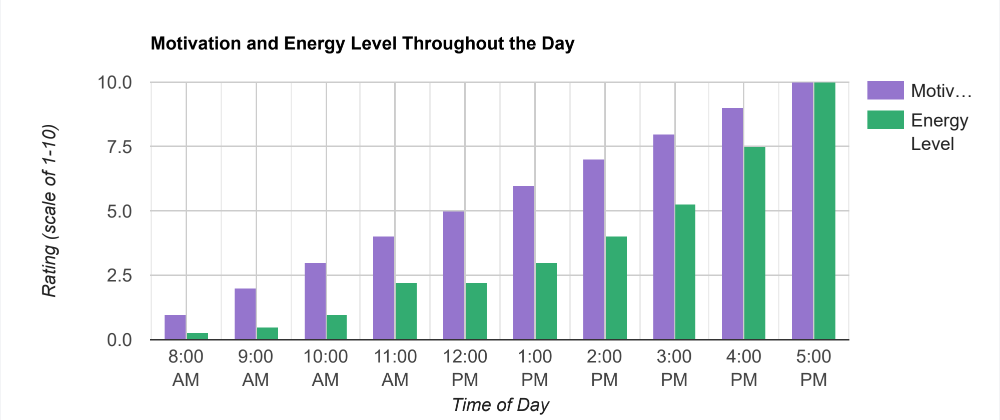
Visualization: Column Chart

Overview

A *column chart* is a vertical bar chart rendered in the browser using SVG (http://www.w3.org/Graphics/SVG/) or VML (http://en.wikipedia.org/wiki/Vector_Markup_Language), whichever is appropriate for the user's browser. Like all Google charts, column charts display tooltips when the user hovers over the data. For a horizontal version of this chart, see the bar chart (/chart/interactive/docs/gallery/barchart).

Examples

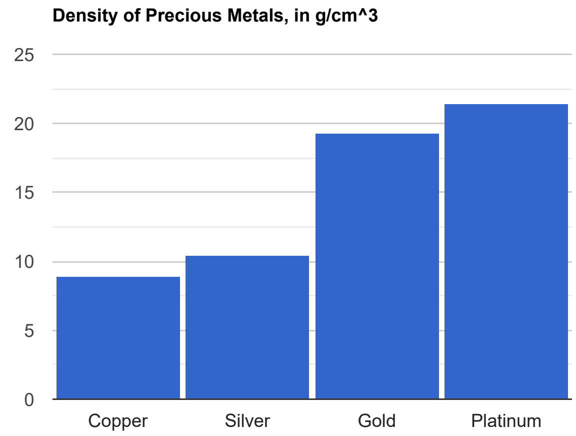
ustomizable column colors



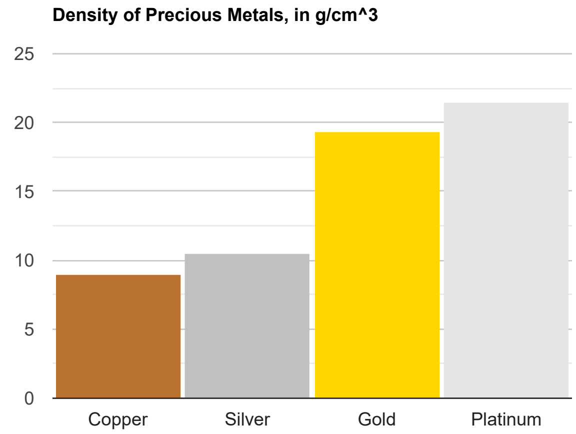
Code it yourself on JSFiddle

Coloring columns

Let's chart the densities of four precious metals:



Above, all colors are the default blue. That's because they're all part of the same series; if there were a second series, that would have been colored red. We can customize these colors with the *style role* (/chart/interactive/docs/roles#stylerole):



There are three different ways to choose the colors, and our data table showcases them all: RGB values, English color names, and a CSS-like declaration:

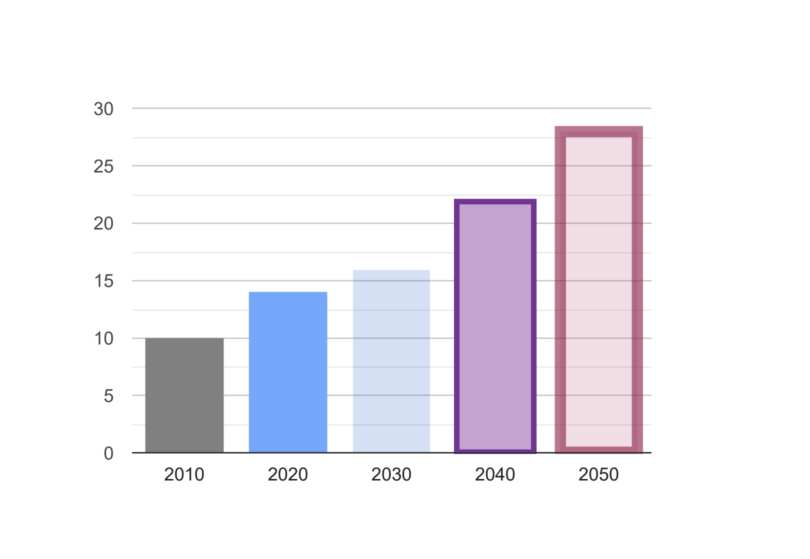
```
   var data = google.visualization.arrayToDataTable([
     ['Element', 'Density', { role: 'style' }],
     ['Copper', 8.94, '#b87333'],            // RGB value
     ['Silver', 10.49, 'silver'],            // English color name
     ['Gold', 19.30, 'gold'],

   ['Platinum', 21.45, 'color: #e5e4e2' ], // CSS-style declaration
  ]);
```

Column styles

The style role (/chart/interactive[illegible]/roles#stylerole) lets your control several aspects of column appearance with CSS-like declarations:

- `color`
- `opacity`
- `fill-color`
- `fill-opacity`
- `stroke-color`
- `stroke-opacity`
- `stroke-width`

We don't recommend that you mix styles too freely inside a chart—pick a style and stick with it—but to demonstrate all the style attributes, here's a sampler:



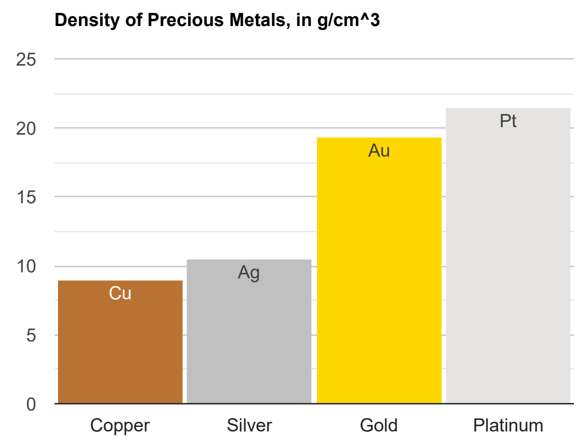
The first two columns each use a specific `color` (the first with an English name, the second with an RGB value). No `opacity` was chosen, so the default of 1.0 (fully opaque) is used; that's why the second column obscures the gridline behind it. In the third column, an `opacity` of 0.2 is used, revealing the gridline. In the fourth, three style attributes are used: `stroke-color` and `stroke-width` to draw the border, and `fill-color` to specify the color of the rectangle inside. The rightmost column additionally uses `stroke-opacity` and `fill-opacity` to choose opacities for the border and fill:

```
function drawChart() {
   var data = google.visualization.arrayToDataTable([
     ['Year', 'Visitations', { role: 'style' } ],
     ['2010', 10, 'color: gray'],
     ['2020', 14, 'color: #76A7FA'],
     ['2030', 16, 'opacity: 0.2'],
     ['2040', 22, 'stroke-color: #703593; stroke-width: 4; fill-color: #C5A5CF'],
     ['2050', 28, 'stroke-color: #871B47; stroke-opacity: 0.6; stroke-width: 8; fill-color: #BC5679; fill-opaci
   ]);
```

Labeling columns

Charts have several kinds of labels, such as tick labels, legend labels, and labels in the tooltips. In this section, we'll see how to put labels inside (or near) the columns in a column chart.

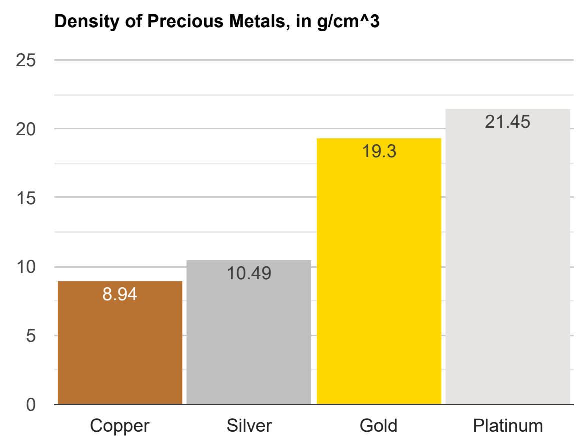
column with the appropriate chemical symbol. We can do that with the *annotation* role:



In our data table, we define a new column with `{ role: 'annotation' }` to hold our column labels:

```
      var data = google.visualization.arrayToDataTable([
        ['Element', 'Density', { role: 'style' }, { role: 'annotation' } ],
        ['Copper', 8.94, '#b87333', 'Cu' ],
        ['Silver', 10.49, 'silver', 'Ag' ],
        ['Gold', 19.30, 'gold', 'Au' ],
        ['Platinum', 21.45, 'color: #e5e4e2', 'Pt' ]
     ]);
```

While users can hover over the columns to see the data values, you might want to include them on the columns themselves:



This is a little more complicated than it should be, because we create a `DataView` to specify the annotation for each column.

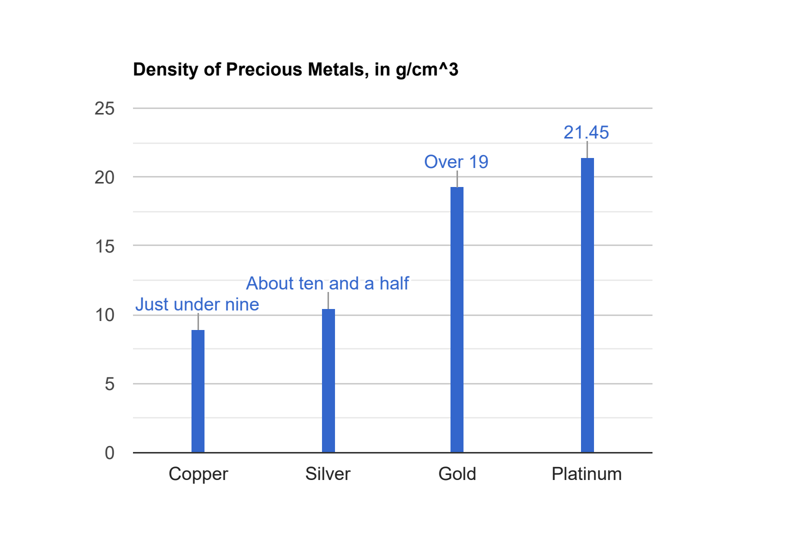
```
  <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
  <script type="text/javascript">
    google.charts.load("current", {packages:['corechart']});
    google.charts.setOnLoadCallback(drawChart);
    function drawChart() {
      var data = google.visualization.arrayToDataTable([
                                         { role: "style" } ],
        ["Copper", 8.94, "#b87333"],
        ["Silver", 10.49, "silver"],
        ["Gold", 19.30, "gold"],
        ["Platinum", 21.45, "color: #e5e4e2"]
      ]);

      var view = new google.visualization.DataView(data);
      view.setColumns([0, 1,
                       { calc: "stringify",
                         sourceColumn: 1,
                         type: "string",
                         role: "annotation" },
                       2]);

      var options = {
        title: "Density of Precious Metals, in g/cm^3",
        width: 600,
        height: 400,
        bar: {groupWidth: "95%"},
        legend: { position: "none" },
      };
      var chart = new google.visualization.ColumnChart(document.getElementById("columnchart_values"));
      chart.draw(view, options);
  }
  </script>
<div id="columnchart_values" style="width: 900px; height: 300px;"></div>
```

If we wanted to format the value differently, we could define a formatter (//developers.google.com/chart/interactive/docs/reference#formatters) and wrap it in a function like this:

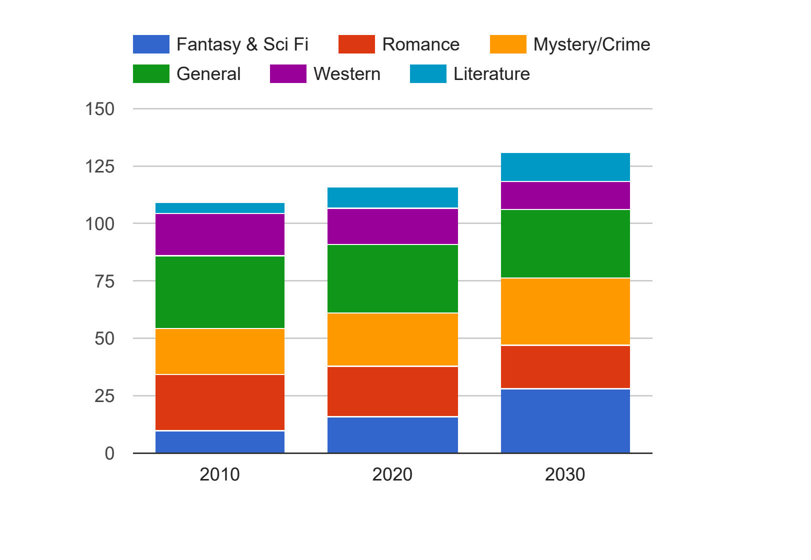
```
   function getValueAt(column, dataTable, row) {
    return dataTable.getFormattedValue(row, column);
   }
```

Then we could call it with `calc: getValueAt.bind(undefined, 1)`.

If the label is too big to fit entirely inside the column, it's displayed outside:



A *stacked column chart* is a column chart that places related values atop one another. If there are any negative values, they are stacked in reverse order below the chart's baseline. It's typically used when a category naturally divides into components. For instance, consider some hypothetical book sales, divided by genre and compared across time:

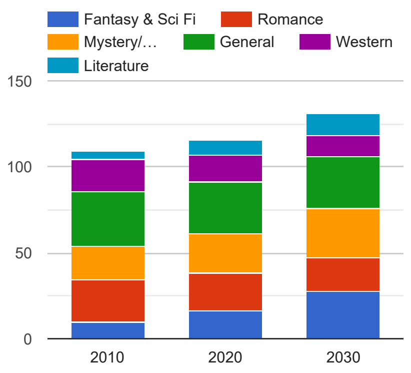


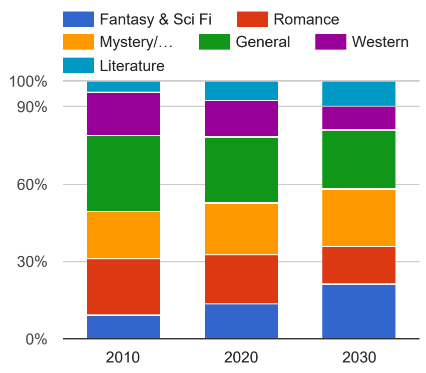
You create a stacked column chart by setting the `isStacked` option to `true`:

```
var data = google.visualization.arrayToDataTable([
  ['Genre', 'Fantasy & Sci Fi', 'Romance', 'Mystery/Crime', 'General',
   'Western', 'Literature', { role: 'annotation' } ],
  ['2010', 10, 24, 20, 32, 18, 5, ''],
  ['2020', 16, 22, 23, 30, 16, 9, ''],
  ['2030', 28, 19, 29, 30, 12, 13, '']
]);

var options = {
  width: 600,
  height: 400,
  legend: { position: 'top', maxLines: 3 },
  bar: { groupWidth: '75%' },
  isStacked: true,
};
```

Stacked column charts also support 100% stacking, where the stacks of elements at each domain-value are rescaled such that they add up to 100%. The options for this are `isStacked: 'percent'`, which formats each value as a percentage of 100%, and `isStacked: 'relative'`, which formats each value as a fraction of 1. There is also an `isStacked: 'absolute'` option, which is functionally equivalent to `isStacked: true`.

ı the 100% stacked chart on the right, the tick values are based on the relative 0-1 scale as fractions of 1, but the axis values are displayed as percentages. T se the percentage axis ticks are the result of applying a format of "#.##%" to the relative 0-1 scale values. When using `isStacked: 'percent'`, be sure to sp ks/axis values using the relative 0-1 scale.





Stacked (#stacked)100% Stacked (#100-stacked)

```
var options_fullStacked = {
  isStacked: 'percent',
  height: 300,
  legend: {position: 'top', maxLines: 3},
  vAxis: {
    minValue: 0,
    ticks: [0, .3, .6, .9, 1]
  }
};
```

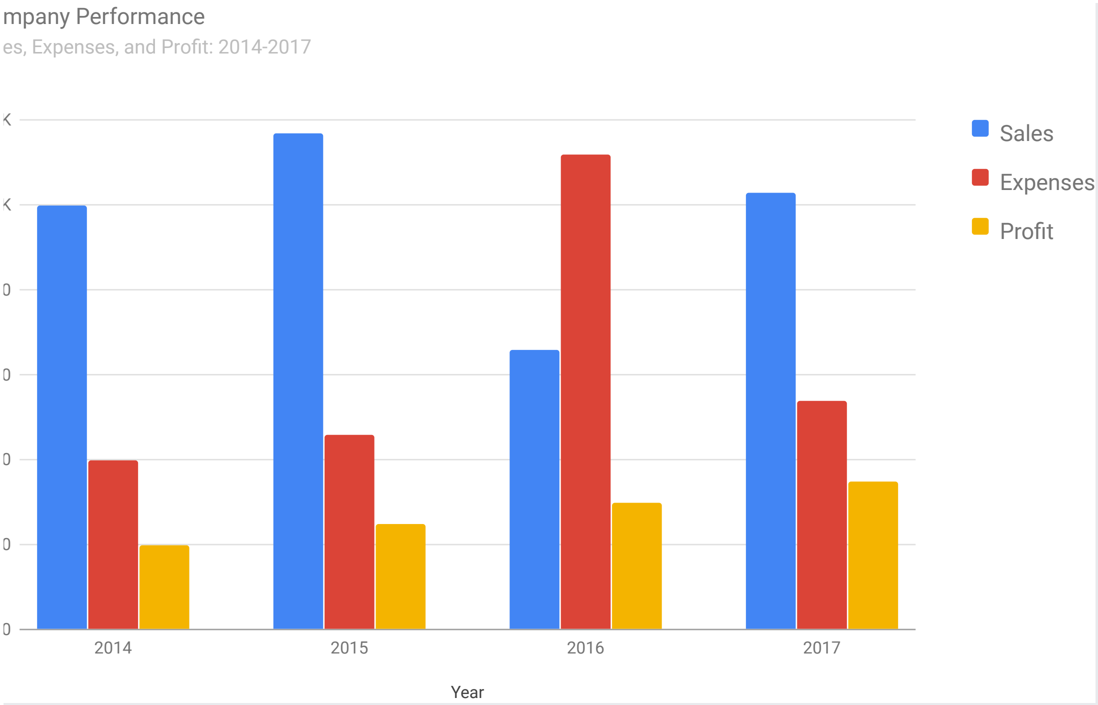
Creating Material column charts

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

Since bar charts and column charts are essentially identical but for orientation, we call both Material Bar Charts, regardless of whether the bars are vertical (classically, a column chart) or horizontal (a bar chart). In Material, the only difference is in the `bars` option. When set to `'horizontal'`, the orientation will resemble the traditional Classic Bar Chart; otherwise, the bars will be vertical.

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



Material Column Charts have many small improvements over Classic Column Charts, including an improved color palette, rounded corners, clearer label formatting, tighter default spacing between series, softer gridlines and titles (and the addition of subtitles).

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

      function drawChart() {
        var data = google.visualization.arrayToDataTable([
          ['Year', 'Sales', 'Expenses', 'Profit'],
          ['2014', 1000, 400, 200],
          ['2015', 1170, 460, 250],
          ['2016', 660, 1120, 300],
          ['2017', 1030, 540, 350]
        ]);

        var options = {
          chart: {
            title: 'Company Performance',
            subtitle: 'Sales, Expenses, and Profit: 2014-2017',
          }
        };

        var chart = new google.charts.Bar(document.getElementById('columnchart_material'));

        chart.draw(data, google.charts.Bar.convertOptions(options));
      }
    </script>
  </head>
  <body>
    <div id="columnchart_material" style="width: 800px; height: 500px;"></div>
  </body>
</html>
```

aterial 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. ıd a list of options that are not yet supported in this issue (https://github.com/google/google-visualization-issues/issues/2143).

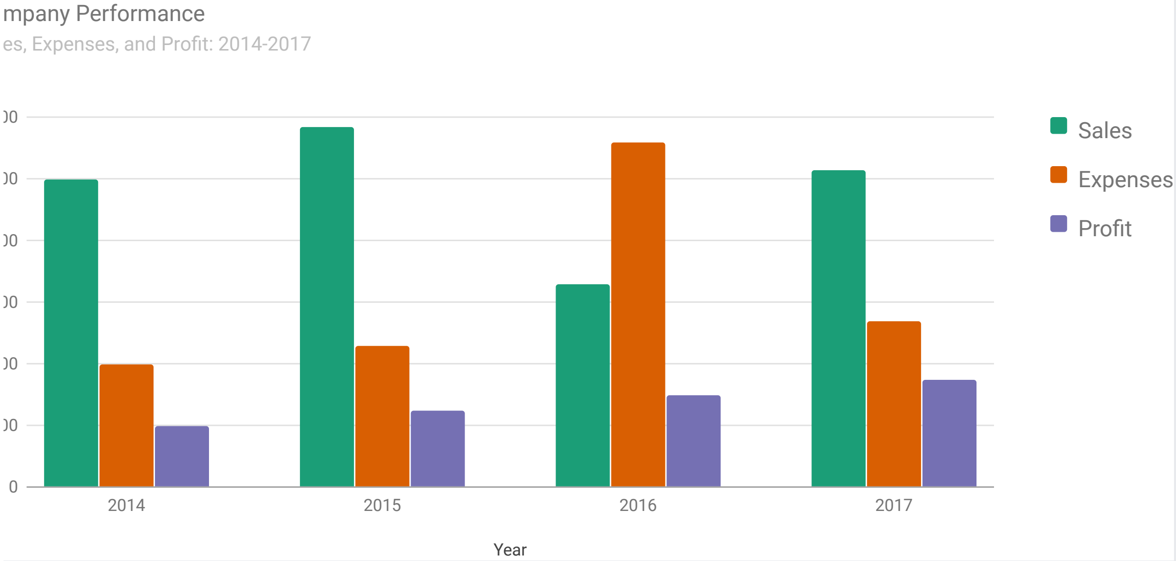
he way options are declared is not finalized, so if you are using any of the classic options, you must convert your to material options by replacing this line:

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

this:

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

Using `google.charts.Bar.convertOptions()` allows you to take advantage of certain features, such as the `hAxis/vAxis.format` preset options.



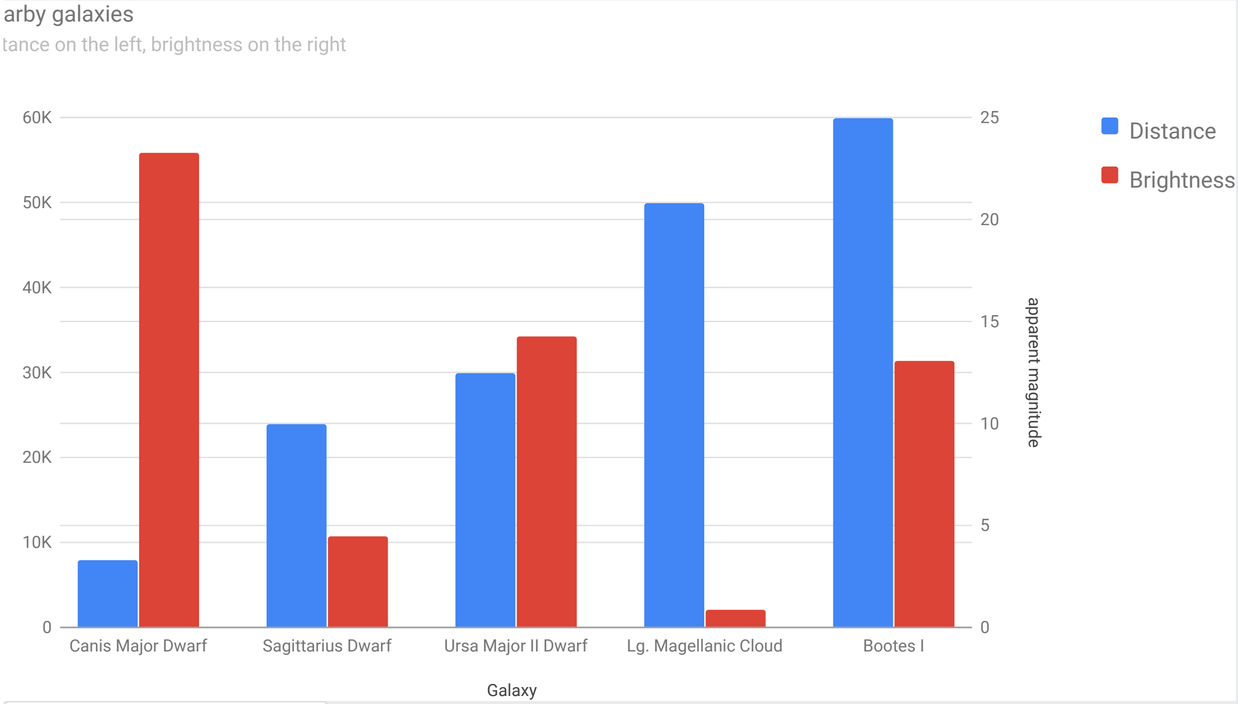
No Format Scientific Notation Decimal Short

Code it yourself on JSFiddle

Dual-Y charts

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

Change to Classic



Code it yourself on JSFiddle

Note that not only are our two y-axes labeled differently ("parsecs" versus "apparent magnitude") 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 (`'distance'` and `'brightness'`; 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 `'distance'` axis on the left (labeled "parsecs") and the `'brightness'` axis on the right (labeled "apparent magnitude").

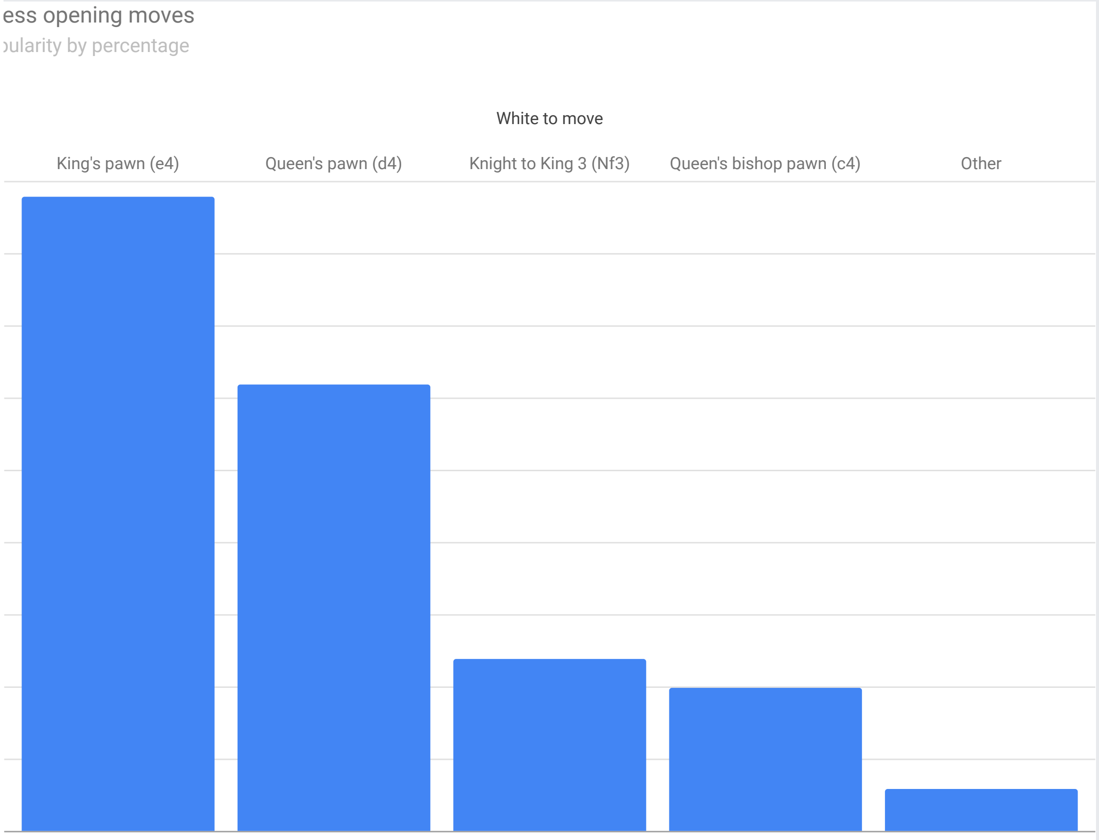
```
<html>
  <head>
    <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
    <script type="text/javascript">
      google.charts.load('current', {'packages':['corechart', 'bar']});
      google.charts.setOnLoadCallback(drawStuff);

      function drawStuff() {

        var button = document.getElementById('change-chart');
        var chartDiv = document.getElementById('chart_div');

        var data = google.visualization.arrayToDataTable([
          ['Galaxy', 'Distance', 'Brightness'],
          ['Canis Major Dwarf', 8000, 23.3],
          ['Sagittarius Dwarf', 24000, 4.5],
          ['Ursa Major II Dwarf', 30000, 14.3],
          ['Lg. Magellanic Cloud', 50000, 0.9],
          ['Bootes I', 60000, 13.1]
        ]);

        var materialOptions = {
          width: 900
```

```
            title: 'Nearby galaxies',
            subtitle: 'distance on the left, brightness on the right'
          },
          series: {
            0: { axis: 'distance' }, // Bind series 0 to an axis named 'distance'.
            1: { axis: 'brightness' } // Bind series 1 to an axis named 'brightness'.
          },
          axes: {
            y: {
              distance: {label: 'parsecs'}, // Left y-axis.
              brightness: {side: 'right', label: 'apparent magnitude'} // Right y-axis.
            }
          }
        };

        var classicOptions = {
          width: 900,
          series: {
            0: {targetAxisIndex: 0},
            1: {targetAxisIndex: 1}
          },
          title: 'Nearby galaxies - distance on the left, brightness on the right',
          vAxes: {
            // Adds titles to each axis.
            0: {title: 'parsecs'},
            1: {title: 'apparent magnitude'}
          }
        };

        function drawMaterialChart() {
          var materialChart = new google.charts.Bar(chartDiv);
          materialChart.draw(data, google.charts.Bar.convertOptions(materialOptions));
          button.innerText = 'Change to Classic';
          button.onclick = drawClassicChart;
        }

        function drawClassicChart() {
          var classicChart = new google.visualization.ColumnChart(chartDiv);
          classicChart.draw(data, classicOptions);
          button.innerText = 'Change to Material';
          button.onclick = drawMaterialChart;
        }

        drawMaterialChart();
    };
    </script>
  </head>
  <body>
    <button id="change-chart">Change to Classic</button>
    <br><br>
    <div id="chart_div" style="width: 800px; height: 500px;"></div>
  </body>
</html>
```

Top-X charts

Top-X axes are available only for Material charts (i.e., those with package **bar**).

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:



Code it yourself on JSFiddle

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

      function drawStuff() {
        var data = new google.visualization.arrayToDataTable([
          ['Move', 'Percentage'],
          ["King's pawn (e4)", 44],
          ["Queen's pawn (d4)", 31],
          ["Knight to King 3 (Nf3)", 12],
          ["Queen's bishop pawn (c4)", 10],
          ['Other', 3]
        ]);

        var options = {
          width: 800,
          legend: { position: 'none' },
          chart: {
            title: 'Chess opening moves',
            subtitle: 'popularity by percentage' },
          axes: {
            x: {
              0: { side: 'top', label: 'White to move'} // Top x-axis.
          bar: { groupWidth: "90%" }
        };

        var chart = new google.charts.Bar(document.getElementById('top_x_div'));
        // Convert the Classic options to Material options.
        chart.draw(data, google.charts.Bar.convertOptions(options));
      };
    </script>
  </head>
  <body>
    <div id="top_x_div" style="width: 800px; height: 600px;"></div>
  </body>
</html>
```

Loading

The `google.charts.load` package name is `"corechart"`. The visualization's class name is `google.visualization.ColumnChart`.

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

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

For Material Column Charts, the `google.charts.load` package name is `"bar"`. (Not a typo: the Material Bar Chart handles both orientations.) The visualization's class name is `google.charts.Bar`. (Not a typo: the Material Bar Chart handles both orientations.)

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

```
var chart = new google.charts.Bar(container);
```

Data format

Each row in the table represents a group of adjacent bars.

Rows: Each row in the table represents a group of bars.

Columns:

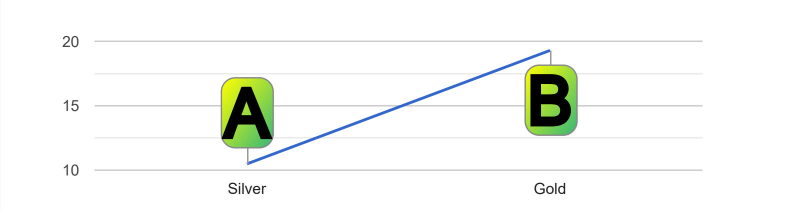
	Column 0	Column 1
Purpose:	- X-axis group labels (discrete (/chart/interactive/docs/customizing_axes#Discrete_vs_Continuous)) - X-axis values (continuous (/chart/interactive/docs/customizing_axes#Discrete_vs_Continuous))	Bar 1 values in this group
	discrete :/interactive/docs/customizing_axes#Discrete_vs_Continuous)	number

	) • number, date, datetime or timeofday (continuous (/chart/interactive/docs/customizing_axes#Discrete_vs_Continuous))	
Role:	domain	data
Optional column roles (/chart/interactive/docs/roles) :	• annotation (/chart/interactive/docs/roles#annotationrole) • annotationText (/chart/interactive/docs/roles#annotationtextrole)	• annotation (/chart/interactive/docs/roles#annotationrole) • annotationText (/chart/interactive/docs/roles#annotationtextr • certainty (/chart/interactive/docs/roles#certaintyrole) • interval (/chart/interactive/docs/roles#intervalrole) • scope (/chart/interactive/docs/roles#scoperol • style (/chart/interactive/docs/roles#stylerole) • tooltip (/chart/interactive/docs/roles#tooltiprol

Configuration options

Name	
animation.duration	The duration of the animation, in milliseconds. For details, see the animation documentation (/chart/interactive/docs/animation). **Type:** number **Default:** 0
animation.easing	The easing function applied to the animation. The following options are available: • '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 chart will start at the baseline and animate to its final state. **Type:** boolean **Default** false
annotations.alwaysOutside	In Bar (https://developers.google.com/chart/interactive/docs/gallery/barchart) and Column (https://developers.google.com/chart/interactive/docs/gallery/columnchart) charts, if set to `true`, draws all annotations outside of the Bar/Column. **Type:** boolean **Default:** false
	For charts that support annotations (https://developers.google.com/chart/interactive/docs/roles), the `annotations.boxStyle` object controls the appearance of the boxes surrounding annotations:

```
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, and scatter charts. It is not supported by the Annotation Chart (/chart/interactive/docs/gallery/annotationchart).

Type: object
Default: null

annotations.datum

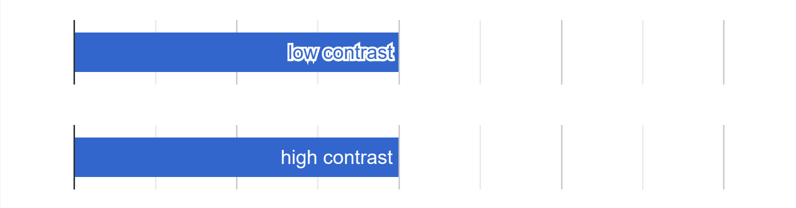
For charts that support annotations (/chart/interactive/docs/roles), the `annotations.datum` object lets you override Google Charts' choice for annotations provided for individual data elements (such as values displayed with each bar on a bar chart). You can control the color with `annotations.datum.stem.color`, the stem length with `annotations.datum.stem.length`, and the style with `annotations.datum.style`.
Type: object
Default: color is "black"; length is 12; style is "point".

annotations.domain

For charts that support annotations (/chart/interactive/docs/roles), the `annotations.domain` object lets you override Google Charts' choice for annotations provided for a domain (the major axis of the chart, such as the X axis on a typical line chart). You can control the color with `annotations.domain.stem.color`, the stem length with `annotations.domain.stem.length`, and the style with `annotations.domain.style`.
Type: object
Default: color is "black"; length is 5; style is "point".

annotations.highContrast

For charts that support annotations (https://developers.google.com/chart/interactive/docs/roles), the `annotations.highContrast` boolean lets you override Google Charts' choice of the annotation color. By default, `annotations.highContrast` is true, which causes Charts to select an annotation color with good contrast: light colors on dark backgrounds, and dark on light. If you set `annotations.highContrast` to false and don't specify your own annotation color, Google Charts will use the default series color for the annotation:



Type: boolean
Default: true

annotations.stem

For charts that support annotations (/chart/interactive/docs/roles), the `annotations.stem` object lets you override Google Charts' choice for the stem style. You can control color with `annotations.stem.color` and the stem length with `annotations.stem.length`. Note that the stem length option has no effect on annotations with style `'line'`: for `'line'` datum annotations, the stem length is always the same as the text, and for `'line'` domain annotations, the stem extends across the entire chart.
Type: object
Default: color is "black"; length is 5 for domain annotations and 12 for datum annotations.

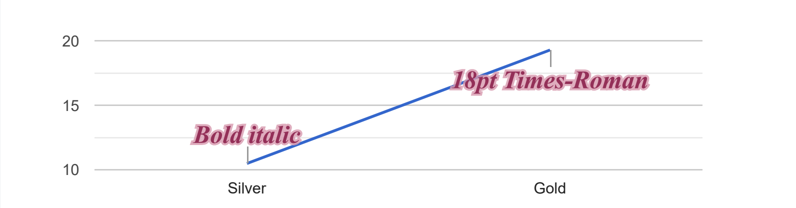
annotations.style

For charts that support annotations (/chart/interactive/docs/roles), the `annotations.style` option lets you override Google Charts' choice of the annotation type. It can be either `'line'` or `'point'`.
Type: string
Default: 'point'

annotations.textStyle

For charts that support annotations (https://developers.google.com/chart/interactive/docs/roles), the `annotations.textStyle` object controls the appearance of the text of the annotation:

```
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, and scatter charts. It is not supported by the Annotation Chart (https://developers.google.com/chart/interactive/docs/gallery/annotationchart).

Type: object
Default: null

axisTitlesPosition

Where to place the axis titles, compared to the chart area. Supported values:

	• 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 simple HTML color string, for example: `'red'` or `'#00cc00'`, or an object with the following properties. **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'
bar.groupWidth	The width of a group of bars, specified in either of these formats: • Pixels (e.g. 50). • Percentage of the available width for each group (e.g. '20%'), where '100%' means that groups have no space between them. **Type:** number or string **Default:** The golden ratio (http://en.wikipedia.org/wiki/Golden_ratio), approximately '61.8%'.
bars	Whether the bars in a **Material** Bar Chart (/chart/interactive/docs/gallery/barchart#Material) are vertical or horizontal. This option has no effect on Classic Bar Charts or Classic Column Charts. **Type:** 'horizontal' or 'vertical' **Default:** 'vertical'
chartArea	An object with members to configure the placement and size of the chart area (where the chart itself is drawn, excluding axis and legends). Two formats are supported: a number, or a number followed by %. A simple number is a value in pixels; a number followed by % is a percentage. Example: **`chartArea: {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 hex string (e.g., '#fdc') or an English color name. When an object is used, the following properties can be provided: • `stroke`: the color, provided as a hex string or English color name. • `strokeWidth`: if provided, draws a border around the chart area of the given width (and with the 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
	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
chart.subtitle	For Material Charts (/chart/interactive/docs/gallery/barchart#Material), this option specifies the subtitle. Only Material Charts support subtitles. **Type:** string **Default:** null
chart.title	For Material Charts (/chart/interactive/docs/gallery/barchart#Material), this option specifies the title. **Type:** string **Default:** null
colors	The colors to use for the chart elements. An array of strings, where each element is an HTML color string, for example: `colors:['red','#004411']`. **Type:** Array of strings **Default:** default colors
dataOpacity	The transparency of data points, with 1.0 being completely opaque and 0.0 fully transparent. In scatter, histogram, bar, and column charts, this refers to the visible data: dots in the scatter chart and rectangles in the others. In charts where *selecting data* creates a dot, such as the line and area charts, this refers to the circles that appear upon hover or selection. The combo chart exhibits both behaviors, and this option has no effect on other charts. (To change the opacity of a trendline, see trendline opacity (https://developers.google.com/chart/interactive/docs/gallery/trendlines#Example4).) **Type:** number **Default:** 1.0
enableInteractivity	Whether the chart throws user-based events or reacts to user interaction. If false, the chart will not throw 'select' or other interaction-based events (but *will* throw ready or error events), and will not display hovertext or otherwise change depending on user input. **Type:** boolean **Default:** true
explorer	The `explorer` option allows users to pan and zoom Google charts. `explorer: {}` provides the default explorer behavior, enabling users to pan horizontally and vertically by dragging, and to zoom in 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 or dates). **Type:** object **Default:** null
explorer.actions	The Google Charts explorer supports three actions: • `dragToPan`: Drag to pan around the chart horizontally and vertically. To pan only along the horizontal axis, use `explorer: { axis: 'horizontal' }`. Similarly for the vertical axis. • `dragToZoom`: The explorer's default behavior is to zoom in and out when the user scrolls. If `explorer: { actions: ['dragToZoom', 'rightClickToReset'] }` is used, dragging across a rectangular area zooms into that area. We recommend using `rightClickToReset` whenever `dragToZoom` is used. See `explorer.maxZoomIn`, `explorer.maxZoomOut`, and `explorer.zoomDelta` for zoom customizations.

	• `rightClickToReset`: Right clicking on the chart returns it to the original pan and zoom level. **Type:** Array of strings **Default:** ['dragToPan', 'rightClickToReset']
explorer.axis	By default, users can pan both horizontally and vertically when the `explorer` option is used. If you want to users to only pan horizontally, use `explorer: { axis: 'horizontal' }`. Similarly, `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. To ensure that users don't pan 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 able to zoom in enough that they'll see only 25% of the original view. Setting `explorer: { maxZoomIn: .5 }` would let users 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 able to zoom out far enough that the chart will take up only 1/4 of the available space. Setting `explorer: { maxZoomOut: 8 }` would let users zoom out far enough that the chart would take up only 1/8 of the available space. **Type:** number **Default:** 4
explorer.zoomDelta	When users zoom in or out, `explorer.zoomDelta` determines how much they zoom by. The smaller the number, the smoother and slower the zoom. **Type:** number **Default:** 1.5
focusTarget	The type of the entity that receives focus on mouse hover. Also affects which entity is selected by mouse click, and which data table element is associated with events. Can be one of the following: • 'datum' - Focus on a single data point. Correlates to a cell in the data table. • 'category' - Focus on a grouping of all data points along the major axis. Correlates to a row in the data table. In focusTarget 'category' the tooltip displays all the category values. This may be useful for comparing values of different series. **Type:** string **Default:** 'datum'
fontSize	The default font size, in pixels, of all text in the chart. You can override this using properties for specific chart elements. **Type:** number **Default:** automatic
fontName	The default font face for all text in the chart. You can override this using properties for specific chart elements. **Type:** string **Default:** 'Arial'
forceIFrame	Draws the chart inside an inline frame. (Note that on IE8, this option is ignored; all IE8 charts are drawn in i-frames.) **Type:** boolean **Default:** false
	An object with members to configure various horizontal axis elements. To specify properties of this 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.

This option is only supported for a `continuous` (/chart/interactive/docs/customizing_axes#Terminology) axis.

Type: number
Default: automatic

hAxis.baselineColor

The color of the baseline for the horizontal axis. Can be any HTML color string, for example: `'red'` or `'#00cc00'`.

This option is only supported for a `continuous` (/chart/interactive/docs/customizing_axes#Terminology) axis.

Type: number
Default: 'black'

hAxis.direction

The direction in which the values along the horizontal axis grow. Specify `-1` to reverse the order of the values.

Type: 1 or -1
Default: 1

hAxis.format

A format string for numeric or date axis labels.

For number axis labels, this is a subset of the decimal formatting ICU pattern set (http://icu-project.org/apiref/icu4c/classDecimalFormat.html#_details). For instance, `{format:'#,###%'}` will display values "1,000%", "750%", and "50%" for values 10, 7.5, and 0.5. You can also supply any of the following:

- `{format: 'none'}`: displays numbers with no formatting (e.g., 8000000)
- `{format: 'decimal'}`: displays numbers with thousands separators (e.g., 8,000,000)
- `{format: 'scientific'}`: displays numbers in scientific notation (e.g., 8e6)
- `{format: 'currency'}`: displays numbers in the local currency (e.g., $8,000,000.00)
- `{format: 'percent'}`: displays numbers as percentages (e.g., 800,000,000%)
- `{format: 'short'}`: displays abbreviated numbers (e.g., 8M)
- `{format: 'long'}`: displays numbers as full words (e.g., 8 million)

For date axis labels, this is a subset of the date formatting ICU pattern set (http://icu-project.org/apiref/icu4c/classSimpleDateFormat.html#_details). For instance, `{format:'MMM d, y'}` will display the value "Jul 1, 2011" for the date of July first in 2011.

The actual formatting applied to the label is derived from the locale the API has been loaded with. For more details, see loading charts with a specific locale (/chart/interactive/docs/library_loading_enhancements#loadwithlocale).

In computing tick values and gridlines, several alternative combinations of all the relevant gridline options will be considered and alternatives will be rejected if the formatted tick labels would be duplicated or overlap. So you can specify `format:"#"` if you want to only show integer tick values, but be aware that if no alternative satisfies this condition, no gridlines or ticks will be shown.

This option is only supported for a `continuous` (/chart/interactive/docs/customizing_axes#Terminology) axis.

Type: string
Default: auto

hAxis.gridlines

An object with properties to configure the gridlines on the horizontal axis. Note that horizontal axis gridlines are drawn vertically. To specify properties of this object, you can use object literal notation, as shown here:

```
{color: '#333', minSpacing: 20}
```

This option is only supported for a **continuous** (/chart/interactive/docs/customizing_axes#Terminology) axis.

Type: object
Default: null

hAxis.gridlines.color

The color of the horizontal gridlines inside the chart area. Specify a valid HTML color string.

Type: string
Default: '#CCC'

hAxis.gridlines.count

The approximate number of horizontal gridlines inside the chart area. If you specify a positive number for `gridlines.count`, it will be used to compute the `minSpacing` between gridlines. You can specify a value of `1` to only draw one gridline, or `0` to draw no gridlines. Specify -1, which is the default, to automatically compute the number of gridlines based on other options.

Type: number
Default: -1

hAxis.gridlines.interval

An array of sizes (as data values, not pixels) between adjacent gridlines. This option is only for numeric axes at this time, but it is analogous to the `gridlines.units.<unit>.interval` options which are used only for dates and times. For linear scales, the default is `[1, 2, 2.5, 5]` which means the gridline values can fall on every unit (1), on even units (2), or on multiples of 2.5 or 5. Any power of 10 times these values is also considered (e.g. [10, 20, 25, 50] and [.1, .2, .25, .5]). For log scales, the default is `[1, 2, 5]`.

Type: number between 1 and 10, not including 10.
Default: computed

hAxis.gridlines.minSpacing

The minimum screen space, in pixels, between hAxis major gridlines. The default for major gridlines is `40` for linear scales, and `20` for log scales. If you specify the `count` and not the `minSpacing`, the minSpacing is computed from the count. And conversely, if you specify the `minSpacing` and not the `count`, the count is computed from the minSpacing. If you specify both, the `minSpacing` overrides.

Type: number
Default: computed

hAxis.gridlines.multiple

All gridline and tick values must be a multiple of this option's value. Note that, unlike for intervals, powers of 10 times the multiple are not considered. So you can force ticks to be integers by specifying `gridlines.multiple = 1`, or force ticks to be multiples of 1000 by specifying `gridlines.multiple = 1000`.

Type: number
Default: 1

hAxis.gridlines.units

Overrides the default format for various aspects of date/datetime/timeofday data types when used with chart computed gridlines. Allows formatting for years, months, days, hours, minutes, 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 (/chart/interactive/docs/datesandtimes#axesgridlinesticks). **Type:** object **Default:** null
hAxis.minorGridlines	An object with members to configure the minor gridlines on the horizontal axis, similar to the hAxis.gridlines option. This option is only supported for a `continuous` (/chart/interactive/docs/customizing_axes#Terminology) axis. **Type:** object **Default:** null
hAxis.minorGridlines.color	The color of the horizontal minor gridlines inside the chart area. Specify a valid HTML color string. **Type:** string **Default:** A blend of the gridline and background colors
hAxis.minorGridlines.count	The `minorGridlines.count` option is mostly deprecated, except for disabling minor gridlines by setting the count to 0. The number of minor gridlines now depends entirely on the interval between major gridlines (see `hAxis.gridlines.interval`) and the minimum required space (see `hAxis.minorGridlines.minSpacing`). **Type:** number **Default:**1
hAxis.minorGridlines.interval	The minorGridlines.interval option is like the major gridlines interval option, but the interval that is chosen will always be an even divisor of the major gridline interval. The default interval for linear scales is `[1, 1.5, 2, 2.5, 5]`, and for log scales is `[1, 2, 5]`. **Type:** number **Default:**1
hAxis.minorGridlines.minSpacing	The minimum required space, in pixels, between adjacent minor gridlines, and between minor and major gridlines. The default value is 1/2 the minSpacing of major gridlines for linear scales, and 1/5 the minSpacing for log scales. **Type:** number **Default:**computed
hAxis.minorGridlines.multiple	Same as for major `gridlines.multiple`. **Type:** number **Default:** 1
hAxis.minorGridlines.units	Overrides the default format for various aspects of date/datetime/timeofday data types when used with chart computed minorGridlines. Allows formatting for years, months, days, hours, minutes, 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 (/chart/interactive/docs/datesandtimes#axesgridlinesticks).

Type: object
Default: null

hAxis.logScale

`hAxis` property that makes the horizontal axis a logarithmic scale (requires all values to be positive). Set to true for yes.

This option is only supported for a `continuous` (/chart/interactive/docs/customizing_axes#Terminology) axis.

Type: boolean
Default: false

hAxis.scaleType

`hAxis` property that makes the horizontal axis a logarithmic scale. Can be one of the following:

- null - No logarithmic scaling is performed.
- 'log' - Logarithmic scaling. Negative and zero values are not plotted. This option is the same as setting `hAxis: { logscale: true }`.
- 'mirrorLog' - Logarithmic scaling in which negative and zero values are plotted. The plotted value of a negative number is the negative of the log of the absolute value. Values close to 0 are plotted on a linear scale.

This option is only supported for a `continuous` (/chart/interactive/docs/customizing_axes#Terminology) axis.

Type: string
Default: null

hAxis.textPosition

Position of the horizontal axis text, relative to the chart area. Supported values: 'out', 'in', 'none'.

Type: string
Default: 'out'

hAxis.textStyle

An object that specifies the horizontal axis 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 `'#00cc00'`. Also see `fontName` and `fontSize`.

Type: object
Default: `{color: 'black', fontName: <global-font-name>, fontSize: <global-font-size>}`

hAxis.ticks

Replaces the automatically generated X-axis ticks with the specified array. Each element of the array should be either a valid tick value (such as a number, date, datetime, or timeofday), or an object. If it's an object, it should have a `v` property for the tick value, and an optional `f` property containing the literal string to be displayed as the label.

The viewWindow will be automatically expanded to include the min and max ticks unless you specify a `viewWindow.min` or `viewWindow.max` to override.

Examples:

- `hAxis: { ticks: [5,10,15,20] }`
- `hAxis: { ticks: [{v:32, f:'thirty two'}, {v:64, f:'sixty four'}] }`
- `hAxis: { ticks: [new Date(2014,3,15), new Date(2013,5,15)] }`
- `hAxis: { ticks: [16, {v:32, f:'thirty two'}, {v:64, f:'sixty four'}, 128] }`

This option is only supported for a `continuous` (/chart/interactive/docs/customizing_axes#Terminology) axis.

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 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 `'#00cc00'`. Also see `fontName` and `fontSize`.

Type: object
Default: `{color: 'black', fontName: <global-font-name>, fontSize: <global-font-size>}`

hAxis.allowContainerBoundaryTextCutoff

If false, will hide outermost labels rather than allow them to be cropped by the chart container. If true, will allow label cropping.

This option is only supported for a `discrete` (/chart/interactive/docs/customizing_axes#Terminology) axis.

Type: boolean
Default: false

hAxis.slantedText

If true, draw the horizontal axis text at an angle, to help fit more text along the axis; if false, draw horizontal axis text upright. Default behavior is to slant text if it cannot all fit when drawn upright. Notice that this option is available only when the `hAxis.textPosition` is set to 'out' (which is the default). The default is `false` for dates and times.

Type: boolean
Default: automatic

hAxis.slantedTextAngle

The angle of the horizontal axis text, if it's drawn slanted. Ignored if `hAxis.slantedText` is `false`, or is in auto mode, and the chart decided to draw the text horizontally. If the angle is positive, the rotation is counter-clockwise, and if negative, it is clockwise.

Type: number, -90—90
Default: 30

hAxis.maxAlternation

Maximum number of levels of horizontal axis text. If axis text labels become too crowded, the server might shift neighboring labels up or down in order to fit labels closer together. This value specifies the most number of levels to use; the server can use fewer levels, if labels can fit without overlapping. For dates and times, the default is 1.

Type: number
Default: 2

hAxis.maxTextLines

Maximum number of lines allowed for the text labels. Labels can span multiple lines if they are too long, and the number of lines is, by default, limited by the height of the available space.

Type: number
Default: auto

hAxis.minTextSpacing

Minimum horizontal spacing, in pixels, allowed between two adjacent text labels. If the labels are spaced too densely, or they are too long, the spacing can drop below this threshold, and in this case one of the label-unclutter measures will be applied (e.g, truncating the labels or dropping some of them).

Type: number
Default: The value of `hAxis.textStyle.fontSize`

hAxis.showTextEvery	How many horizontal axis labels to show, where 1 means show every label, 2 means show every other label, and so on. Default is to try to show as many labels as possible without overlapping. **Type:** number **Default:** automatic
hAxis.maxValue	Moves the max value of the horizontal axis to the specified value; this will be rightward in most charts. Ignored if this is set to a value smaller than the maximum x-value of the data. `hAxis.viewWindow.max` overrides this property. This option is only supported for a `continuous` (/chart/interactive/docs/customizing_axes#Terminology) axis. **Type:** number **Default:** automatic
hAxis.minValue	Moves the min value of the horizontal axis to the specified value; this will be leftward in most charts. Ignored if this is set to a value greater than the minimum x-value of the data. `hAxis.viewWindow.min` overrides this property. This option is only supported for a `continuous` (/chart/interactive/docs/customizing_axes#Terminology) axis. **Type:** number **Default:** automatic
hAxis.viewWindowMode	Specifies how to scale the horizontal axis to render the values within the chart area. The following string values are supported: • 'pretty' - Scale the horizontal values so that the maximum and minimum data values are rendered a bit inside the left and right of the chart area. The viewWindow is expanded to the nearest major gridline for numbers, or the nearest minor gridline for dates and times. • 'maximized' - Scale the horizontal values so that the maximum and minimum data values touch the left and right of the chart area. This will cause `haxis.viewWindow.min` and `haxis.viewWindow.max` to be ignored. • 'explicit' - A deprecated option for specifying the left and right scale values of the chart area. (Deprecated because it's redundant with `haxis.viewWindow.min` and `haxis.viewWindow.max`.) Data values outside these values will be cropped. You must specify an `hAxis.viewWindow` object describing the maximum and minimum values to show. This option is only supported for a `continuous` (/chart/interactive/docs/customizing_axes#Terminology) axis. **Type:** string **Default:** Equivalent to 'pretty', but `haxis.viewWindow.min` and `haxis.viewWindow.max` take precedence if used.
hAxis.viewWindow	Specifies the cropping range of the horizontal axis. **Type:** object **Default:** null
hAxis.viewWindow.max	• For a `continuous` (/chart/interactive/docs/customizing_axes#Terminology) axis: The maximum horizontal data value to render. • For a `discrete` (/chart/interactive/docs/customizing_axes#Terminology) axis: The zero-based row index where the cropping window ends. Data points at this index and higher will be cropped out. In conjunction with `vAxis.viewWindowMode.min`, it defines a half-opened range [min, max) that denotes the element indices to display. In other words, every index such that `min <= index < max` will be displayed. Ignored when `hAxis.viewWindowMode` is 'pretty' or 'maximized'. **Type:** number **Default:** auto
hAxis.viewWindow.min	• For a `continuous` (/chart/interactive/docs/customizing_axes#Terminology) axis: The minimum horizontal data value to render.

- For a `discrete` (/chart/interactive/docs/customizing_axes#Terminology) axis:

  The zero-based row index where the cropping window begins. Data points at indices lower than this will be cropped out. In conjunction with `vAxis.viewWindowMode.max`, it defines a half-opened range [min, max) that denotes the element indices to display. In other words, every index such that `min <= index < max` will be displayed.

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

isStacked

If set to true, stacks the elements for all series at each domain value. **Note:** In Column (/chart/interactive/docs/gallery/columnchart), Area (/chart/interactive/docs/gallery/areachart), and SteppedArea (/chart/interactive/docs/gallery/steppedareachart) charts, Google Charts reverses the order of legend items to better correspond with the stacking of the series elements (E.g. series 0 will be the bottom-most legend item). This **does not** apply to Bar (/chart/interactive/docs/gallery/barchart) Charts.

The `isStacked` option also supports 100% stacking, where the stacks of elements at each domain value are rescaled to add up to 100%.

The options for `isStacked` are:

- `false` — elements will not stack. This is the default option.
- `true` — stacks elements for all series at each domain value.
- `'percent'` — stacks elements for all series at each domain value and rescales them such that they add up to 100%, with each element's value calculated as a percentage of 100%.
- `'relative'` — stacks elements for all series at each domain value and rescales them such that they add up to 1, with each element's value calculated as a fraction of 1.
- `'absolute'` — functions the same as `isStacked: true`.

For 100% stacking, the calculated value for each element will appear in the tooltip after its actual value.

The target axis will default to tick values based on the relative 0-1 scale as fractions of 1 for `'relative'`, and 0-100% for `'percent'` (**Note:** when using the `'percent'` option, the axis/tick values are displayed as percentages, however the actual values are the relative 0-1 scale values. This is because the percentage axis ticks are the result of applying a format of "#.##%" to the relative 0-1 scale values. When using `isStacked: 'percent'`, be sure to specify any ticks/gridlines using the relative 0-1 scale values). You can customize the gridlines/tick values and formatting using the appropriate `hAxis/vAxis` options.

100% stacking only supports data values of type `number`, and must have a baseline of zero.

Type: boolean/string
Default: false

legend

An object with members to configure various aspects of the legend. To specify properties of this object, you can use object literal notation, as shown here:

```
{position: 'top', textStyle: {color: 'blue', fontSize: 16}}
```

Type: object
Default: null

legend.pageIndex

Initial selected zero-based page index of the legend.

Type: number
Default: 0

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 associated with it. So if you want 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` option.
- 'top' - Above the chart.

Type: string
Default: 'right'

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 -- of the legend. For example, in a 'right' legend, 'start' and 'end' are at the top and bottom, respectively; for a 'top' legend, 'start' and 'end' would be at the left and right of the area, respectively.

The default value depends on the legend's position. For 'bottom' legends, the default is 'center'; other legends default to 'start'.

Type: string
Default: automatic

legend.textStyle

An object that specifies the legend text style. The object has this format:

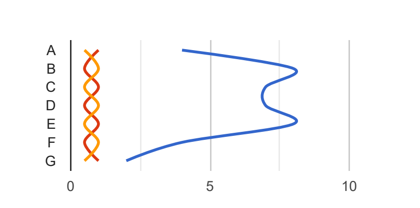
```
{ color: <string>,
  fontName: <string>,
  fontSize: <number>,
  bold: <boolean>,
  italic: <boolean> }
```

The `color` can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

Type: object
Default: `{color: 'black', fontName: <global-font-name>, fontSize: <global-font-size>}`

orientation

The orientation of the chart. When set to `'vertical'`, rotates the axes of the chart so that (for instance) a column chart becomes a bar chart, and an area chart grows rightward instead of up:



Type: string
Default: 'horizontal'

reverseCategories

If set to true, will draw series from right to left. The default is to draw left-to-right.

This option is only supported for a **discrete** (/chart/interactive/docs/customizing_axes#Terminology) **major** (/chart/interactive/docs/customizing_axes#Terminology) axis.

Type: boolean

Default: false

series

An array of objects, each describing the format of the corresponding series in the chart. To use default values for a series, specify an empty object {}. If a series or a value is not specified, the global value will be used. Each object supports the following properties:

- **`annotations`** - An object to be applied to annotations for this series. This can be used to control, for instance, the **`textStyle`** for the series:

  ```
  series: {
    0: {
      annotations: {
        textStyle: {fontSize: 12, color: 'red' }
      }
    }
  }
  ```

  See the various **`annotations`** options for a more complete list of what can be customized.
- **`color`** - The color to use for this series. Specify a valid HTML color string.
- **`targetAxisIndex`** - Which axis to assign this series to, where 0 is the default axis, and 1 is the opposite axis. Default value is 0; set to 1 to define a chart where different series are rendered against different axes. At least one series much be allocated to the default axis. You can define a different scale for different axes.
- **`visibleInLegend`** - A boolean value, where true means that the series should have a legend entry, and false means that it should not. Default is true.

You can specify either an array of objects, each of which applies to the series in the order given, or you can specify an object where each child has a numeric key indicating which series it applies to. For example, the following two declarations are identical, and declare the first series as black and absent 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 achieve a specific chart behavior or visual effect. Currently only one theme is available:

- 'maximized' - Maximizes the area of the chart, and draws the legend and all of the labels inside the chart area. Sets the following options:

  ```
  chartArea: {width: '100%', height: '100%'},
  legend: {position: 'in'},
  titlePosition: 'in', axisTitlesPosition: 'in',
  hAxis: {textPosition: 'in'}, vAxis: {textPosition: 'in'}
  ```

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 values:

- 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 `'#00cc00'`. Also see `fontName` and `fontSize`.

Type: object
Default: `{color: 'black', fontName: <global-font-name>, fontSize: <global-font-size>}`

tooltip

An object with members to configure various tooltip elements. To specify properties of this object, you 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 bounds of the chart on all sides.

Note: This only applies to HTML tooltips. If this is enabled with SVG tooltips, any overflow outside of the chart bounds will be cropped. See Customizing Tooltip Content (/chart/interactive/docs/customizing_tooltip_content) for more details.

Type: boolean
Default: false

tooltip.isHtml

If set to true, use HTML-rendered (rather than SVG-rendered) tooltips. See Customizing Tooltip Content (/chart/interactive/docs/customizing_tooltip_content) for more details.

★ **Note:** customization of the HTML tooltip content via the tooltip column data role (/chart/interactive/docs/roles#tooltiprole) is **not** supported by the Bubble Chart (/chart/interactive/docs/gallery/bubblechart) visualization.

Type: boolean
Default: false

tooltip.showColorCode

If true, show colored squares next to the series information in the tooltip. The default is true when `focusTarget` is set to 'category', otherwise the default is false.

Type: boolean
Default: automatic

tooltip.textStyle

An object that specifies the tooltip 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 `'#00cc00'`. Also see `fontName` and `fontSize`.

Type: object
Default: `{color: 'black', fontName: <global-font-name>, fontSize: <global-font-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 element.
- 'none' - The tooltip will not be displayed.
- 'selection' - The tooltip will be displayed when the user selects the element.

Type: string
Default: 'focus'

trendlines

Displays trendlines (https://developers.google.com/chart/interactive/docs/gallery/trendlines) on the charts that support them. By default, linear trendlines are used, but this can be customized with the `trendlines.n.type` option.

Trendlines are specified on a per-series basis, so most of the time your options will look like this:

```
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/interactive/docs/gallery/trendlines), 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/gallery/trendlines) of `type: 'polynomial'`, the degree of the polynomial (2 for quadratic, 3 for cubic, and so on). (The default degree may change from 3 to 2 in an upcoming release of Google Charts.)

Type: number
Default: 3

f set, the trendline (https://developers.google.com/chart/interactive/docs/gallery/trendlines) will 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/trendlines), in pixels. **Type:** number **Default:** 2
trendlines.n.opacity	The transparency of the trendline (https://developers.google.com/chart/interactive/docs/gallery/trendlines), from 0.0 (transparent) to 1.0 (opaque). **Type:** number **Default:** 1.0
trendlines.n.pointSize	Trendlines (https://developers.google.com/chart/interactive/docs/gallery/trendlines) are constucted by stamping a bunch of dots on the chart; this rarely-needed option lets you customize the size of the dots. The trendline's `lineWidth` option will usually be preferable. However, you'll need this option if you're using the global `pointSize` option and want a different point size for your trendlines. **Type:** number **Default:** 1
trendlines.n.pointsVisible	Trendlines (https://developers.google.com/chart/interactive/docs/gallery/trendlines) are constucted by stamping a bunch of dots on the chart. The trendline's `pointsVisible` option determines whether 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/trendlines#r2) in the legend or trendline tooltip. **Type:** boolean **Default:** false
trendlines.n.type	Whether the trendlines (https://developers.google.com/chart/interactive/docs/gallery/trendlines) is `'linear'` (the default), `'exponential'`, or `'polynomial'`. **Type:** string **Default:** linear
trendlines.n.visibleInLegend	Whether the trendline (https://developers.google.com/chart/interactive/docs/gallery/trendlines) equation appears in the legend. (It will appear in the trendline tooltip.) **Type:** boolean **Default:** false
vAxes	Specifies properties for individual vertical axes, if the chart has multiple vertical axes. Each child object is a `vAxis` object, and can contain all the properties supported by `vAxis`. These property values override any global settings for the same property. To specify a chart with multiple vertical axes, first define a new axis using `series.targetAxisIndex`, then configure the axis using `vAxes`. The following example assigns series 2 to the right axis and specifies a custom title and text style for it:

```
{
  series: {
    2: {
      targetAxisIndex:1
    }
  },
  vAxes: {
    1: {
      title:'Losses',
      textStyle: {color: 'red'}
    }
  }
}
```

This property can be either an object or an array: the object is a collection of objects, each with a numeric label that specifies the axis that it defines--this is the format shown above; the array is an array of objects, one per axis. For example, the following array-style notation is identical to the `vAxis` object shown above:

```
vAxes: [
  {}, // Nothing specified for axis 0
  {
    title:'Losses',
    textStyle: {color: 'red'} // Axis 1
  }
]
```

Type: Array of object, or object with child objects
Default: null

vAxis

An object with members to configure various vertical axis elements. To specify properties of this object, 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 baseline is larger than the highest grid line or smaller than the lowest grid line, it will be rounded to the closest gridline.

Type: number
Default: automatic

vAxis.baselineColor

Specifies the color of the baseline for the vertical axis. Can be any HTML color string, for example: `'red'` or `'#00cc00'`.

Type: number
Default: 'black'

vAxis.direction

The direction in which the values along the vertical axis grow. By default, low values are on the bottom of the chart. Specify `-1` to reverse the order of the values.

Type: 1 or -1
Default: 1

vAxis.format

A format string for numeric axis labels. This is a subset of the ICU pattern set (http://icu-project.org/apiref/icu4c/classDecimalFormat.html#_details). For instance, `{format:'#,###%'}` will display values "1,000%", "750%", and "50%" for values 10, 7.5, and 0.5. You can also supply any of the following:

- `{format: 'none'}`: displays numbers with no formatting (e.g., 8000000)
- `{format: 'decimal'}`: displays numbers with thousands separators (e.g., 8,000,000)
- `{format: 'scientific'}`: displays numbers in scientific notation (e.g., 8e6)
- `{format: 'currency'}`: displays numbers in the local currency (e.g., $8,000,000.00)
- `{format: 'percent'}`: displays numbers as percentages (e.g., 800,000,000%)
- `{format: 'short'}`: displays abbreviated numbers (e.g., 8M)

- `{format: 'long'}`: displays numbers as full words (e.g., 8 million)

The actual formatting applied to the label is derived from the locale the API has been loaded with. For more details, see loading charts with a specific locale (/chart/interactive/docs/library_loading_enhancements#loadwithlocale).

In computing tick values and gridlines, several alternative combinations of all the relevant gridline options will be considered and alternatives will be rejected if the formatted tick labels would be duplicated or overlap. So you can specify `format:"#"` if you want to only show integer tick values, but be aware that if no alternative satisfies this condition, no gridlines or ticks will be shown.

Type: string
Default: auto

vAxis.gridlines

An object with members to configure the gridlines on the vertical axis. Note that vertical axis gridlines are drawn horizontally. To specify properties of this object, you can use object literal notation, as shown here:

```
{color: '#333', minSpacing: 20}
```

Type: object
Default: null

vAxis.gridlines.color

The color of the vertical gridlines inside the chart area. Specify a valid HTML color string.

Type: string
Default: '#CCC'

vAxis.gridlines.count

The approximate number of horizontal gridlines inside the chart area. If you specify a positive number for `gridlines.count`, it will be used to compute the `minSpacing` between gridlines. You can specify a value of `1` to only draw one gridline, or `0` to draw no gridlines. Specify -1, which is the default, to automatically compute the number of gridlines based on other options.

Type: number
Default: -1

vAxis.gridlines.interval

An array of sizes (as data values, not pixels) between adjacent gridlines. This option is only for numeric axes at this time, but it is analogous to the `gridlines.units.<unit>.interval` options which are used only for dates and times. For linear scales, the default is `[1, 2, 2.5, 5]` which means the gridline values can fall on every unit (1), on even units (2), or on multiples of 2.5 or 5. Any power of 10 times these values is also considered (e.g. [10, 20, 25, 50] and [.1, .2, .25, .5]). For log scales, the default is `[1, 2, 5]`.

Type: number between 1 and 10, not including 10.
Default: computed

vAxis.gridlines.minSpacing

The minimum screen space, in pixels, between hAxis major gridlines. The default for major gridlines is `40` for linear scales, and `20` for log scales. If you specify the `count` and not the `minSpacing`, the minSpacing is computed from the count. And conversely, if you specify the `minSpacing` and not the `count`, the count is computed from the minSpacing. If you specify both, the `minSpacing` overrides.

Type: number
Default: computed

vAxis.gridlines.multiple

All gridline and tick values must be a multiple of this option's value. Note that, unlike for intervals, powers of 10 times the multiple are not considered. So you can force ticks to be integers by specifying `gridlines.multiple = 1`, or force ticks to be multiples of 1000 by specifying `gridlines.multiple = 1000`.

Type: number
Default: 1

vAxis.gridlines.units

Overrides the default format for various aspects of date/datetime/timeofday data types when used with chart computed gridlines. Allows formatting for years, months, days, hours, minutes, 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 (/chart/interactive/docs/datesandtimes#axesgridlinesticks).

Type: object
Default: null

vAxis.minorGridlines

An object with members to configure the minor gridlines on the vertical axis, similar to the vAxis.gridlines option.

Type: object
Default: null

vAxis.minorGridlines.color

The color of the vertical minor gridlines inside the chart area. Specify a valid HTML color string.

Type: string
Default: A blend of the gridline and background colors

vAxis.minorGridlines.count

The minorGridlines.count option is mostly deprecated, except for disabling minor gridlines by setting the count to 0. The number of minor gridlines depends on the interval between major gridlines (see vAxis.gridlines.interval) and the minimum required space (see vAxis.minorGridlines.minSpacing).

Type: number
Default: 1

vAxis.minorGridlines.interval

The minorGridlines.interval option is like the major gridlines interval option, but the interval that is chosen will always be an even divisor of the major gridline interval. The default interval for linear scales is `[1, 1.5, 2, 2.5, 5]`, and for log scales is `[1, 2, 5]`.

Type: number
Default:1

vAxis.minorGridlines.minSpacing

The minimum required space, in pixels, between adjacent minor gridlines, and between minor and major gridlines. The default value is 1/2 the minSpacing of major gridlines for linear scales, and 1/5 the minSpacing for log scales.

Type: number
Default:computed

vAxis.minorGridlines.multiple

Same as for major `gridlines.multiple`.

Type: number
Default: 1

vAxis.minorGridlines.units

Overrides the default format for various aspects of date/datetime/timeofday data types when used with chart computed minorGridlines. Allows formatting for years, months, days, hours, minutes, 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 (/chart/interactive/docs/datesandtimes#axesgridlinesticks).

Type: object
Default: null

vAxis.logScale

If true, makes the vertical axis a logarithmic scale. Note: All values must be positive.

Type: boolean
Default: false

vAxis.scaleType

`vAxis` property that makes the vertical axis a logarithmic scale. Can be one of the following:

- null - No logarithmic scaling is performed.
- 'log' - Logarithmic scaling. Negative and zero values are not plotted. This option is the same as setting `vAxis: { logscale: true }`.
- 'mirrorLog' - Logarithmic scaling in which negative and zero values are plotted. The plotted value of a negative number is the negative of the log of the absolute value. Values close to 0 are plotted on a linear scale.

This option is only supported for a `continuous` (/chart/interactive/docs/customizing_axes#Terminology) axis.

Type: string
Default: null

vAxis.textPosition

Position of the vertical axis text, relative to the chart area. Supported values: 'out', 'in', 'none'.

Type: string
Default: 'out'

vAxis.textStyle

An object that specifies the vertical axis 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 `'#00cc00'`. Also see `fontName` and `fontSize`.

Type: object
Default: `{color: 'black', fontName: <global-font-name>, fontSize: <global-font-size>}`

vAxis.ticks

Replaces the automatically generated Y-axis ticks with the specified array. Each element of the array should be either a valid tick value (such as a number, date, datetime, or timeofday), or an object. If it's an object, it should have a `v` property for the tick value, and an optional `f` property containing the literal string to be displayed as the label.

The viewWindow will be automatically expanded to include the min and max ticks unless you specify a `viewWindow.min` or `viewWindow.max` to override.

Examples:

- `vAxis: { ticks: [5,10,15,20] }`
- `vAxis: { ticks: [{v:32, f:'thirty two'}, {v:64, f:'sixty four'}] }`
- `vAxis: { ticks: [new Date(2014,3,15), new Date(2013,5,15)] }`
- `vAxis: { ticks: [16, {v:32, f:'thirty two'}, {v:64, f:'sixty four'}, 128] }`

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 this format:

```
{ color: <string>,
  fontName: <string>,
  fontSize: <number>,
  bold: <boolean>,
  italic: <boolean> }
```

The `color` can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

Type: object
Default: `{color: 'black', fontName: <global-font-name>, fontSize: <global-font-size>}`

vAxis.maxValue

Moves the max value of the vertical axis to the specified value; this will be upward in most charts. Ignored if this is set to a value smaller than the maximum y-value of the data. `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 be downward in most charts. Ignored if this is set to a value greater than the minimum y-value of the data. `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 chart area. The following string values are supported:

- 'pretty' - Scale the vertical values so that the maximum and minimum data values are rendered a bit inside the bottom and top of the chart area. The viewWindow is expanded to the nearest major gridline for numbers, or the nearest minor gridline for dates and times.
- 'maximized' - Scale the vertical values so that the maximum and minimum data values touch the top and bottom of the chart area. This will cause `vaxis.viewWindow.min` and `vaxis.viewWindow.max` to be ignored.
- 'explicit' - A deprecated option for specifying the top and bottom scale values of the chart area. (Deprecated because it's redundant with `vaxis.viewWindow.min` and `vaxis.viewWindow.max`. Data values outside these values will be cropped. You must specify a `vAxis.viewWindow` object describing the maximum and minimum values to show.

Type: string
Default: Equivalent to 'pretty', but `vaxis.viewWindow.min` and `vaxis.viewWindow.max` take 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 vertical 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 `ready` (#Events)event is fired. `Extended description` (/chart/interactive/docs/reference#visdraw). **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 element `id`. The format for `id` isn't yet documented (they're the return values of event handlers (https://developers.google.com/chart/interactive/docs/events)), but here are 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 chart
	`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) chart:
	`cli.getBoundingBox('vAxis#0#gridline')`
	Bounding box of the chart data of a horizontal (e.g., bar) chart:
	`cli.getBoundingBox('hAxis#0#gridline')`
	are relative to the container of the chart. Call this *after* the chart is drawn.

	Return Type: object

`getChartAreaBoundingBox()`	Returns an object containing the left, top, width, and height of the chart content (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 is drawn.
	Return Type: object

`getChartLayoutInterface()`	Returns an object containing information about the onscreen placement of the 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 from the chart container's left edge. Can be negative. Example: `chart.getChartLayoutInterface().getHAxisValue(400)`. 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 (/chart/interactive/docs/printing). **Return Type:** string
`getSelection()`	Returns an array of the selected chart entities. Selectable entities are bars, legend entries and categories. A bar corresponds to a cell in the data table, a legend entry to a column (row index is null), and a category to a row (column index is null). For this chart, only one entity can be selected at any given moment. `Extended description` (/chart/interactive/docs/reference#visgetselection). **Return Type:** Array of selection elements
`getVAxisValue(position, optional_axis_index)`	Returns the logical vertical value at `position`, which is an offset from the chart container's top edge. Can be negative. Example: `chart.getChartLayoutInterface().getVAxisValue(300)`. Call this *after* the chart is drawn. **Type:** number

`getXLocation(position, optional_axis_index)`	Returns the screen x-coordinate of `position` relative to the chart's container. Example: `chart.getChartLayoutInterface().getXLocation(400)`. 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 container. Example: `chart.getChartLayoutInterface().getYLocation(300)`. Call this *after* the chart is drawn. **Return Type:** number
`removeAction(actionID)`	Removes the tooltip action with the requested `actionID` from the chart. **Return Type: none**
`setAction(action)`	Sets a tooltip action to be executed when the user clicks on the action text. The `setAction` method takes an object as its action parameter. This object should specify 3 properties: `id`— the ID of the action being set, `text` —the text that should appear in the tooltip for the action, and `action` — the function that should be run when a user clicks on the action text. Any and all tooltip actions should be set prior to calling the chart's `draw()` method. Extended description (/chart/interactive/docs/reference#vissetaction). **Return Type: none**
`setSelection()`	Selects the specified chart entities. Cancels any previous selection. Selectable entities are bars, legend entries and categories. A bar corresponds to a cell in the data table, a legend entry to a column (row index is null), and a category to a row (column index is null). For this chart, only one entity can be selected at a time. **`Extended description`** (/chart/interactive/docs/reference#vissetselection). **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 (/chart/interactive/docs/basic_interactivity), Handling Events (/chart/interactive/docs/events), and Firing Events (/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
`legendpagination`	Fired when the user clicks legend pagination arrows. Passes back the current legend zero-based page index and the total number of pages. **Properties:** currentPageIndex, totalPages

`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.

Last updated 2021-05-03 UTC.