See Gradient Definitions for the format of a linear or radial gradient. The default value is '#7593bd' (blue).

EXAMPLE:

```
waterfallProperties: {
   zeroRiserColor: {
         type: 'linear',
         start: {x: '0%', y: '0%'},end: {x: '100%', y: '100%'},
         stops: [[0, 'bisque'],[1, 'lightblue']]
   }
}
```



NOTES:

You can also use series-specific properties to format these risers individually. Example:

```
series: [
   {series: 0, group: 0, color: 'lightblue', border: {width: 2,
color: 'blue'}},
   {series: 0, group: 3, color: 'cyan', border: {width: 2, color:
'teal'}},
   {series: 0, group: 6, color: 'turquoise', border: {width: 2,
color: 'green'}},
]
```

Section 5: Special Features

Error Bars (errorBars)

These properties define error bars that can be drawn on bar risers, line risers, area risers, bubble markers, and scatter markers to visualize where risers are above/below expected values

```
errorBars: {
   yData: undefined,
   xData: undefined,
   hatWidth: '50%',
   line: {
         color: 'black',
         width: 1,
         dash: ''
   },
   marker: {
         color: 'grey',
         shape: 'diamond',
         rotation: 0,
         border: {
               width: 1,
               color: 'black',
               dash: ''
         }
   }
},
```

PARAMETERS:

yData: undefined for no error bars on Y-axis data or one or more arrays defining high, marker position (optional), and low values for each riser. Each array can be: [highValue,lowValue] or [highValue, markerPosition, lowValue].

xData: undefined for no error bars on X-axis data or one or more arrays defining high, marker position (optional), and low values for each riser. Each array can be: [highValue,lowValue] or [highValue, markerPosition, lowValue].

hatWidth: a number in pixels or a string that expresses a percentage of the width of the riser.

line/color: a color defined by a keyword or numerical specification string that defines the color of the line and hats. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string. The default value is black.

line/width: a number that defines the width of the error bar line. The default value is 1.

line/dash: a string that defines the line's dash style. Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes (e.g., dash: '1 1' draws a dotted line). Use '' for a solid line.

marker/color: a color defined by a keyword or numerical specification string or a gradient defined by a string or JSON object. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string. See Gradient Definitions for the format of a linear or radial gradient.

marker/shape: a string that defines the shape of the marker: arrow, bar, circle, cross, diamond, fiveStar, hexagon, hourglass, house, pirateCross, plus, sixStar, square, thinPlus, tick, or triangle. The default value is 'diamond'. Note that bar, cross, and tick markers require a border width and color.

marker/rotation: a number 0...360 that defines the angle (in degrees) of the marker. The default value is zero.

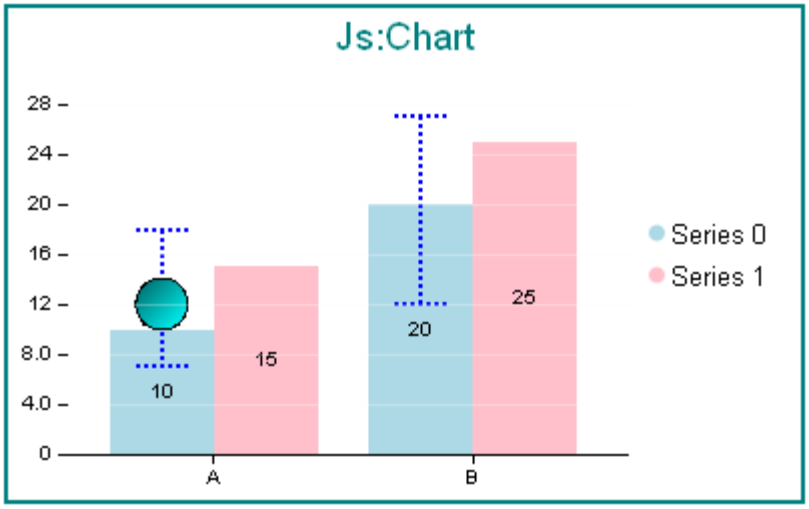
marker/border/width: a number defines the width of the border in pixels.

marker/border/color: a color defined by a keyword or numerical specification string. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string. The default value is 'black'.

marker/border/dash: a string that defines the border dash style. The default value is '' (a solid line). Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes.

EXAMPLE:

```
blaProperties: {orientation: 'vertical'},
legend: {visible: true},
errorBars: {
hatWidth: '50%',
   yData: [[[7,12,18],[12,27]]],
   line: {
         color: 'blue',
         width: 2,
         dash: '2 2'
   },
   marker: {
         color: 'linear-gradient(0,0,100%,100%, 20% teal, 95% cyan)',
         shape: 'circle'
         }
},
series: [
{series: 0, color: 'lightblue'},
{series: 1, color: 'pink'},
]
```



Reference Lines (referenceLines)

These properties define one or more reference lines to draw on an axis.

```
referenceLines: [
   {
   value: number | 'string',
   axis: 'string',
   line: {
         color: 'string',
         width: number,
         dash: 'string'
   },
   label: {
         text: 'string',
         font: 'string',
         color: 'string'
   },
   anchor: 'string',
   showValue: boolean
   }
]
```

PARAMETERS:

value: a number or string that defines where on the axis to draw the line. For a numeric axis, the number must be a value that is visible on the axis and a string must be a percentage value in the range '0%'...'100%'. For the ordinal axis, a string must be a group label that is visible on the axis and a number must be in the range 0...1 (e.g., .5 will draw the reference line in the center of the chart).

axis: a string that defines the axis on which to draw the line: 'x', 'y', 'y2' or undefined (do not draw line).

line/color: a color defined by a keyword or numerical specification string that defines the color of the line. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string.

line/width: a number that defines the width of the line.

line/dash: a string that defines the line's dash style. Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes (e.g., dash: '1 1' draws a dotted line). Use '' for a solid line.

label/text: (optional) a string that defines the label to draw beside a reference line. Use '' or undefined Empty string, or undefined, means draw no label

label/font; a string that defines the size and type face of the label. See http://www.w3.org/TR/CSS2/fonts.html#font-shorthand for the format of this string.

label/color: a color defined by a keyword or numerical specification string that defines the color of the label. See http://www.w3.org/TR/css3-color/#colorunits for the format of a color string.

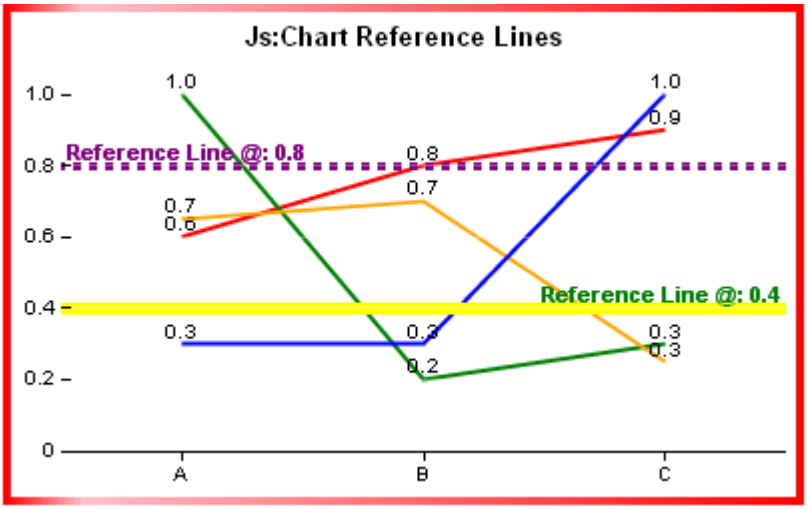
anchor: a string that defines where to draw the reference label relative to the line: 'start' (same side as axis labels) or 'end' (opposite side)

showValue: true/false: true = draw *value* at the specified *anchor* location relative to the line. It will be appended to the label (if defined). false = do not draw label..

EXAMPLES:

```
chartType: 'line',
```

```
blaProperties: {orientation: 'vertical', seriesLayout: 'absolute'},
title: {text: 'Js:Chart Reference Lines'},
referenceLines: [
   {
         value: 0.4,
         axis: 'y',
         line: {
               color: 'yellow',
               width: 6,
               dash: ''
         },
         label: {
               text: 'Reference Line @:',
               font: 'bold 8pt Sans-Serif',
               color: 'green'
         },
         anchor: 'end',
         showValue: true
   },
   {
         value: 0.8,
         axis: 'y',
         line: {
               color: 'purple',
               width: 4,
               dash: '4 4'
         },
         label: {
               text: 'Reference Line @: ',
               font: 'bold 8pt Sans-Serif',
               color: 'purple'
         },
         anchor: 'start',
         showValue: true
   },
]
```



Reader Comments

In an ongoing effort to produce effective documentation, the Documentation Services staff at Information Builders welcomes any opinion you can offer regarding this manual.

Please use this form to relay suggestions for improving this publication or to alert us to corrections. Identify specific pages where applicable. You can contact us through the following methods:

Mail:	Documentation Services - Customer Support Information Builders, Inc. Two Penn Plaza New York, NY 10121-2898
Fax:	(212) 967-0460
E-mail:	*books_info@ibi.com*
Web form:	*http://www.informationbuilders.com/bookstore/derf.html*

Name:_____

Company:_____

Address:_____

Telephone:_____ Date:_____

Email:_____

Comments:

Information Builders

WebFOCUS

JavaScript Charting Engine API Guide

Information Builders
Two Penn Plaza
New York, NY 10121-2898

Printed on recycled paper in the U.S.A.