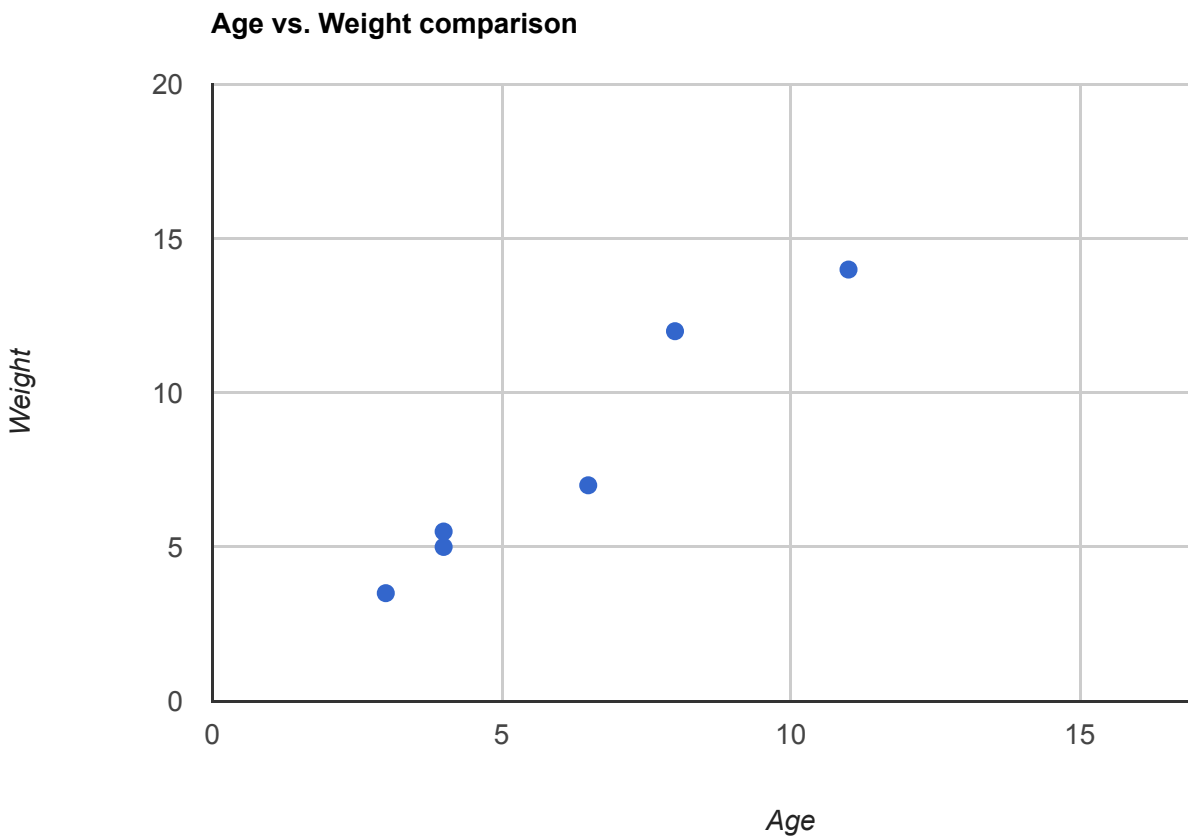
Visualization: Scatter Chart

Overview

Scatter charts plot points on a graph. When the user hovers over the points, tooltips are displayed with more information.

Google scatter charts are rendered within the browser using SVG (http://www.w3.org/Graphics/SVG/) or VML (http://en.wikipedia.org/wiki/Vector_Markup_Language) depending on browser capabilities.

Example



CODE IT YOURSELF ON JSFIDDLE

```
<html>
  <head>
```

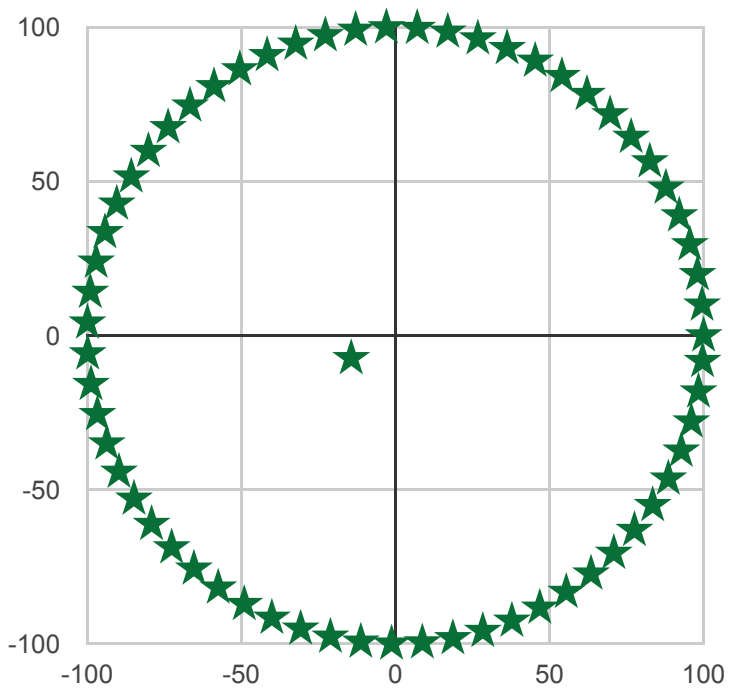
```
    <script type="text/javascript" src="https://www.gstatic.com/charts/loader
    <script type="text/javascript">
      google.charts.load('current', {'packages':['corechart']});
      google.charts.setOnLoadCallback(drawChart);

      function drawChart() {
        var data = google.visualization.arrayToDataTable([
          ['Age', 'Weight'],
          [ 8,      12],
          [ 4,      5.5],
          [ 11,     14],
          [ 4,      5],
          [ 3,      3.5],
          [ 6.5,    7]
        ]);

        var options = {
          title: 'Age vs. Weight comparison',
          hAxis: {title: 'Age', minValue: 0, maxValue: 15},
          vAxis: {title: 'Weight', minValue: 0, maxValue: 15},
          legend: 'none'
        };

        var chart = new google.visualization.ScatterChart(document.getElementI

        chart.draw(data, options);
      }
    </script>
  </head>
  <body>
    <div id="chart_div" style="width: 900px; height: 500px;"></div>
  </body>
</html>
```

Changing and animating shapes

By default, scatter charts represent the elements of your dataset with circles. You can specify other shapes with the **`pointShape`** option, detailed in the Customizing Points (https://developers.google.com/chart/interactive/docs/points) documentation.

As with most other Google Charts, you can animate them using events (https://developers.google.com/chart/interactive/docs/events). You can add an event listener for the first **`ready`** event and redraw the chart after making the desired modifications. After the first **`ready`** event, you can listen to the **`animationfinish`** event to repeat the process,

resulting in a continuous animation. The `animation` option controls how the redraw occurs: immediately (no animation) or smoothly, and if smoothly how quickly and with what behavior.



GOOD PARTS FULL HTML

```
<html>
<head>
  <script type="text/javascript" src="https://www.gstatic.com/charts/loader.
  <script type="text/javascript">
    google.charts.load("current", {packages:["corechart"]});
    google.charts.setOnLoadCallback(drawChart);

    function drawChart() {
      var data = new google.visualization.DataTable();
      data.addColumn('number');
      data.addColumn('number');

      var radius = 100;
      for (var i = 0; i < 6.28; i += 0.1) {
        data.addRow([radius * Math.cos(i), radius * Math.sin(i)]);
```

```
      }

      // Our central point, which will jiggle.
      data.addRow([0, 0]);

      var options = {
        legend: 'none',
        colors: ['#087037'],
        pointShape: 'star',
        pointSize: 18,
        animation: {
          duration: 200,
          easing: 'inAndOut',
        }
      };

      var chart = new google.visualization.ScatterChart(document.getElementB

      // Start the animation by listening to the first 'ready' event.
      google.visualization.events.addOneTimeListener(chart, 'ready', randomW

      // Control all other animations by listening to the 'animationfinish' 
      google.visualization.events.addListener(chart, 'animationfinish', rand

      chart.draw(data, options);

      function randomWalk() {
        var x = data.getValue(data.getNumberOfRows() - 1, 0);
        var y = data.getValue(data.getNumberOfRows() - 1, 1);
        x += 5 * (Math.random() - 0.5);
        y += 5 * (Math.random() - 0.5);
        if (x * x + y * y > radius * radius) {
          // Out of bounds. Bump toward center.
          x += Math.random() * ((x < 0) ? 5 : -5);
          y += Math.random() * ((y < 0) ? 5 : -5);
        }
        data.setValue(data.getNumberOfRows() - 1, 0, x);
        data.setValue(data.getNumberOfRows() - 1, 1, y);
        chart.draw(data, options);
      }
    }
  </script>
  </head>
  <body>
    <div id="animatedshapes_div" style="width: 500px; height: 500px;"></div>
  </body>
</html>
```

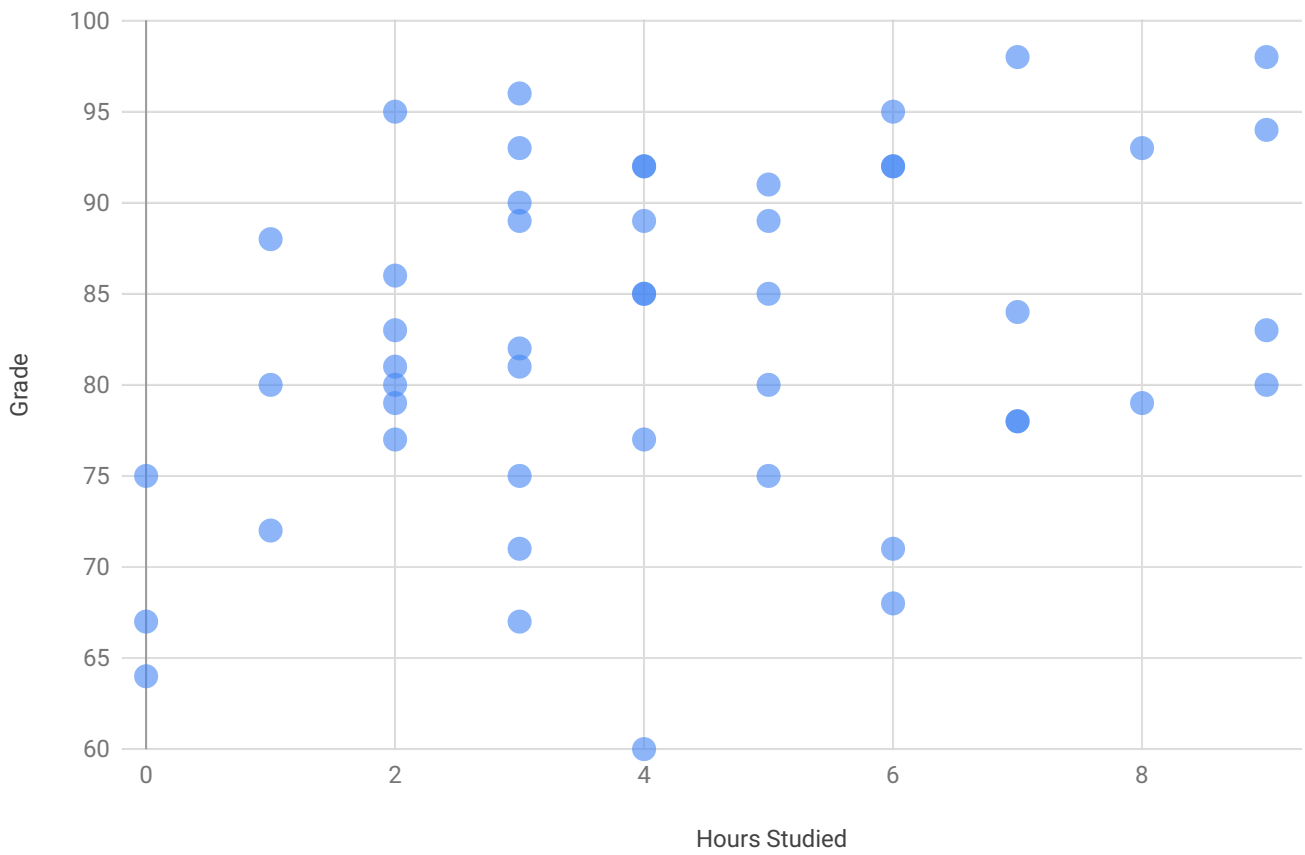
Creating Material scatter charts

In 2014, Google announced guidelines intended to support a common look and feel across its properties and apps (such as Android apps) that run on Google platforms. We call this effort *Material Design*. We'll be providing "Material" versions of all our core charts; you're welcome to use them if you like how they look.

Creating a Material Scatter Chart is similar to creating what we'll now call a "Classic" Scatter Chart. You load the Google Visualization API (although with the `'scatter'` package instead of the `'corechart'` package), define your datatable, and then create an object (but of class `google.charts.Scatter` instead of `google.visualization.ScatterChart`).

Note: Material Charts will not work in old versions of Internet Explorer. (IE8 and earlier versions don't support SVG, which Material Charts require.)

Students' Final Grades
based on hours studied



CODE IT YOURSELF ON JSFIDDLE

Material Scatter Charts have many small improvements over Classic Scatter Charts, including variable opacity for legibility of overlapping points, an improved color palette, clearer label formatting, tighter default spacing, softer gridlines and titles (and the addition of subtitles).

```
google.charts.load('current', {'packages':['scatter']});
google.charts.setOnLoadCallback(drawChart);

function drawChart () {

  var data = new google.visualization.DataTable();
  data.addColumn('number', 'Hours Studied');
  data.addColumn('number', 'Final');

  data.addRows([
    [0, 67], [1, 88], [2, 77],
    [3, 93], [4, 85], [5, 91],
    [6, 71], [7, 78], [8, 93],
    [9, 80], [10, 82],[0, 75],
    [5, 80], [3, 90], [1, 72],
    [5, 75], [6, 68], [7, 98],
    [3, 82], [9, 94], [2, 79],
    [2, 95], [2, 86], [3, 67],
    [4, 60], [2, 80], [6, 92],
    [2, 81], [8, 79], [9, 83],
    [3, 75], [1, 80], [3, 71],
    [3, 89], [4, 92], [5, 85],
    [6, 92], [7, 78], [6, 95],
    [3, 81], [0, 64], [4, 85],
    [2, 83], [3, 96], [4, 77],
    [5, 89], [4, 89], [7, 84],
    [4, 92], [9, 98]
  ]);

  var options = {
    width: 800,
    height: 500,
    chart: {
      title: 'Students\' Final Grades',
      subtitle: 'based on hours studied'
    },
    hAxis: {title: 'Hours Studied'},
    vAxis: {title: 'Grade'}
  };
```

```
    var chart = new google.charts.Scatter(document.getElementById('scatte

    chart.draw(data, google.charts.Scatter.convertOptions(options));
  }
```

The Material Charts are in **beta**. The appearance and interactivity are largely final, but many of the options available in Classic Charts are not yet available in them. You can find a list of options that are not yet supported in this issue (https://github.com/google/google-visualization-issues/issues/2143).

Also, the way options are declared is not finalized, so you must convert your options by replacing this line:

```
chart.draw(data, options);
```

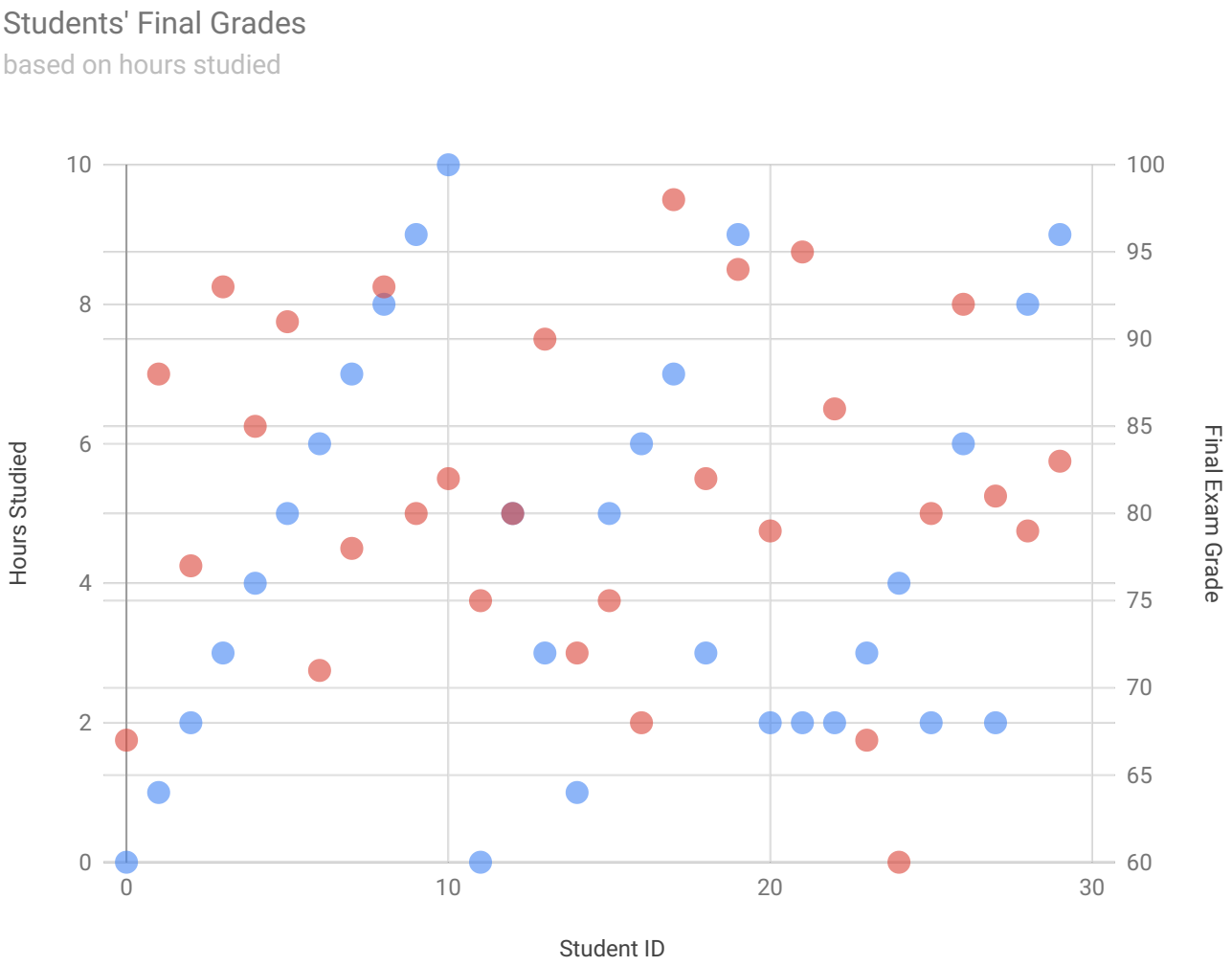
...with this:

```
chart.draw(data, google.charts.Scatter.convertOptions(options));
```

Dual-Y charts

Note: Dual-Y axes are available only for Material charts (i.e., those with package `scatter`).

Sometimes you'll want to display two series in a scatter chart, with two independent y-axes: a left axis for one series, and a right axis for another:



CODE IT YOURSELF ON JSFIDDLE

Note that not only are our two y-axes labeled differently ("Final Exam Grade" versus "Hours Studied") but they each have their own independent scales and gridlines. If you want to customize this behavior, use the `vAxis.gridlines` options.

In the code below, the `axes` and `series` options together specify the dual-Y appearance of the chart. The `series` option specifies which axis to use for each (`'final grade'` and `'hours studied'`; they needn't have any relation to the column names in the datatable). The `axes` option then makes this chart a dual-Y chart, placing the `'Final Exam Grade'` axis on the left and the `'Hours Studied'` axis on the right.

```
google.charts.load('current', {'packages':['scatter']});
google.charts.setOnLoadCallback(drawChart);

function drawChart () {

  var data = new google.visualization.DataTable();
  data.addColumn('number', 'Student ID');
  data.addColumn('number', 'Hours Studied');
```

```
    data.addColumn('number', 'Final');

    data.addRows([
      [0, 0, 67],  [1, 1, 88],   [2, 2, 77],
      [3, 3, 93],  [4, 4, 85],   [5, 5, 91],
      [6, 6, 71],  [7, 7, 78],   [8, 8, 93],
      [9, 9, 80],  [10, 10, 82], [11, 0, 75],
      [12, 5, 80], [13, 3, 90],  [14, 1, 72],
      [15, 5, 75], [16, 6, 68],  [17, 7, 98],
      [18, 3, 82], [19, 9, 94],  [20, 2, 79],
      [21, 2, 95], [22, 2, 86],  [23, 3, 67],
      [24, 4, 60], [25, 2, 80],  [26, 6, 92],
      [27, 2, 81], [28, 8, 79],  [29, 9, 83]
    ]);

    var options = {
      chart: {
        title: 'Students\' Final Grades',
        subtitle: 'based on hours studied'
      },
      width: 800,
      height: 500,
      series: {
        0: {axis: 'hours studied'},
        1: {axis: 'final grade'}
      },
      axes: {
        y: {
          'hours studied': {label: 'Hours Studied'},
          'final grade': {label: 'Final Exam Grade'}
        }
      }
    };

    var chart = new google.charts.Scatter(document.getElementById('scatte

    chart.draw(data, options);

  }
```

Top-X charts

Note: Top-X axes are available only for Material charts (i.e., those with package `scatter`).

If you want to put the X-axis labels and title on the top of your chart rather than the bottom, you can do that in Material charts with the `axes.x` option:

Students' Final Grades
based on hours studied

CODE IT YOURSELF ON JSFIDDLE

```
google.charts.load('current', {'packages':['scatter']});
google.charts.setOnLoadCallback(drawChart);

function drawChart () {

  var data = new google.visualization.DataTable();
  data.addColumn('number', 'Hours Studied');
  data.addColumn('number', 'Final');

  data.addRows([
    [0, 67],  [1, 88],  [2, 77],
    [3, 93],  [4, 85],  [5, 91],
    [6, 71],  [7, 78],  [8, 93],
    [9, 80],  [10, 82], [0, 75],
    [5, 80],  [3, 90],  [1, 72],
    [5, 75],  [6, 68],  [7, 98],
```

```
        [3, 82],  [9, 94],  [2, 79],
        [2, 95],  [2, 86],  [3, 67],
        [4, 60],  [2, 80],  [6, 92],
        [2, 81],  [8, 79],  [9, 83],
        [3, 75],  [1, 80],  [3, 71],
        [3, 89],  [4, 92],  [5, 85],
        [6, 92],  [7, 78],  [6, 95],
        [3, 81],  [0, 64],  [4, 85],
        [2, 83],  [3, 96],  [4, 77],
        [5, 89],  [4, 89],  [7, 84],
        [4, 92],  [9, 98]
      ]);

      var options = {
        width: 800,
        height: 500,
        chart: {
          title: 'Students\' Final Grades',
          subtitle: 'based on hours studied'
        },
        axes: {
          x: {
            0: {side: 'top'}
          }
        }
      };

      var chart = new google.charts.Scatter(document.getElementById('scatte

      chart.draw(data, google.charts.Scatter.convertOptions(options));
    }
```

Loading

The `google.charts.load` package name is `"corechart"`.

```
google.charts.load("current", {packages: ["corechart"]});
```

For Material Scatter Charts, the `google.charts.load` package name is `"scatter"`.

```
google.charts.load("current", {packages: ["scatter"]});
```

The visualization's class name is `google.visualization.ScatterChart`.

```
var visualization = new google.visualization.ScatterChart(container);
```

For Material Scatter Charts, the visualization's class name is `google.charts.Scatter`.

```
var visualization = new google.charts.Scatter(container);
```

Data format

Rows: Each row in the table represents a set of data points with the same x-axis value.

Columns:

	Column 0	Column 1
Purpose:	Data point X values	Series 1 Y values
Data Type:	string, number, or date/datetime/timeofday	string, number, or
Role:	data	data
Optional column roles (https://developers.google.com/chart/interactive/docs/roles):	*None*	• certainty (https://devel • emphasis (https://devel • scope (https://devel • tooltip (https://devel

To specify multiple series, specify two or more Y-axis columns, and specify Y values in only one Y column:

X-values	Series 1 Y Values	Series 2 Y Values
10	null	75
20	null	18
33	null	22

55	16	null
14	61	null
48	3	null

Configuration options

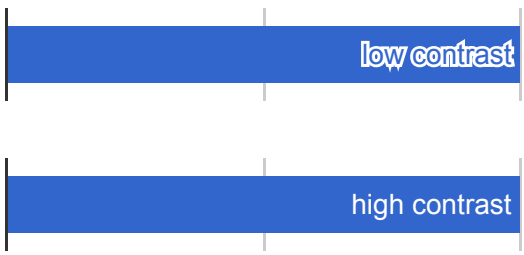
Name	

aggregationTarget	How multiple data selections are rolled up into tooltips:
	• `'category'`: Group selected data by x-value.
	• `'series'`: Group selected data by series.
	• `'auto'`: Group selected data by x-value if all selections have the sa otherwise.
	• `'none'`: Show only one tooltip per selection.
	`aggregationTarget` will often be used in tandem with `selectionM` e.g.:
	`var options = {`
	`  // Allow multiple`
	`  // simultaneous selections.`
	`  selectionMode: 'multiple',`
	`  // Trigger tooltips`
	`  // on selections.`
	`  tooltip: {trigger: 'selection'},`
	`  // Group selections`
	`  // by x-value.`
	`  aggregationTarget: 'category',`
	`};`
	Type: string
	Default: 'auto'

animation.duration	The duration of the animation, in milliseconds. For details, see the anim (https://developers.google.com/chart/interactive/docs/animation). **Type:** number **Default:** 0
animation.easing	The easing function applied to the animation. The following options ar

	• 'linear' - Constant speed. • 'in' - Ease in - Start slow and speed up. • 'out' - Ease out - Start fast and slow down. • 'inAndOut' - Ease in and out - Start slow, speed up, then slow down. **Type:** string **Default:** 'linear'
animation.startup	Determines if the chart will animate on the initial draw. If **`true`**, the cha animate to its final state. **Type:** boolean **Default** false
annotations.boxStyle	For charts that support annotations (https://developers.google.com/cl **`annotations.boxStyle`** object controls the appearance of the boxe

```
var options = {
  annotations: {
    boxStyle: {
      // Color of the box outline.
      stroke: '#888',
      // Thickness of the box outline.
      strokeWidth: 1,
      // x-radius of the corner curvature.
      rx: 10,
      // y-radius of the corner curvature.
      ry: 10,
      // Attributes for linear gradient fill.
      gradient: {
        // Start color for gradient.
        color1: '#fbf6a7',
        // Finish color for gradient.
        color2: '#33b679',
        // Where on the boundary to start and
        // end the color1/color2 gradient,
        // relative to the upper left corner
        // of the boundary.
        x1: '0%', y1: '0%',
        x2: '100%', y2: '100%',
        // If true, the boundary for x1,
        // y1, x2, and y2 is the box. If
        // false, it's the entire chart.
        useObjectBoundingBoxUnits: true
      }
    }
  }
```

	`};`  This option is currently supported for area, bar, column, combo, line, ar supported by the Annotation Chart (https://developers.google.com/chart/interactive/docs/gallery/annota **Type:** object **Default:** null
annotations.datum	For charts that support annotations (https://developers.google.com/cl **`annotations.datum`** object lets you override Google Charts' choice individual data elements (such as values displayed with each bar on a color with **`annotations.datum.stem.color`**, the stem length with **`annotations.datum.stem.length`**, and the style with **`annotatic`** **Type:** object **Default:** color is "black"; length is 12; style is "point".
annotations.domain	For charts that support annotations (https://developers.google.com/cl **`annotations.domain`** object lets you override Google Charts' choice domain (the major axis of the chart, such as the X axis on a typical line with **`annotations.domain.stem.color`**, the stem length with **`annotations.domain.stem.length`**, and the style with **`annotati`** **Type:** object **Default:** color is "black"; length is 5; style is "point".
annotations.highContrast	For charts that support annotations (https://developers.google.com/cl **`annotations.highContrast`** boolean lets you override Google Cha color. By default, **`annotations.highContrast`** is true, which cause color with good contrast: light colors on dark backgrounds, and dark o **`annotations.highContrast`** to false and don't specify your own ar will use the default series color for the annotation:  **Type:** boolean

	Default: true
annotations.stem	For charts that support annotations (https://developers.google.com/cl **`annotations.stem`** object lets you override Google Charts' choice fc color with **`annotations.stem.color`** and the stem length with **`ann`** that the stem length option has no effect on annotations with style **`'l`**: annotations, the stem length is always the same as the text, and for **`'l`** stem extends across the entire chart. **Type:** object **Default:** color is "black"; length is 5 for domain annotations and 12 for
annotations.style	For charts that support annotations (https://developers.google.com/cl **`annotations.style`** option lets you override Google Charts' choice either **`'line'`** or **`'point'`**. **Type:** string **Default:** 'point'
annotations.textStyle	For charts that support annotations (https://developers.google.com/cl **`annotations.textStyle`** object controls the appearance of the tex

```
var options = {
  annotations: {
    textStyle: {
      fontName: 'Times-Roman',
      fontSize: 18,
      bold: true,
      italic: true,
      // The color of the text.
      color: '#871b47',
      // The color of the text outline.
      auraColor: '#d799ae',
      // The transparency of the text.
      opacity: 0.8
    }
  }
};
```

This option is currently supported for area, bar, column, combo, line, ar
supported by the Annotation Chart

	(https://developers.google.com/chart/interactive/docs/gallery/annota **Type:** object **Default:** null
axisTitlesPosition	Where to place the axis titles, compared to the chart area. Supported v • in - Draw the axis titles inside the chart area. • out - Draw the axis titles outside the chart area. • none - Omit the axis titles. **Type:** string **Default:** 'out'
backgroundColor	The background color for the main area of the chart. Can be either a si example: `'red'` or `'#00cc00'`, or an object with the following proper **Type:** string or object **Default:** 'white'
backgroundColor.stroke	The color of the chart border, as an HTML color string. **Type:** string **Default:** '#666'
backgroundColor.strokeWidth	The border width, in pixels. **Type:** number **Default:** 0
backgroundColor.fill	The chart fill color, as an HTML color string. **Type:** string **Default:** 'white'
chart.title	For Material Charts (https://developers.google.com/chart/interactive/ this option specifies the title. **Type:** string **Default:** null
chart.subtitle	For Material Charts (https://developers.google.com/chart/interactive/ this option specifies the subtitle. Only Material Charts support subtitle **Type:** string **Default:** null
chartArea	An object with members to configure the placement and size of the ch drawn, excluding axis and legends). Two formats are supported: a num A simple number is a value in pixels; a number followed by % is a perce `{left:20,top:0,width:'50%',height:'75%'}`

	Type: object **Default:** null
chartArea.backgroundColor	Chart area background color. When a string is used, it can be either a h English color name. When an object is used, the following properties c: • `stroke`: the color, provided as a hex string or English color name. • `strokeWidth`: if provided, draws a border around the chart area of color of `stroke`). **Type:** string or object **Default:** 'white'
chartArea.left	How far to draw the chart from the left border. **Type:** number or string **Default:** auto
chartArea.top	How far to draw the chart from the top border. **Type:** number or string **Default:** auto
chartArea.width	Chart area width. **Type:** number or string **Default:** auto
chartArea.height	Chart area height. **Type:** number or string **Default:** auto
colors	The colors to use for the chart elements. An array of strings, where eac string, for example: `colors:['red','#004411']`. **Type:** Array of strings **Default:** default colors
crosshair	An object containing the crosshair (https://developers.google.com/cha properties for the chart. **Type:** object **Default:** null
crosshair.color	The crosshair color, expressed as either a color name (e.g., "blue") or a **Type:** string **Type:** default
crosshair.focused	An object containing the crosshair properties upon focus. Example: `crosshair: { focused: { color: '#3bc', opacit`

	Type: object **Default:** default
crosshair.opacity	The crosshair opacity, with `0.0` being fully transparent and `1.0` fully op **Type:** number **Default:** 1.0
crosshair.orientation	The crosshair orientation, which can be 'vertical' for vertical hairs only, only, or 'both' for traditional crosshairs. **Type:** string **Default:** 'both'
crosshair.selected	An object containing the crosshair properties upon selection. Example: `crosshair: { selected: { color: '#3bc', opaci` **Type:** object **Default:** default
crosshair.trigger	When to display crosshairs: on `'focus'`, `'selection'`, or `'both'`. **Type:** string **Default:** 'both'
curveType	Controls the curve of the lines when the line width is not zero. Can be c • 'none' - Straight lines without curve. • 'function' - The angles of the line will be smoothed. **Type:**string **Default:** 'none'
dataOpacity	The transparency of data points, with 1.0 being completely opaque and histogram, bar, and column charts, this refers to the visible data: dots i rectangles in the others. In charts where *selecting data* creates a dot, s this refers to the circles that appear upon hover or selection. The comb and this option has no effect on other charts. (To change the opacity o opacity (https://developers.google.com/chart/interactive/docs/gallery **Type:** number **Default:** 1.0
enableInteractivity	Whether the chart throws user-based events or reacts to user interacti throw 'select' or other interaction-based events (but *will* throw ready or hovertext or otherwise change depending on user input. **Type:** boolean **Default:** true
explorer	The `explorer` option allows users to pan and zoom Google charts. `ex` default explorer behavior, enabling users to pan horizontally and vertica

	and out by scrolling. This feature is **experimental** and may change in future releases. **Note:** The explorer only works with continuous axes (such as numbers **Type:** object **Default:** null
explorer.actions	The Google Charts explorer supports three actions: • `dragToPan`: Drag to pan around the chart horizontally and verticall horizontal axis, use `explorer: { axis: 'horizontal' }`. Sir • `dragToZoom`: The explorer's default behavior is to zoom in and out `explorer: { actions: ['dragToZoom', 'rightClickToI` across a rectangular area zooms into that area. We recommend usi whenever `dragToZoom` is used. See `explorer.maxZoomIn`, `expl` `explorer.zoomDelta` for zoom customizations. • `rightClickToReset`: Right clicking on the chart returns it to the **Type:** Array of strings **Default:** ['dragToPan', 'rightClickToReset']
explorer.axis	By default, users can pan both horizontally and vertically when the `exp` want to users to only pan horizontally, use `explorer: { axis: 'ho` `explorer: { axis: 'vertical' }` enables vertical-only panning **Type:** string **Default:** both horizontal and vertical panning
explorer.keepInBounds	By default, users can pan all around, regardless of where the data is. T beyond the original chart, use `explorer: { keepInBounds: true` **Type:** boolean **Default:** false
explorer.maxZoomIn	The maximum that the explorer can zoom in. By default, users will be a they'll see only 25% of the original view. Setting `explorer: { maxZo` zoom in only far enough to see half of the original view. **Type:** number **Default:** 0.25
explorer.maxZoomOut	The maximum that the explorer can zoom out. By default, users will be that the chart will take up only 1/4 of the available space. Setting `expl` would let users zoom out far enough that the chart would take up only **Type:** number **Default:** 4

explorer.zoomDelta	When users zoom in or out, **`explorer.zoomDelta`** determines how r the number, the smoother and slower the zoom. **Type:** number **Default:** 1.5
fontSize	The default font size, in pixels, of all text in the chart. You can override chart elements. **Type:** number **Default:** automatic
fontName	The default font face for all text in the chart. You can override this usin elements. **Type:** string **Default:** 'Arial'
forceIFrame	Draws the chart inside an inline frame. (Note that on IE8, this option is in i-frames.) **Type:** boolean **Default:** false

hAxis	An object with members to configure various horizontal axis elements. object, you can use object literal notation, as shown here:
	
	`{`
	`  title: 'Hello',`
	`  titleTextStyle: {`
	`  color: '#FF0000'`
	`  }`
	`}`
	
	Type: object
	Default: null

hAxis.baseline	The baseline for the horizontal axis. **Type:** number **Default:** automatic
hAxis.baselineColor	The color of the baseline for the horizontal axis. Can be any HTML colo **`'#00cc00'`**. **Type:** number **Default:** 'black'
hAxis.direction	The direction in which the values along the horizontal axis grow. Specif values.

	Type: 1 or -1 **Default:** 1
hAxis.format	A format string for numeric axis labels. This is a subset of the ICU patt (http://icu-project.org/apiref/icu4c/classDecimalFormat.html#_details `{format:'#,###%'}` will display values "1,000%", "750%", and "50%" can also supply any of the following: • `{format: 'none'}`: displays numbers with no formatting (e.g., 8 • `{format: 'decimal'}`: displays numbers with thousands separa • `{format: 'scientific'}`: displays numbers in scientific notati • `{format: 'currency'}`: displays numbers in the local currency • `{format: 'percent'}`: displays numbers as percentages (e.g., 8 • `{format: 'short'}`: displays abbreviated numbers (e.g., 8M) • `{format: 'long'}`: displays numbers as full words (e.g., 8 millio The actual formatting applied to the label is derived from the locale the more details, see loading charts with a specific locale (https://developers.google.com/chart/interactive/docs/library_loading . **Type:** string **Default:** auto
hAxis.gridlines	An object with members to configure the gridlines on the horizontal ax object, you can use object literal notation, as shown here: `{color: '#333', count: 4}` **Type:** object **Default:** null
hAxis.gridlines.color	The color of the horizontal gridlines inside the chart area. Specify a val **Type:** string **Default:** '#CCC'
hAxis.gridlines.count	The number of horizontal gridlines inside the chart area. Minimum valu automatically compute the number of gridlines. **Type:** number **Default:** 5
hAxis.gridlines.units	Overrides the default format for various aspects of date/datetime/time with chart computed gridlines. Allows formatting for years, months, da milliseconds.

	General format is:
	
	`gridlines: {`
	`  units: {`
	`  years: {format: [/*format strings here*/]},`
	`  months: {format: [/*format strings here*/]},`
	`  days: {format: [/*format strings here*/]}`
	`  hours: {format: [/*format strings here*/]}`
	`  minutes: {format: [/*format strings here*/]}`
	`  seconds: {format: [/*format strings here*/]},`
	`  milliseconds: {format: [/*format strings here*/]`
	`  }`
	`}`
	
	Additional information can be found in Dates and Times (https://developers.google.com/chart/interactive/docs/datesandtime
	
	Type: object
	Default: null

hAxis.minorGridlines	An object with members to configure the minor gridlines on the horizo hAxis.gridlines option. **Type:** object **Default:** null
hAxis.minorGridlines.color	The color of the horizontal minor gridlines inside the chart area. Specif **Type:** string **Default:** A blend of the gridline and background colors
hAxis.minorGridlines.count	The number of horizontal minor gridlines between two regular gridlines **Type:** number **Default:** 0

hAxis.minorGridlines.units	Overrides the default format for various aspects of date/datetime/time with chart computed minorGridlines. Allows formatting for years, mont seconds, and milliseconds.
	
	General format is:
	
	`gridlines: {`
	`  units: {`
	`  years: {format: [/*format strings here*/]},`
	`  months: {format: [/*format strings here*/]},`
	`  days: {format: [/*format strings here*/]}`
	`  hours: {format: [/*format strings here*/]}`
	`  minutes: {format: [/*format strings here*/]}`

	`seconds: {format: [/*format strings here*/]},`
	`milliseconds: {format: [/*format strings here*/]`
	`}`
	`}`
	
	Additional information can be found in Dates and Times (https://developers.google.com/chart/interactive/docs/datesandtime
	
	Type: object
	Default: null

hAxis.logScale	**`hAxis`** property that makes the horizontal axis a logarithmic scale (req Set to true for yes. **Type:** boolean **Default:** false
hAxis.scaleType	**`hAxis`** property that makes the horizontal axis a logarithmic scale. Car • null - No logarithmic scaling is performed. • 'log' - Logarithmic scaling. Negative and zero values are not plotted. setting **`hAxis: { logscale: true }`**. • 'mirrorLog' - Logarithmic scaling in which negative and zero values a negative number is the negative of the log of the absolute value. \ linear scale. **Type:** string **Default: null**
hAxis.textPosition	Position of the horizontal axis text, relative to the chart area. Supporte **Type:** string **Default:** 'out'

hAxis.textStyle	An object that specifies the horizontal axis text style. The object has th
	
	`{ color: <string>,`
	`fontName: <string>,`
	`fontSize: <number>,`
	`bold: <boolean>,`
	`italic: <boolean> }`
	
	The **`color`** can be any HTML color string, for example: **`'red'`** or **`'#00`** **`fontSize`**.
	
	Type: object

	Default: `{color: 'black', fontName: <global-font-name>, size>}`
hAxis.ticks	Replaces the automatically generated X-axis ticks with the specified ar should be either a valid tick value (such as a number, date, datetime, or an object, it should have a `v` property for the tick value, and an optional string to be displayed as the label. Examples: • `hAxis: { ticks: [5,10,15,20] }` • `hAxis: { ticks: [{v:32, f:'thirty two'}, {v:64, f` • `hAxis: { ticks: [new Date(2014,3,15), new Date(20` • `hAxis: { ticks: [16, {v:32, f:'thirty two'}, {v:64 }` **Type:** Array of elements **Default:** auto
hAxis.title	`hAxis` property that specifies the title of the horizontal axis. **Type:** string **Default:** null

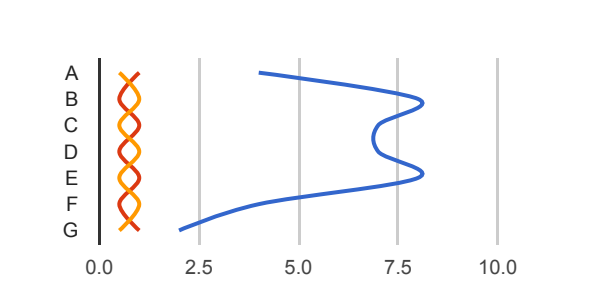
hAxis.titleTextStyle	An object that specifies the horizontal axis title text style. The object ha
	
	`{ color: <string>,`
	`  fontName: <string>,`
	`  fontSize: <number>,`
	`  bold: <boolean>,`
	`  italic: <boolean> }`
	
	The `color` can be any HTML color string, for example: `'red'` or `'#00` `fontSize`.
	
	Type: object
	Default: `{color: 'black', fontName: <global-font-name>, size>}`

hAxis.maxValue	Moves the max value of the horizontal axis to the specified value; this v Ignored if this is set to a value smaller than the maximum x-value of th `hAxis.viewWindow.max` overrides this property. **Type:** number **Default:** automatic
hAxis.minValue	Moves the min value of the horizontal axis to the specified value; this v Ignored if this is set to a value greater than the minimum x-value of the

	`hAxis.viewWindow.min` overrides this property. **Type:** number **Default:** automatic
hAxis.viewWindowMode	Specifies how to scale the horizontal axis to render the values within th string values are supported: • 'pretty' - Scale the horizontal values so that the maximum and minir bit inside the left and right of the chart area. This will cause **`haxis`** **`haxis.viewWindow.max`** to be ignored. • 'maximized' - Scale the horizontal values so that the maximum and left and right of the chart area. This will cause **`haxis.viewWindo`** **`haxis.viewWindow.max`** to be ignored. • 'explicit' - A deprecated option for specifying the left and right scale (Deprecated because it's redundant with **`haxis.viewWindow.mi`** **`haxis.viewWindow.max`**.) Data values outside these values will b **`hAxis.viewWindow`** object describing the maximum and minimur **Type:** string **Default:** Equivalent to 'pretty', but **`haxis.viewWindow.min`** and **`haxi`** precedence if used.
hAxis.viewWindow	Specifies the cropping range of the horizontal axis. **Type:** object **Default:** null
hAxis.viewWindow.max	The maximum horizontal data value to render. Ignored when **`hAxis.viewWindowMode`** is 'pretty' or 'maximized'. **Type:** number **Default:** auto
hAxis.viewWindow.min	The minimum horizontal data value to render. Ignored when **`hAxis.viewWindowMode`** is 'pretty' or 'maximized'. **Type:** number **Default:** auto
height	Height of the chart, in pixels. **Type:** number **Default:** height of the containing element
legend	An object with members to configure various aspects of the legend. To you can use object literal notation, as shown here: `{position: 'top', textStyle: {color: 'blue', fontSiz`

	Type: object **Default:** null
legend.alignment	Alignment of the legend. Can be one of the following: • 'start' - Aligned to the start of the area allocated for the legend. • 'center' - Centered in the area allocated for the legend. • 'end' - Aligned to the end of the area allocated for the legend. Start, center, and end are relative to the style -- vertical or horizontal -- c 'right' legend, 'start' and 'end' are at the top and bottom, respectively; fo would be at the left and right of the area, respectively. The default value depends on the legend's position. For 'bottom' legenc legends default to 'start'. **Type:** string **Default:** automatic
legend.maxLines	Maximum number of lines in the legend. Set this to a number greater tl legend. Note: The exact logic used to determine the actual number of l This option currently works only when legend.position is 'top'. **Type:** number **Default:** 1
legend.position	Position of the legend. Can be one of the following: • 'bottom' - Below the chart. • 'left' - To the left of the chart, provided the left axis has no series as: the legend on the left, use the option `targetAxisIndex: 1`. • 'in' - Inside the chart, by the top left corner. • 'none' - No legend is displayed. • 'right' - To the right of the chart. Incompatible with the `vAxes` optior • 'top' - Above the chart. **Type:** string **Default:** 'right'

legend.textStyle	An object that specifies the legend text style. The object has this forma
	
	`{ color: <string>,`
	`  fontName: <string>,`
	`  fontSize: <number>,`
	`  bold: <boolean>,`

	`italic: <boolean> }` The **`color`** can be any HTML color string, for example: **`'red'`** or **`'#00`** **`fontSize`**. **Type:** object **Default:** **`{color: 'black', fontName: <global-font-name>,`** **`size>}`**
lineWidth	Line width in pixels. Use zero to hide all lines and show only the points. **Type:** number **Default:** 0
orientation	The orientation of the chart. When set to **`'vertical'`**, rotates the axe instance) a column chart becomes a bar chart, and an area chart grow  **Type:** string **Default:** 'horizontal'
pointShape	The shape of individual data elements: 'circle', 'triangle', 'square', 'diamo points documentation (https://developers.google.com/chart/interacti **Type:** string **Default:** 'circle'
pointSize	Diameter of data points, in pixels. Use zero to hide all points. You can c series using the **`series`** property. If you're using a trendline (https://developers.google.com/chart/interactive/docs/gallery/trendli the **`pointSize`** of the points comprising it, which will change the appa avoid this, you can override it with the **`trendlines.n.pointsize`** o **Type:** number **Default:** 7
pointsVisible	Determines whether points will be displayed. Set to **`false`** to hide all p for individual series using the **`series`** property. If you're using a trendli (https://developers.google.com/chart/interactive/docs/gallery/trendli option will affect the visibility of the points on all trendlines unless you **`trendlines.n.pointsVisible`** option.

	This can also be overridden using the style role (https://developers.google.com/chart/interactive/docs/roles#stylerol **`{visible: true}`**". **Type:** boolean **Default:** true
selectionMode	When **`selectionMode`** is **`'multiple'`**, users may select multiple dat **Type:** string **Default:** 'single'

series	An array of objects, each describing the format of the corresponding s values for a series, specify an empty object {}. If a series or a value is n be used. Each object supports the following properties:
	
	• **`color`** - The color to use for this series. Specify a valid HTML color
	• **`labelInLegend`** - The description of the series to appear in the ch
	• **`lineWidth`** - Overrides the global **`lineWidth`** value for this series.
	• **`pointShape`** - Overrides the global **`pointShape`** value for this seri
	• **`pointSize`** - Overrides the global **`pointSize`** value for this series.
	• **`pointsVisible`** - Overrides the global **`pointsVisible`** value for
	• **`visibleInLegend`** - A boolean value, where true means that the s and false means that it should not. Default is true.
	
	You can specify either an array of objects, each of which applies to the can specify an object where each child has a numeric key indicating wl example, the following two declarations are identical, and declare the f from the legend, and the fourth as red and absent from the legend:
	
	`series: [`
	`  {color: 'black', visibleInLegend: false}, {}, {},`
	`  {color: 'red', visibleInLegend: false}`
	`]`
	`series: {`
	`  0:{color: 'black', visibleInLegend: false},`
	`  3:{color: 'red', visibleInLegend: false}`
	`}`
	
	Type: Array of objects, or object with nested objects
	Default: {}

theme	A theme is a set of predefined option values that work together to achi visual effect. Currently only one theme is available:

	• 'maximized' - Maximizes the area of the chart, and draws the legenc chart area. Sets the following options:
	`chartArea: {width: '100%', height: '100%'},`
	`legend: {position: 'in'},`
	`titlePosition: 'in', axisTitlesPosition: 'in',`
	`hAxis: {textPosition: 'in'}, vAxis: {textPosition`
	
	Type: string
	Default: null

title	Text to display above the chart. **Type:** string **Default:** no title
titlePosition	Where to place the chart title, compared to the chart area. Supported v • in - Draw the title inside the chart area. • out - Draw the title outside the chart area. • none - Omit the title. **Type:** string **Default:** 'out'

titleTextStyle	An object that specifies the title text style. The object has this format:
	
	`{ color: <string>,`
	`fontName: <string>,`
	`fontSize: <number>,`
	`bold: <boolean>,`
	`italic: <boolean> }`
	
	The **`color`** can be any HTML color string, for example: **`'red'`** or **`'#00`** **`fontSize`**.
	
	Type: object
	Default: **`{color: 'black', fontName: <global-font-name>, size>}`**

tooltip	An object with members to configure various tooltip elements. To spec can use object literal notation, as shown here: `{textStyle: {color: '#FF0000'}, showColorCode: true}` **Type:** object

	Default: null
tooltip.ignoreBounds	If set to `true`, allows the drawing of tooltips to flow outside of the bou **Note:** This only applies to HTML tooltips. If this is enabled with SVG to the chart bounds will be cropped. See Customizing Tooltip Content (https://developers.google.com/chart/interactive/docs/customizing_t **Type:** boolean **Default:** false
tooltip.isHtml	If set to true, use HTML-rendered (rather than SVG-rendered) tooltips. S (https://developers.google.com/chart/interactive/docs/customizing_t ★ **Note:** customization of the HTML tooltip content via the tooltip column (https://developers.google.com/chart/interactive/docs/roles#tooltipr Bubble Chart (https://developers.google.com/chart/interactive/docs/g **Type:** boolean **Default:** false
tooltip.showColorCode	If true, show colored squares next to the series information in the toolt **Type:** boolean **Default:** false

tooltip.textStyle	An object that specifies the tooltip text style. The object has this forma
	
	`{ color: <string>,`
	`  fontName: <string>,`
	`  fontSize: <number>,`
	`  bold: <boolean>,`
	`  italic: <boolean> }`
	
	The `color` can be any HTML color string, for example: `'red'` or `'#00` `fontSize`.
	
	Type: object
	Default: `{color: 'black', fontName: <global-font-name>, size>}`

tooltip.trigger	The user interaction that causes the tooltip to be displayed: • 'focus' - The tooltip will be displayed when the user hovers over the • 'none' - The tooltip will not be displayed. • 'selection' - The tooltip will be displayed when the user selects the e **Type:** string

	Default: 'focus'

trendlines	Displays trendlines (https://developers.google.com/chart/interactive/ charts that support them. By default, linear trendlines are used, but this **`trendlines.`*`n`*`.type`** option.
	
	Trendlines are specified on a per-series basis, so most of the time your
	
	`var options = {`
	`  trendlines: {`
	`  0: {`
	`  type: 'linear',`
	`  color: 'green',`
	`  lineWidth: 3,`
	`  opacity: 0.3,`
	`  showR2: true,`
	`  visibleInLegend: true`
	`  }`
	`  }`
	`}`
	
	Type: object
	Default: null

trendlines.n.color	The color of the trendline (https://developers.google.com/chart/intera expressed as either an English color name or a hex string. **Type:** string **Default:** default series color
trendlines.n.degree	For trendlines (https://developers.google.com/chart/interactive/docs/ **`'polynomial'`**, the degree of the polynomial (**2** for quadratic, **3** for cu degree may change from 3 to 2 in an upcoming release of Google Char **Type:** number **Default:** 3
trendlines.n.labelInLegend	If set, the trendline (https://developers.google.com/chart/interactive/ appear in the legend as this string. **Type:** string **Default:** null
trendlines.n.lineWidth	The line width of the trendline (https://developers.google.com/chart/interactive/docs/gallery/trendli **Type:** number **Default:** 2

trendlines.n.opacity	The transparency of the trendline (https://developers.google.com/chart/interactive/docs/gallery/trendli 1.0 (opaque). **Type:** number **Default:** 1.0
trendlines.n.pointSize	Trendlines (https://developers.google.com/chart/interactive/docs/gal by stamping a bunch of dots on the chart; this rarely-needed option lets dots. The trendline's **`lineWidth`** option will usually be preferable. How you're using the global **`pointSize`** option and want a different point s **Type:** number **Default:** 1
trendlines.n.pointsVisible	Trendlines (https://developers.google.com/chart/interactive/docs/gal by stamping a bunch of dots on the chart. The trendline's **`pointsVisi** the points for a particular trendline are visible. **Type:** boolean **Default:** true
trendlines.n.showR2	Whether to show the coefficient of determination (https://developers.google.com/chart/interactive/docs/gallery/trendli tooltip. **Type:** boolean **Default:** false
trendlines.n.type	Whether the trendlines (https://developers.google.com/chart/interacti **`'linear'`** (the default), **`'exponential'`**, or **`'polynomial'`**. **Type:** string **Default:** linear
trendlines.n.visibleInLegend	Whether the trendline (https://developers.google.com/chart/interactiv equation appears in the legend. (It will appear in the trendline tooltip.) **Type:** boolean **Default:** false
vAxis	An object with members to configure various vertical axis elements. To you can use object literal notation, as shown here: `{title: 'Hello', titleTextStyle: {color: '#FF0000'}}` **Type:** object **Default:** null
vAxis.baseline	**`vAxis`** property that specifies the baseline for the vertical axis. If the b grid line or smaller than the lowest grid line, it will be rounded to the clo

	Type: number **Default:** automatic
vAxis.baselineColor	Specifies the color of the baseline for the vertical axis. Can be any HTM `'red'` or `'#00cc00'`. **Type:** number **Default:** 'black'
vAxis.direction	The direction in which the values along the vertical axis grow. Specify - values. **Type:** 1 or -1 **Default:** 1
vAxis.format	A format string for numeric axis labels. This is a subset of the ICU patt (http://icu-project.org/apiref/icu4c/classDecimalFormat.html#_details `{format:'#,###%'}` will display values "1,000%", "750%", and "50%" can also supply any of the following: • `{format: 'none'}`: displays numbers with no formatting (e.g., 8 • `{format: 'decimal'}`: displays numbers with thousands separa • `{format: 'scientific'}`: displays numbers in scientific notati • `{format: 'currency'}`: displays numbers in the local currency • `{format: 'percent'}`: displays numbers as percentages (e.g., 8 • `{format: 'short'}`: displays abbreviated numbers (e.g., 8M) • `{format: 'long'}`: displays numbers as full words (e.g., 8 millio The actual formatting applied to the label is derived from the locale the more details, see loading charts with a specific locale (https://developers.google.com/chart/interactive/docs/library_loading . **Type:** string **Default:** auto
vAxis.gridlines	An object with members to configure the gridlines on the vertical axis. object, you can use object literal notation, as shown here: `{color: '#333', count: 4}` **Type:** object **Default:** null
vAxis.gridlines.color	The color of the vertical gridlines inside the chart area. Specify a valid **Type:** string

	Default: '#CCC'
vAxis.gridlines.count	The number of vertical gridlines inside the chart area. Minimum value i compute the number of gridlines. **Type:** number **Default:** 5

vAxis.gridlines.units	Overrides the default format for various aspects of date/datetime/time with chart computed gridlines. Allows formatting for years, months, da milliseconds.
	
	General format is:
	
	`gridlines: {`
	`  units: {`
	`  years: {format: [/*format strings here*/]},`
	`  months: {format: [/*format strings here*/]},`
	`  days: {format: [/*format strings here*/]}`
	`  hours: {format: [/*format strings here*/]}`
	`  minutes: {format: [/*format strings here*/]}`
	`  seconds: {format: [/*format strings here*/]},`
	`  milliseconds: {format: [/*format strings here*/]`
	`  }`
	`}`
	
	Additional information can be found in Dates and Times (https://developers.google.com/chart/interactive/docs/datesandtime
	
	Type: object
	Default: null

vAxis.minorGridlines	An object with members to configure the minor gridlines on the vertica vAxis.gridlines option. **Type:** object **Default:** null
vAxis.minorGridlines.color	The color of the vertical minor gridlines inside the chart area. Specify a **Type:** string **Default:** A blend of the gridline and background colors
vAxis.minorGridlines.count	The number of vertical minor gridlines between two regular gridlines. **Type:** number **Default:** 0
vAxis.minorGridlines.units	Overrides the default format for various aspects of date/datetime/time

	with chart computed minorGridlines. Allows formatting for years, mont seconds, and milliseconds.
	
	General format is:
	
	`gridlines: {`
	`  units: {`
	`  years: {format: [/*format strings here*/]},`
	`  months: {format: [/*format strings here*/]},`
	`  days: {format: [/*format strings here*/]}`
	`  hours: {format: [/*format strings here*/]}`
	`  minutes: {format: [/*format strings here*/]}`
	`  seconds: {format: [/*format strings here*/]},`
	`  milliseconds: {format: [/*format strings here*/]`
	`  }`
	`}`
	
	Additional information can be found in Dates and Times (https://developers.google.com/chart/interactive/docs/datesandtime
	
	Type: object
	Default: null

vAxis.logScale	If true, makes the vertical axis a logarithmic scale. Note: All values mus **Type:** boolean **Default:** false
vAxis.scaleType	`vAxis` property that makes the vertical axis a logarithmic scale. Can b • null - No logarithmic scaling is performed. • 'log' - Logarithmic scaling. Negative and zero values are not plotted. setting `vAxis: { logscale: true }`. • 'mirrorLog' - Logarithmic scaling in which negative and zero values a negative number is the negative of the log of the absolute value. V linear scale. **Type:** string **Default:** null
vAxis.textPosition	Position of the vertical axis text, relative to the chart area. Supported v **Type:** string **Default:** 'out'
vAxis.textStyle	An object that specifies the vertical axis text style. The object has this

	`{ color: <string>,`
	`  fontName: <string>,`
	`  fontSize: <number>,`
	`  bold: <boolean>,`
	`  italic: <boolean> }`
	
	The **`color`** can be any HTML color string, for example: **`'red'`** or **`'#00`**
	`fontSize`.
	
	Type: object
	Default: **`{color: 'black', fontName: <global-font-name>,`**
	`size>}`

vAxis.ticks	Replaces the automatically generated Y-axis ticks with the specified ar should be either a valid tick value (such as a number, date, datetime, or an object, it should have a **`v`** property for the tick value, and an optional string to be displayed as the label. Examples: • **`vAxis: { ticks: [5,10,15,20] }`** • **`vAxis: { ticks: [{v:32, f:'thirty two'}, {v:64, f`** • **`vAxis: { ticks: [new Date(2014,3,15), new Date(20`** • **`vAxis: { ticks: [16, {v:32, f:'thirty two'}, {v:6`** **`}`** **Type:** Array of elements **Default:** auto
vAxis.title	**`vAxis`** property that specifies a title for the vertical axis. **Type:** string **Default:** no title

vAxis.titleTextStyle	An object that specifies the vertical axis title text style. The object has
	
	`{ color: <string>,`
	`  fontName: <string>,`
	`  fontSize: <number>,`
	`  bold: <boolean>,`
	`  italic: <boolean> }`
	
	The **`color`** can be any HTML color string, for example: **`'red'`** or **`'#00`**
	`fontSize`.
	
	Type: object

	Default: `{color: 'black', fontName: <global-font-name>, size>}`
vAxis.maxValue	Moves the max value of the vertical axis to the specified value; this will Ignored if this is set to a value smaller than the maximum y-value of th `vAxis.viewWindow.max` overrides this property. **Type:** number **Default:** automatic
vAxis.minValue	Moves the min value of the vertical axis to the specified value; this will Ignored if this is set to a value greater than the minimum y-value of the `vAxis.viewWindow.min` overrides this property. **Type:** number **Default:** null
vAxis.viewWindowMode	Specifies how to scale the vertical axis to render the values within the values are supported: • 'pretty' - Scale the vertical values so that the maximum and minimu inside the top and bottom of the chart area. This will cause `vaxis.` `vaxis.viewWindow.max` to be ignored. • 'maximized' - Scale the vertical values so that the maximum and mi and bottom of the chart area. This will cause `vaxis.viewWindow` `vaxis.viewWindow.max` to be ignored. • 'explicit' - A deprecated option for specifying the top and bottom sc (Deprecated because it's redundant with `vaxis.viewWindow.mi` `vaxis.viewWindow.max`. Data values outside these values will b `vAxis.viewWindow` object describing the maximum and minimur **Type:** string **Default:** Equivalent to 'pretty', but `vaxis.viewWindow.min` and `vaxi` precedence if used.
vAxis.viewWindow	Specifies the cropping range of the vertical axis. **Type:** object **Default:** null
vAxis.viewWindow.max	The maximum vertical data value to render. Ignored when `vAxis.viewWindowMode` is 'pretty' or 'maximized'. **Type:** number **Default:** auto
vAxis.viewWindow.min	The minimum horizontal data value to render. Ignored when `vAxis.viewWindowMode` is 'pretty' or 'maximized'.

	Type: number **Default:** auto
width	Width of the chart, in pixels. **Type:** number **Default:** width of the containing element

Methods

Method	
`draw(data, options)`	Draws the chart. The chart accepts further method calls only after the (#Events)event is fired. `Extended description` (https://developers.google.com/chart/interactive/docs/reference#vis **Return Type:** none
`getAction(actionID)`	Returns the tooltip action object with the requested `actionID`. **Return Type:** object

`getBoundingBox(id)`	Returns an object containing the left, top, width, and height of chart ele The format for `id` isn't yet documented (they're the return values of eve (https://developers.google.com/chart/interactive/docs/events)), but h some examples:
	
	`var cli = chart.getChartLayoutInterface();`
	
	Height of the chart area
	
	`cli.getBoundingBox('chartarea').height`
	
	Width of the third bar in the first series of a bar or column cl
	
	`cli.getBoundingBox('bar#0#2').width`
	
	Bounding box of the fifth wedge of a pie chart
	
	`cli.getBoundingBox('slice#4')`
	
	Bounding box of the chart data of a vertical (e.g., column) c
	
	`cli.getBoundingBox('vAxis#0#gridline')`
	
	Bounding box of the chart data of a horizontal (e.g., bar) cha

	`cli.getBoundingBox('hAxis#0#gridline')` Values are relative to the container of the chart. Call this *after* the chart **Return Type:** object

`getChartAreaBoundingBox()`	Returns an object containing the left, top, width, and height of the chart (i.e., excluding labels and legend):
	
	`var cli = chart.getChartLayoutInterface();`
	
	`cli.getChartAreaBoundingBox().left`
	`cli.getChartAreaBoundingBox().top`
	`cli.getChartAreaBoundingBox().height`
	`cli.getChartAreaBoundingBox().width`
	
	Values are relative to the container of the chart. Call this *after* the chart
	
	Return Type: object

`getChartLayoutInterface()`	Returns an object containing information about the onscreen placement chart and its elements. The following methods can be called on the returned object: • `getBoundingBox` • `getChartAreaBoundingBox` • `getHAxisValue` • `getVAxisValue` • `getXLocation` • `getYLocation` Call this *after* the chart is drawn. **Return Type:** object
`getHAxisValue(position, optional_axis_index)`	Returns the logical horizontal value at `position`, which is an offset fr container's left edge. Can be negative. Example: `chart.getChartLayoutInterface().getHAxisValue` Call this *after* the chart is drawn.

	Return Type: number
`getImageURI()`	Returns the chart serialized as an image URI. Call this *after* the chart is drawn. See Printing PNG Charts (https://developers.google.com/chart/interactive/docs/printing). **Return Type:** string
`getSelection()`	Returns an array of the selected chart entities. Selectable entities are p legend entries. A point corresponds to a cell in the data table, and a leg a column (row index is null). For this chart, only one entity can be selec given moment. `Extended description` (https://developers.google.com/chart/interactive/docs/reference#vis . **Return Type:** Array of selection elements
`getVAxisValue(position, optional_axis_index)`	Returns the logical vertical value at **`position`**, which is an offset from container's top edge. Can be negative. Example: **`chart.getChartLayoutInterface().getVAxisValue`** Call this *after* the chart is drawn. **Return Type:** number
`getXLocation(position, optional_axis_index)`	Returns the screen x-coordinate of **`position`** relative to the chart's co Example: **`chart.getChartLayoutInterface().getXLocation(`** Call this *after* the chart is drawn. **Return Type:** number
`getYLocation(position, optional_axis_index)`	Returns the screen y-coordinate of **`position`** relative to the chart's co Example: **`chart.getChartLayoutInterface().getYLocation(`** Call this *after* the chart is drawn. **Return Type:** number
`removeAction(actionID)`	Removes the tooltip action with the requested **`actionID`** from the cha **Return Type: none**
`setAction(action)`	Sets a tooltip action to be executed when the user clicks on the action The **`setAction`** method takes an object as its action parameter. This should specify 3 properties: **`id`**— the ID of the action being set, **`text`** –

	should appear in the tooltip for the action, and **`action`** — the function be run when a user clicks on the action text. Any and all tooltip actions should be set prior to calling the chart's **`dra`** method. Extended description (https://developers.google.com/chart/interactive/docs/reference#vis **Return Type: none**
`setSelection()`	Selects the specified chart entities. Cancels any previous selection. Se entities are points and legend entries. A point corresponds to a cell in t table, and a legend entry to a column (row index is null). For this chart, entity can be selected at a time. `Extended description` (https://developers.google.com/chart/interactive/docs/reference#vis . **Return Type:** none
`clearChart()`	Clears the chart, and releases all of its allocated resources. **Return Type:** none

Events

For more information on how to use these events, see Basic Interactivity (https://developers.google.com/chart/interactive/docs/basic_interactivity), Handling Events (https://developers.google.com/chart/interactive/docs/events), and Firing Events (https://developers.google.com/chart/interactive/docs/dev/events).

Name	
`animationfinish`	Fired when transition animation is complete. **Properties:** none
`click`	Fired when the user clicks inside the chart. Can be used to identify when the title, data elements, legend entries, axes, gridlines, or labels are clicked. **Properties:** targetID
`error`	Fired when an error occurs when attempting to render the chart. **Properties:** id, message
`onmouseover`	Fired when the user mouses over a visual entity. Passes back the row and column indices of the corresponding data table element.

	Properties: row, column
`onmouseout`	Fired when the user mouses away from a visual entity. Passes back the row and column indices of the corresponding data table element. **Properties:** row, column
`ready`	The chart is ready for external method calls. If you want to interact with the chart, and call methods after you draw it, you should set up a listener for this event *before* you call the `draw` method, and call them only after the event was fired. **Properties:** none
`select`	Fired when the user clicks a visual entity. To learn what has been selected, call `getSelection()` (#Methods). **Properties:** none

Data policy

All code and data are processed and rendered in the browser. No data is sent to any server.

上次更新日期：二月 23, 2017