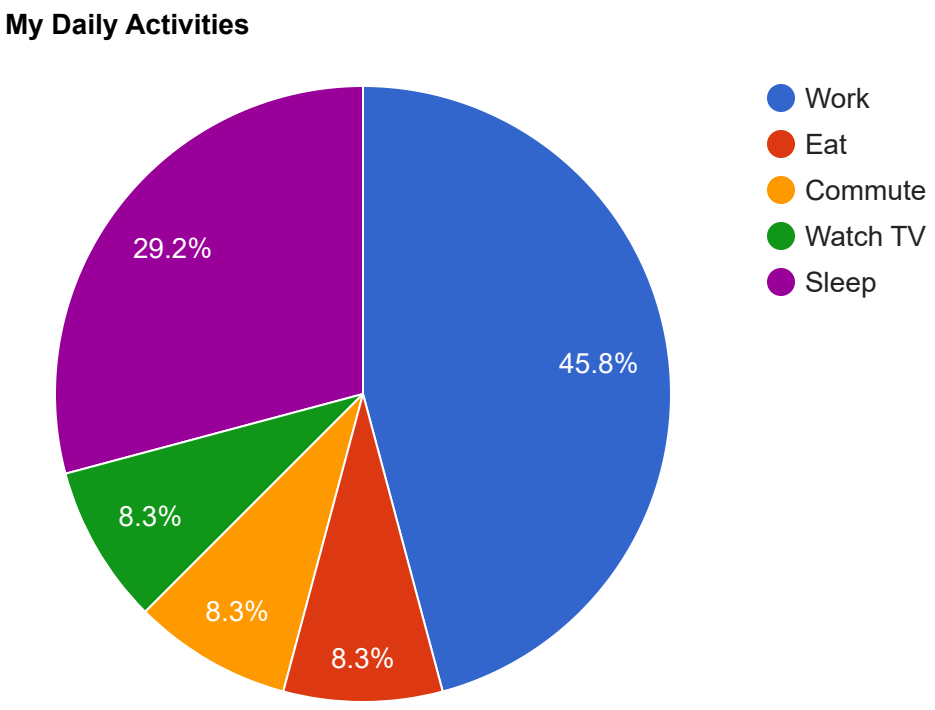
Visualization: Pie Chart

Overview

A pie chart that is rendered within the browser using SVG (http://www.w3.org/Graphics/SVG/) or VML (http://en.wikipedia.org/wiki/Vector_Markup_Language). Displays tooltips when hovering over slices.

Example



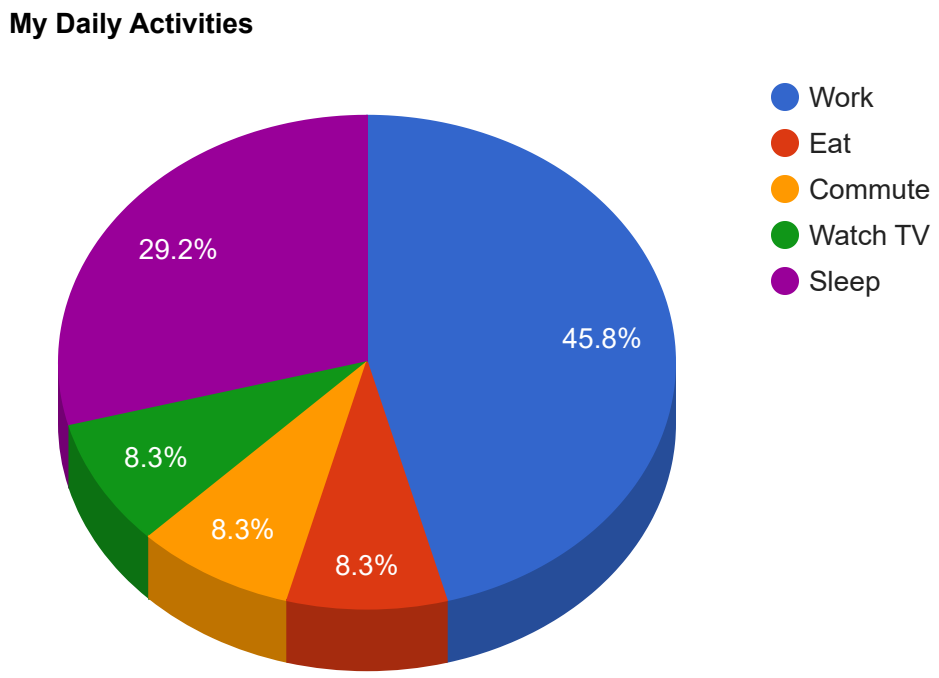
Code it yourself on JSFiddle

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

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

      function drawChart() {

        var data = google.visualization.arrayToDataTable([
          ['Task', 'Hours per Day'],
          ['Work',     11],
          ['Eat',      2],
          ['Commute',  2],
          ['Watch TV', 2],
          ['Sleep',    7]
        ]);

        var options = {
          title: 'My Daily Activities'
        };

        var chart = new google.visualization.PieChart(document.getElementById('p

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

Making a 3D Pie Chart

If you set the **`is3D`** option to **`true`**, your pie chart will be drawn as though it has three dimensions:



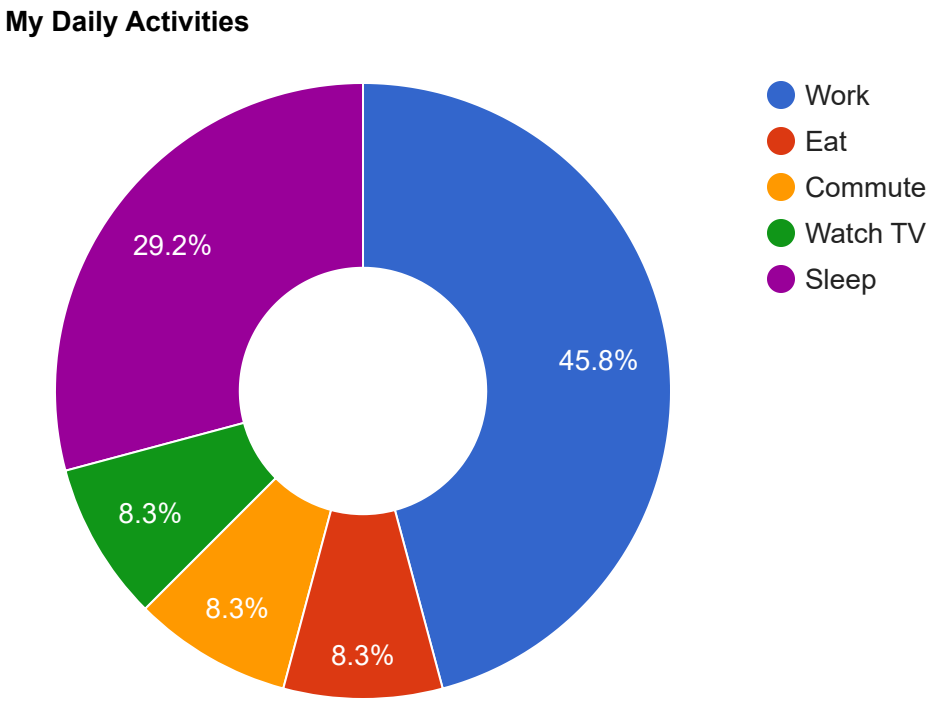
`is3D` is `false` by default, so here we explicitly set it to `true`:

```
<html>
  <head>
    <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js
    <script type="text/javascript">
      google.charts.load("current", {packages:["corechart"]});
      google.charts.setOnLoadCallback(drawChart);
      function drawChart() {
        var data = google.visualization.arrayToDataTable([
          ['Task', 'Hours per Day'],
          ['Work',     11],
          ['Eat',      2],
          ['Commute',  2],
          ['Watch TV', 2],
          ['Sleep',    7]
        ]);
```

```
        var options = {
          title: 'My Daily Activities',
          is3D: true,
        };

        var chart = new google.visualization.PieChart(document.getElementById('p
        chart.draw(data, options);
      }
    </script>
  </head>
  <body>
    <div id="piechart_3d" style="width: 900px; height: 500px;"></div>
  </body>
</html>
```

Making a Donut Chart

A *donut* chart is a pie chart with a hole in the center. You can create donut charts with the **`pieHole`** option:



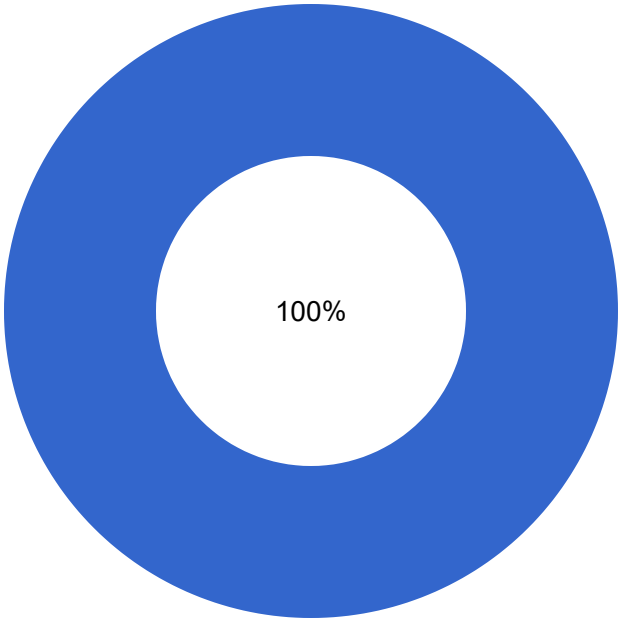
The `pieHole` option should be set to a number between 0 and 1, corresponding to the ratio of radii between the hole and the chart. Numbers between 0.4 and 0.6 will look best on most charts. Values equal to or greater than 1 will be ignored, and a value of 0 will completely shut your piehole.

```
<html>
  <head>
    <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js
    <script type="text/javascript">
      google.charts.load("current", {packages:["corechart"]});
      google.charts.setOnLoadCallback(drawChart);
      function drawChart() {
        var data = google.visualization.arrayToDataTable([
          ['Task', 'Hours per Day'],
          ['Work',     11],
          ['Eat',      2],
          ['Commute',  2],
          ['Watch TV', 2],
```

```
          ['Sleep',    7]
        ]);

        var options = {
          title: 'My Daily Activities',
          pieHole: 0.4,
        };

        var chart = new google.visualization.PieChart(document.getElementById('d
        chart.draw(data, options);
      }
    </script>
  </head>
  <body>
    <div id="donutchart" style="width: 900px; height: 500px;"></div>
  </body>
</html>
```

You can't combine the **`pieHole`** and **`is3D`** options; if you do, **`pieHole`** will be ignored.

Note that Google Charts tries to place the label as close to the center of the slice as possible. If you have a donut chart with just one slice, the center of the slice may fall into the donut hole. In that case, change the color of the label:



Code it yourself on JSFiddle

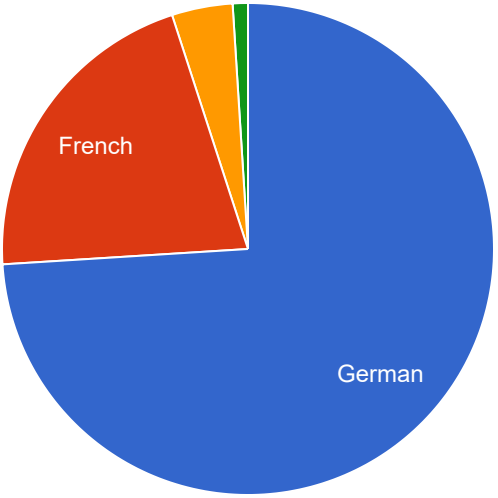
Options Full HTML (#full-html)
(#options)

```
var options = {
  pieHole: 0.5,
  pieSliceTextStyle: {
    color: 'black',
  },
  legend: 'none'
};
```

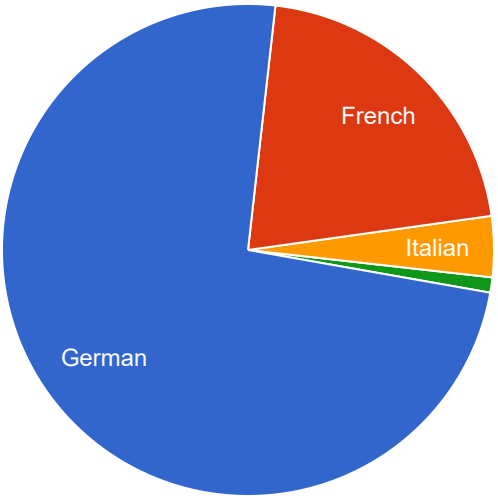
Rotating a Pie Chart

By default, pie charts begin with the left edge of the first slice pointing straight up. You can change that with the `pieStartAngle` option:

Swiss Language Use
(no rotation)



Swiss Language Use
(100 degree rotation)

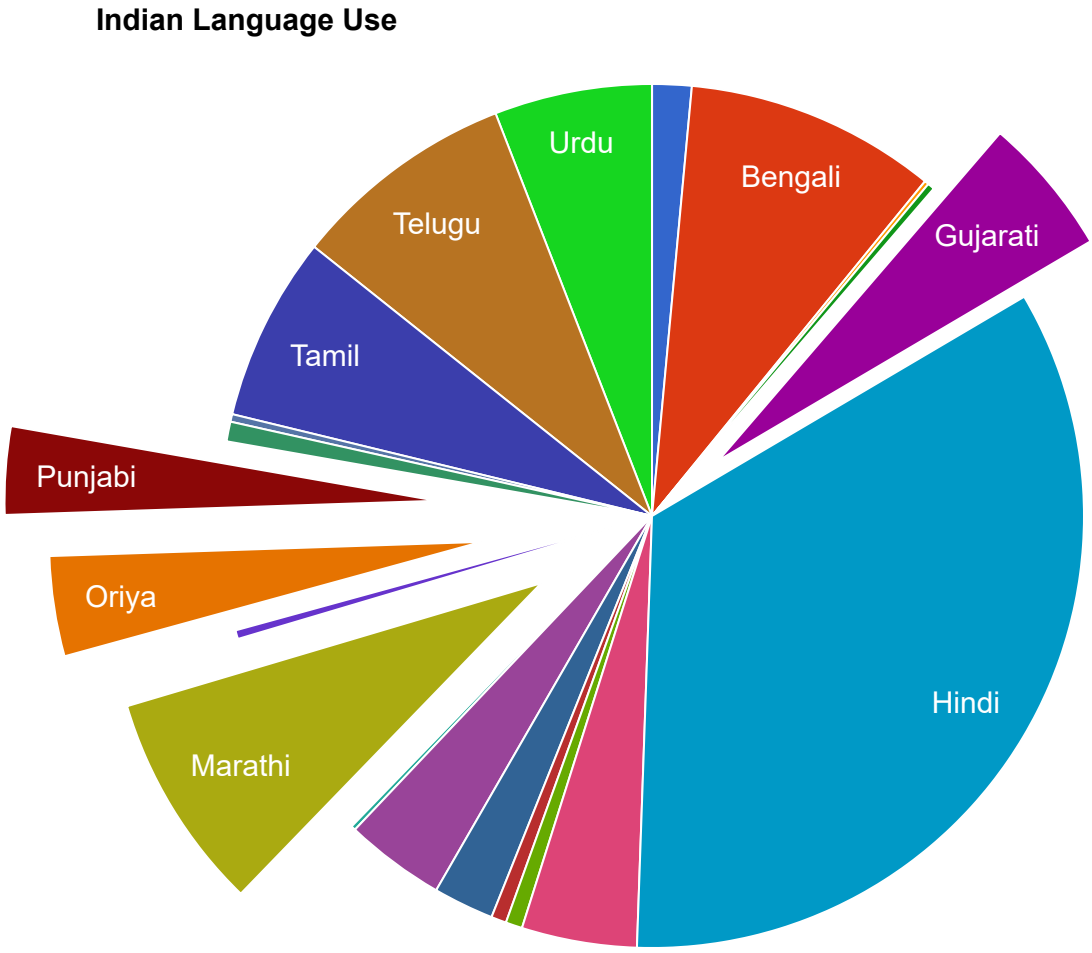


Here, we rotate the chart clockwise 100 degrees with an option of **`pieStartAngle: 100`**. (So chosen because that particular angle happens to make the "Italian" label fit inside the slice.)

```
<html>
  <head>
    <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js
    <script type="text/javascript">
      google.charts.load("current", {packages:["corechart"]});
      google.charts.setOnLoadCallback(drawChart);
      function drawChart() {
        var data = google.visualization.arrayToDataTable([
          ['Language', 'Speakers (in millions)'],
          ['German',  5.85],
          ['French',  1.66],
          ['Italian', 0.316],
          ['Romansh', 0.0791]
        ]);

      var options = {
        legend: 'none',
        pieSliceText: 'label',
        title: 'Swiss Language Use (100 degree rotation)',
        pieStartAngle: 100,
      };

        var chart = new google.visualization.PieChart(document.getElementById('p
        chart.draw(data, options);
      }
    </script>
  </head>
  <body>
    <div id="piechart" style="width: 900px; height: 500px;"></div>
  </body>
</html>
```

Exploding a Slice

You can separate pie slices from the rest of the chart with the `offset` property of the `slices` option:



To separate a slice, create a `slices` object and assign the appropriate slice number an `offset` between 0 and 1. Below, we assign progressively larger offsets to slices 4 (Gujarati), 12 (Marathi), 14 (Oriya), and 15 (Punjabi):

```
<html>
  <head>
    <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js
    <script type="text/javascript">
      google.charts.load("current", {packages:["corechart"]});
      google.charts.setOnLoadCallback(drawChart);
      function drawChart() {
        var data = google.visualization.arrayToDataTable([
          ['Language', 'Speakers (in millions)'],
          ['Assamese', 13], ['Bengali', 83], ['Bodo', 1.4],
          ['Dogri', 2.3], ['Gujarati', 46], ['Hindi', 300],
          ['Kannada', 38], ['Kashmiri', 5.5], ['Konkani', 5],
          ['Maithili', 20], ['Malayalam', 33], ['Manipuri', 1.5],
          ['Marathi', 72], ['Nepali', 2.9], ['Oriya', 33],
          ['Punjabi', 29], ['Sanskrit', 0.01], ['Santhali', 6.5],
          ['Sindhi', 2.5], ['Tamil', 61], ['Telugu', 74], ['Urdu', 52]
        ]);

        var options = {
          title: 'Indian Language Use',
          legend: 'none',
          pieSliceText: 'label',
          slices: {  4: {offset: 0.2},
                    12: {offset: 0.3},
                    14: {offset: 0.4},
                    15: {offset: 0.5},
          },
        };

        var chart = new google.visualization.PieChart(document.getElementById('p
        chart.draw(data, options);
      }
    </script>
  </head>
  <body>
    <div id="piechart" style="width: 900px; height: 500px;"></div>
  </body>
</html>
```

Removing Slices

To omit a slice, change the color to `'transparent'`:

We also used the **`pieStartAngle`** to rotate the chart 135 degrees, **`pieSliceText`** to remove the text from the slices, and **`tooltip.trigger`** to disable tooltips:

```
<html>
  <head>
    <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js
    <script type="text/javascript">
      google.charts.load("current", {packages:["corechart"]});
      google.charts.setOnLoadCallback(drawChart);
      function drawChart() {
        var data = google.visualization.arrayToDataTable([
          ['Pac Man', 'Percentage'],
          ['', 75],
          ['', 25]
        ]);
```

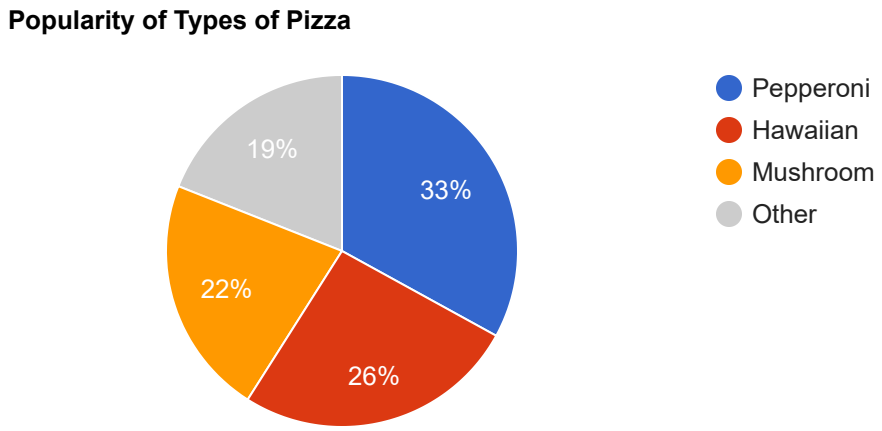
```
        var options = {
          legend: 'none',
          pieSliceText: 'none',
          pieStartAngle: 135,
          tooltip: { trigger: 'none' },
          slices: {
            0: { color: 'yellow' },
            1: { color: 'transparent' }
          }
        };

        var chart = new google.visualization.PieChart(document.getElementById('p
        chart.draw(data, options);
      }
    </script>
  </head>
  <body>
    <div id="pacman" style="width: 900px; height: 500px;"></div>
  </body>
</html>
```

Slice Visibility Threshold

You can set a value as the threshold for a pie slice to render individually. This value corresponds to a fraction of the chart (with the whole chart being of value 1). To set this threshold as a percentage of the whole, divide the percentage desired by 100 (for a 20% threshold, the value would be 0.2).

```
sliceVisibilityThreshold: 5/8 // This is equivalent to 0.625 or 62.5% of the cha
```

Any slices smaller than this threshold will be combined into a single "Other" slice, and will have the combined value of all slices below the threshold.




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

function drawChart() {

  var data = new google.visualization.DataTable();
  data.addColumn('string', 'Pizza');
  data.addColumn('number', 'Popularity');
  data.addRows([
    ['Pepperoni', 33],
    ['Hawaiian', 26],
    ['Mushroom', 22],
    ['Sausage', 10], // Below limit.
    ['Anchovies', 9] // Below limit.
  ]);

  var options = {
    title: 'Popularity of Types of Pizza',
    sliceVisibilityThreshold: .2
  };

  var chart = new google.visualization.PieChart(document.getElementById('cha
  chart.draw(data, options);
}
```

Loading

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

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

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

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

Data Format

Rows: Each row in the table represents a slice.

Columns:

	Column 0	Column 1	...Column *N* (optional)
Purpose:	Slice labels	Slice values	...Optional roles
Data Type:	string	number	...
Role:	domain	data	...
Optional column roles (/chart/interactive/docs/roles)**:**	*None*	*None*	... • tooltip (/chart/interactive/docs/roles#tooltiprole)

Configuration Options

Name	
backgroundColor	The background color for the main area of the chart. Can be either a simpl HTML color string, for example: `'red'` or `'#00cc00'`, or an object with th 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'
chartArea	An object with members to configure the placement and size of the chart a (where the chart itself is drawn, excluding axis and legends). Two formats supported: a number, or a number followed by %. A simple number is a valu 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 s (e.g., '#fdc') or an English color name. When an object is used, the followin 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
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 each element is an HTML color string, for example: `colors:['red','#00441`
	Type: Array of strings **Default:** default colors
enableInteractivity	Whether the chart throws user-based events or reacts to user interaction. I false, the chart will not throw 'select' or other interaction-based events (but throw ready or error events), and will not display hovertext or otherwise ch depending on user input. **Type:** boolean **Default:** true
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 ignc all IE8 charts are drawn in i-frames.) **Type:** boolean **Default:** false
height	Height of the chart, in pixels. **Type:** number **Default:** height of the containing element
is3D	If true, displays a three-dimensional chart.

	Type: boolean **Default:** false
legend	An object with members to configure various aspects of the legend. To spe properties of this object, you can use object literal notation, as shown here `{position: 'top', textStyle: {color: 'blue', fontSize:` **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 -- of th 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 right of the area, respectively. The default value depends on the legend's position. For 'bottom' legends, t default is 'center'; other legends default to 'start'. **Type:** string **Default:** automatic
legend.position	Position of the legend. Can be one of the following: • 'bottom' - Displays the legend below the chart. • 'labeled' - Draws lines connecting slices to their data values. • 'left' - Displays the legend left of the chart. • 'none' - Displays no legend. • 'right' - Displays the legend right of the chart. • 'top' - Displays the legend above the chart. **Type:** string **Default:** 'right'

legend.maxLines	Maximum number of lines in the legend. Set this to a number greater than to add lines to your legend. Note: The exact logic used to determine the ac number of lines rendered is still in flux. This option currently works only when legend.position is 'top'. **Type:** number **Default:** 1

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 **`'#00cc`**
	Also see **`fontName`** and **`fontSize`**.
	
	Type: object
	Default: **`{color: 'black', fontName: <global-font-name>, fontSize: <global-font-size>}`**

pieHole	If between 0 and 1, displays a donut chart. The hole with have a radius equ **`number`** times the radius of the chart. **Type:** number **Default:** 0
pieSliceBorderColor	The color of the slice borders. Only applicable when the chart is two-dimensional. **Type:** string **Default:** 'white'
pieSliceText	The content of the text displayed on the slice. Can be one of the following: • 'percentage' - The percentage of the slice size out of the total. • 'value' - The quantitative value of the slice. • 'label' - The name of the slice. • 'none' - No text is displayed.

	Type: string **Default:** 'percentage'
pieSliceTextStyle	An object that specifies the slice text style. The object has this format: `{color: <string>, fontName: <string>, fontSize: <number` The `color` can be any HTML color string, for example: `'red'` or `'#00cc0` Also see `fontName` and `fontSize`. **Type:** object **Default:** `{color: 'black', fontName: <global-font-name>, fontSize: <global-font-size>}`
pieStartAngle	The angle, in degrees, to rotate the chart by. The default of `0` will orient the leftmost edge of the first slice directly up. **Type:** number **Default:** `0`
reverseCategories	If true, draws slices counterclockwise. The default is to draw clockwise. **Type:** boolean **Default:** false
pieResidueSliceColor	Color for the combination slice that holds all slices below *sliceVisibilityThreshold*. **Type:** string **Default:** '#ccc'
pieResidueSliceLabel	A label for the combination slice that holds all slices below *sliceVisibilityThreshold*. **Type:** string **Default:** 'Other'
slices	An array of objects, each describing the format of the corresponding slice pie. To use default values for a slice, specify an empty object (i.e., `{}`). If a or a value is not specified, the global value will be used. Each object suppo the following properties: • `color` - The color to use for this slice. Specify a valid HTML color strin • `offset` - How far to separate the slice from the rest of the pie, from 0.(at all) to 1.0 (the pie's radius).

- **`textStyle`** - Overrides the global **`pieSliceTextStyle`** for this slice.

You can specify either an array of objects, each of which applies to the slic the order given, or you can specify an object where each child has a numer indicating which slice it applies to. For example, the following two declarat are identical, and declare the first slice as black and the fourth as red:

```
slices: [{color: 'black'}, {}, {}, {color: 'red'}]
slices: {0: {color: 'black'}, 3: {color: 'red'}}
```

Type: Array of objects, or object with nested objects
Default: {}

sliceVisibilityThreshold

The fractional value of the pie, below which a slice will not show individual slices that have not passed this threshold will be combined to a single "Oth slice, whose size is the sum of all their sizes. Default is not to show individ any slice which is smaller than half a degree.

```
// Slices less than 25% of the pie will be
// combined into an "Other" slice.
sliceVisibilityThreshold: .25
```

Type: number
Default: Half a degree (.5/360 or 1/720 or .0014)

title

Text to display above the chart.

Type: string
Default: no title

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 `'#00cc0` 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 the chart on all sides. **Note:** This only applies to HTML tooltips. If this is enabled with SVG tooltip any overflow outside of the chart bounds will be cropped. See Customizing Tooltip Content (/chart/interactive/docs/customizing_tooltip_content) for 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 dat role (/chart/interactive/docs/roles#tooltiprole) is **not** supported by the Bub Chart (/chart/interactive/docs/gallery/bubblechart) visualization. **Type:** boolean **Default:** false
tooltip.showColorCode	If true, show colored squares next to the slice information in the tooltip. **Type:** boolean **Default:** false

tooltip.text	What information to display when the user hovers over a pie slice. The follo values are supported: • 'both' - [*Default*] Display both the absolute value of the slice and the percentage of the whole. • 'value' - Display only the absolute value of the slice. • 'percentage' - Display only the percentage of the whole represented by t slice. **Type:** string **Default:** 'both'

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 **`'#00cc0`** 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 eler • 'none' - The tooltip will not be displayed. • 'selection' - The tooltip will be displayed when the user selects the elem **Type:** string **Default:** 'focus'
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')
```

Values 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(xPosition, optional_axis_index)`	Returns the horizontal data value at **`xPosition`**, which is a pixel 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 slices and legend entries. 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(yPosition, optional_axis_index)`	Returns the vertical data value at **`yPosition`**, which is a pixel offset down from the chart container's top edge. Can be negative. Example: `chart.getChartLayoutInterface().getVAxisValue(300)`. Call this *after* the chart is drawn. **Return Type:** number
`getXLocation(dataValue, optional_axis_index)`	Returns the pixel x-coordinate of **`dataValue`** relative to the left edge of the chart's container. Example: `chart.getChartLayoutInterface().getXLocation(400)`. Call this *after* the chart is drawn. **Return Type:** number
`getYLocation(dataValue, optional_axis_index)`	Returns the pixel y-coordinate of **`dataValue`** relative to the top edge of 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 slices and legend entries. 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

`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. A slice or legend entry correlates to a row in the data table (column index is null). **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. A slice or legend entry correlates to a row in the data table (column index is null). **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 2023-10-13 UTC.