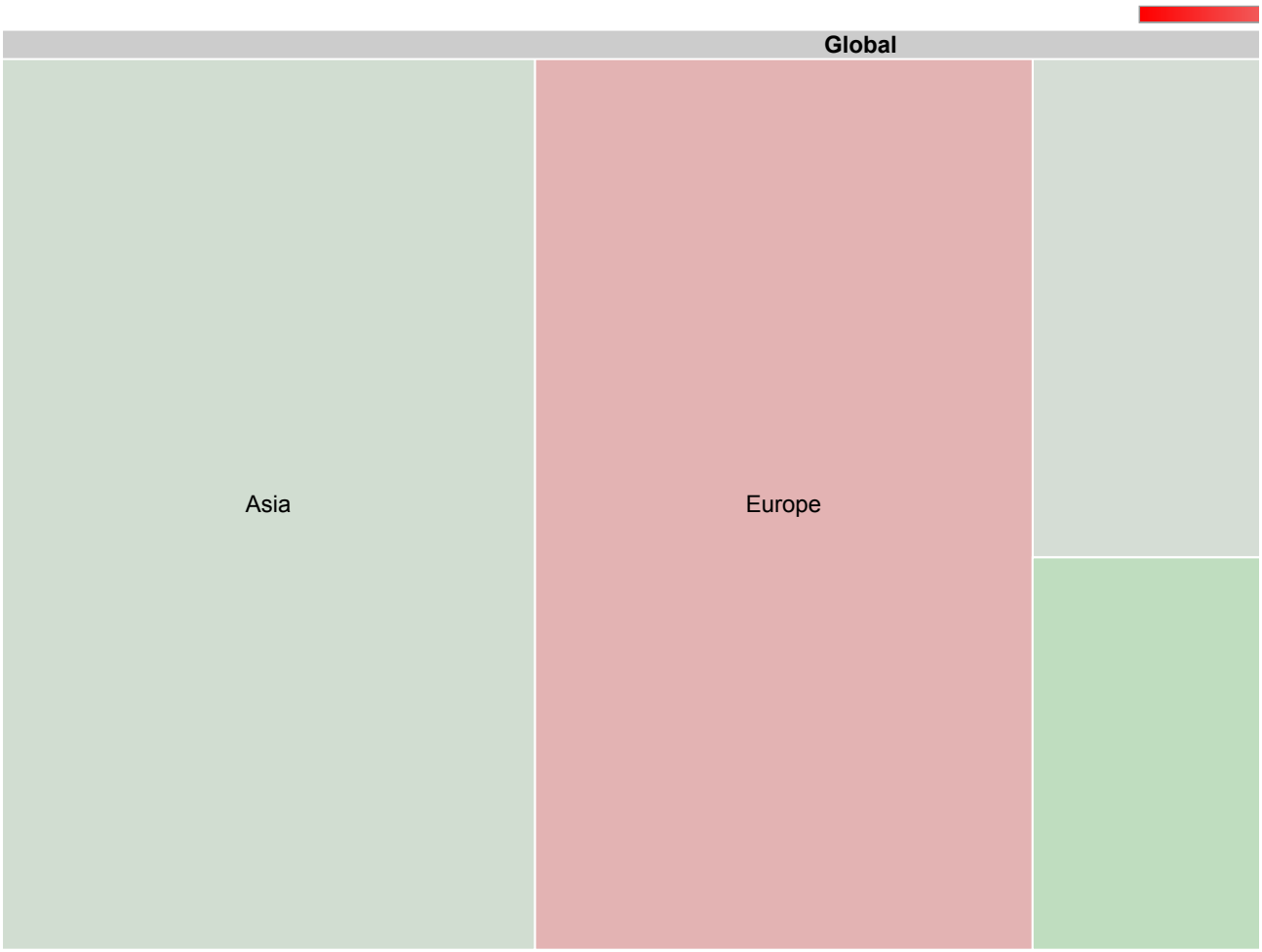
Treemaps

Overview

A visual representation of a data tree, where each node can have zero or more children, and one parent (except for the root, which has no parents). Each node is displayed as a rectangle, sized and colored according to values that you assign. Sizes and colors are valued relative to all other nodes in the graph. You can specify how many levels to display simultaneously, and optionally to display deeper levels in a hinted fashion. If a node is a leaf node, you can specify a size and color; if it is not a leaf, it will be displayed as a bounding box for leaf nodes. The default behavior is to move down the tree when a user left-clicks a node, and to move back up the tree when a user right-clicks the graph.

The total size of the graph is determined by the size of the containing element that you insert in your page. If you have leaf nodes with names too long to show, the name will be truncated with an ellipsis (...).

Example



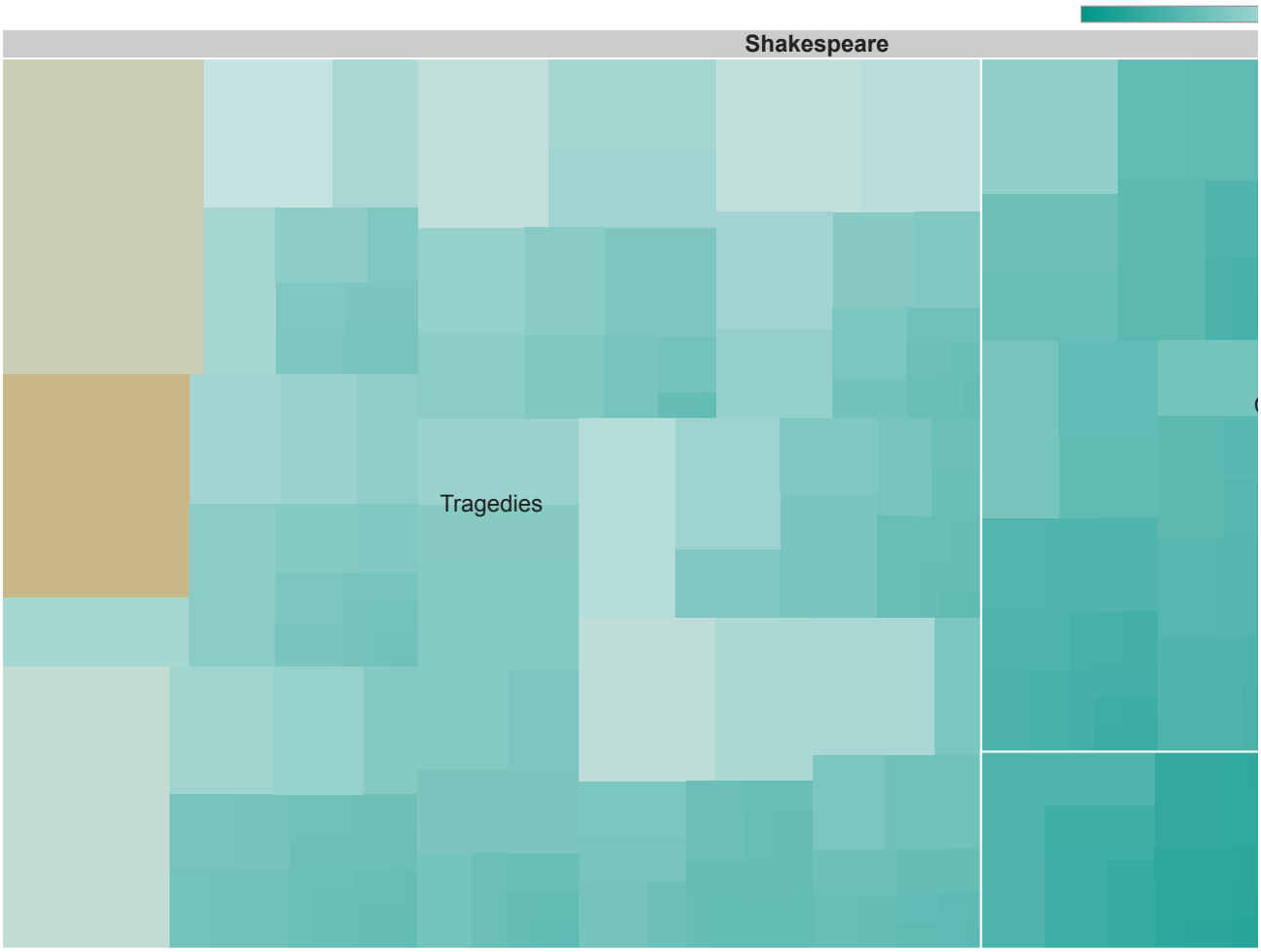
CODE IT YOURSELF ON JSFIDDLE

```
<html>
  <head>
    <script type="text/javascript" src="https://www.gstatic.com/charts/loader
    <script type="text/javascript">
      google.charts.load('current', {'packages':['treemap']});
      google.charts.setOnLoadCallback(drawChart);
      function drawChart() {
        var data = google.visualization.arrayToDataTable([
          ['Location', 'Parent', 'Market trade volume (size)', 'Market increas
          ['Global',    null,                 0,
          ['America',   'Global',             0,
          ['Europe',    'Global',             0,
          ['Asia',      'Global',             0,
          ['Australia', 'Global',             0,
          ['Africa',    'Global',             0,
          ['Brazil',    'America',            11,
          ['USA',       'America',            52,
          ['Mexico',    'America',            24,
          ['Canada',    'America',            16,
          ['France',    'Europe',             42,
```

```
          ['Germany',   'Europe',             31,
          ['Sweden',    'Europe',             22,
          ['Italy',     'Europe',             17,
          ['UK',        'Europe',             21,
          ['China',     'Asia',               36,
          ['Japan',     'Asia',               20,
          ['India',     'Asia',               40,
          ['Laos',      'Asia',               4,
          ['Mongolia',  'Asia',               1,
          ['Israel',    'Asia',               12,
          ['Iran',      'Asia',               18,
          ['Pakistan',  'Asia',               11,
          ['Egypt',     'Africa',             21,
          ['S. Africa', 'Africa',             30,
          ['Sudan',     'Africa',             12,
          ['Congo',     'Africa',             10,
          ['Zaire',     'Africa',             8,
        ]);

        tree = new google.visualization.TreeMap(document.getElementById('char

        tree.draw(data, {
          minColor: '#f00',
          midColor: '#ddd',
          maxColor: '#0d0',
          headerHeight: 15,
          fontColor: 'black',
          showScale: true
        });

      }
    </script>
  </head>
  <body>
    <div id="chart_div" style="width: 900px; height: 500px;"></div>
  </body>
</html>
```

Highlights

You can specify whether elements should highlight when moused over, and set specific colors for certain elements to use when this occurs. To turn on highlighting, set `highlightOnMouseOver: true`. From there, you can set your colors to use for highlighting elements using the various `HighlightColor` options.



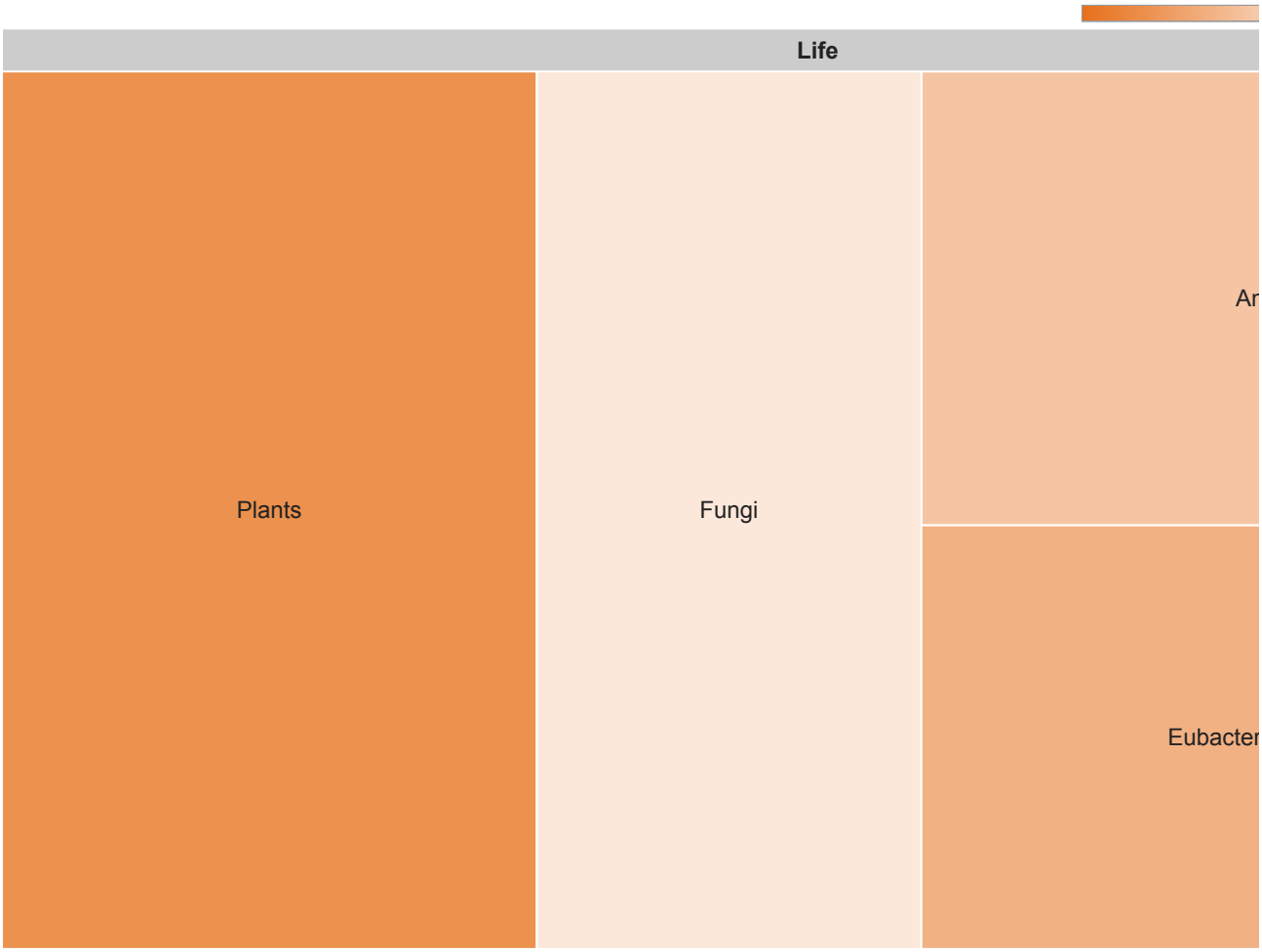
CODE IT YOURSELF ON JSFIDDLE

```
var options = {
  highlightOnMouseOver: true,
  maxDepth: 1,
  maxPostDepth: 2,
  minHighlightColor: '#8c6bb1',
  midHighlightColor: '#9ebcda',
  maxHighlightColor: '#edf8fb',
  minColor: '#009688',
  midColor: '#f7f7f7',
  maxColor: '#ee8100',
  headerHeight: 15,
  showScale: true,
  height: 500,
  useWeightedAverageForAggregation: true
};
```

Tooltips

By default, treemap tooltips are basic, showing the label of the treemap cell. This is useful when the cells are too small to contain the labels, but you can customize them further as described in this section.

Treemap tooltips are customized differently than other charts: you define a function and then set the `generateTooltip` option to that function. Here's a simple example:

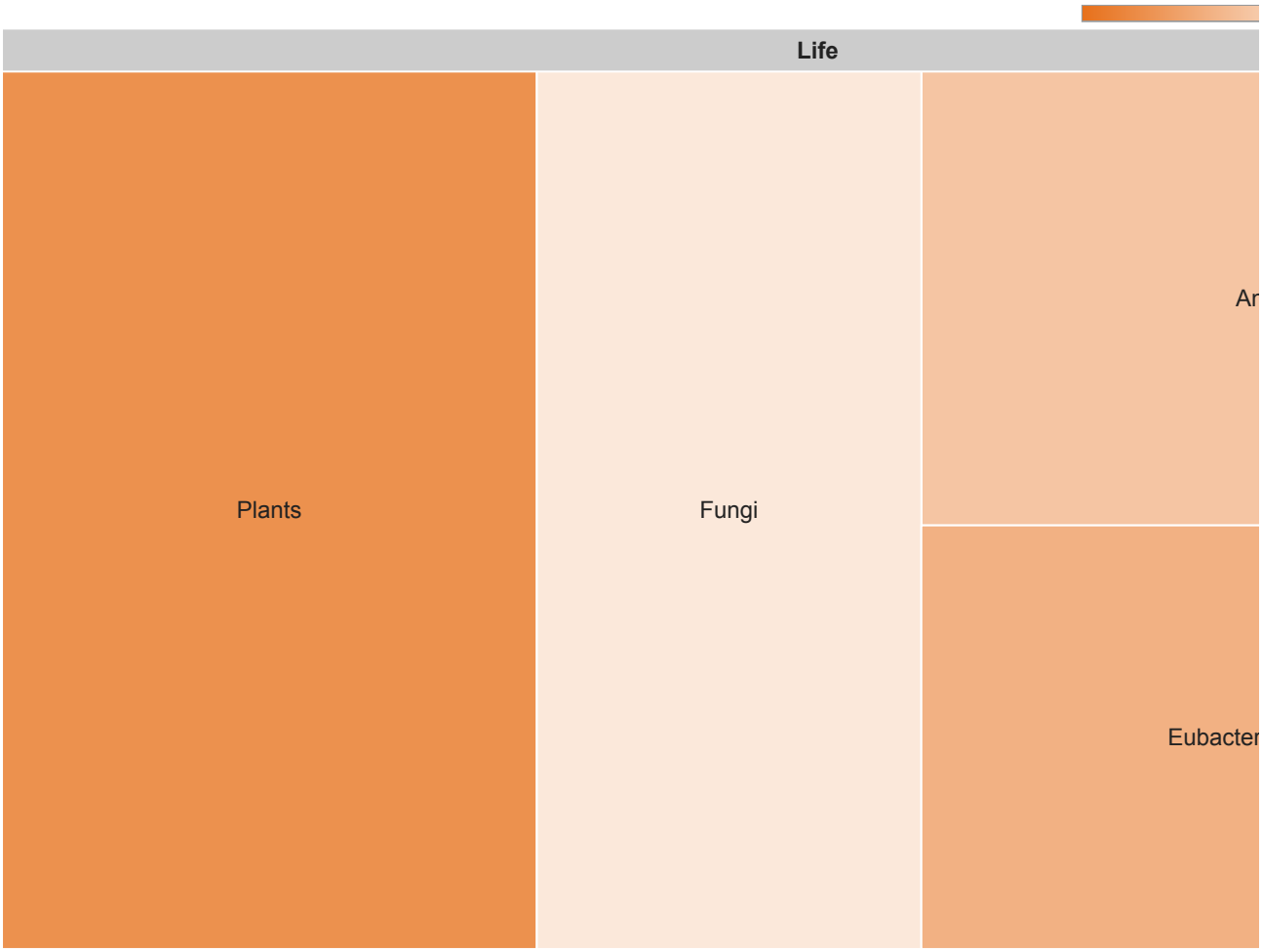


In the above chart, we define a function called `showStaticTooltip` that simply returns a string with the HTML to be shown whenever the user hovers over a treemap cell. Try it! The code to generate that is as follows:

```
var options = {
  minColor: '#e7711c',
  midColor: '#fff',
  maxColor: '#4374e0',
  showScale: true,
  generateTooltip: showStaticTooltip
};

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

```
function showStaticTooltip(row, size, value) {
  return '<div style="background:#fd9; padding:10px; border-style:solid">' +
         'Read more about the <a href="http://en.wikipedia.org/wiki/Kingdom
}
```

The `generateTooltip` function can be any JavaScript you like. Usually, you'll want tooltips that vary based on the data values. The following example shows how to access every field in the datatable.



If you hover over the cells in the above treemap, you'll see a different tooltip for each cell. The treemap tooltip functions all take three values: `row`, `size`, and `value`.

- `row`: the cell's row from the datatable
- `size`: the sum of the value (column 3) of this cell and all its children
- `value`: the color of the cell, expressed as a number from 0 to 1

In `showFullTooltip`, the string we return is an HTML box with five lines:

- Line 1 shows the appropriate row from the datatable, making liberal use of `data.getValue`.
- Line 2 tells you which row that is, which is just the `row` parameter.
- Line 3 gives you information from column 3 of the datatable: the sum of the value of column 3 from this row, plus the values from descendants.
- Line 4 gives you information from column 4 of the datatable. It's the value, but expressed as a number between 0 and 1 corresponding to the color of the cell.

```
  var options = {
    minColor: '#e7711c',
    midColor: '#fff',
    maxColor: '#4374e0',
    showScale: true,
    generateTooltip: showFullTooltip
  };

  tree.draw(data, options);

  function showFullTooltip(row, size, value) {
    return '<div style="background:#fd9; padding:10px; border-style:solid">' +
           '<span style="font-family:Courier"><b>' + data.getValue(row, 0) +
           '</b>, ' + data.getValue(row, 1) + ', ' + data.getValue(row, 2) +
           ', ' + data.getValue(row, 3) + '</span><br>' +
           'Datatable row: ' + row + '<br>' +
           data.getColumnLabel(2) +
           ' (total value of this cell and its children): ' + size + '<br>' +
           data.getColumnLabel(3) + ': ' + value + ' </div>';
  }

}
```

Loading

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

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

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

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

Data Format

Each row in the data table describes one node (a rectangle in the graph). Each node (except the root node) has one or more parent nodes. Each node is sized and colored according to its values relative to the other nodes currently shown.

The data table should have four columns in the following format:

- Column 0 - [*string*] An ID for this node. It can be any valid JavaScript string, including spaces, and any length that a string can hold. This value is displayed as the node header.
- Column 1 - [*string*] - The ID of the parent node. If this is a root node, leave this blank. Only one root is allowed per treemap.
- Column 2 - [*number*] - The size of the node. Any positive value is allowed. This value determines the size of the node, computed relative to all other nodes currently shown. For non-leaf nodes, this value is ignored and calculated from the size of all its children.
- Column 3 - [*optional*, *number*] - An optional value used to calculate a color for this node. Any value, positive or negative, is allowed. The color value is first recomputed on a scale from `minColorValue` to `maxColorValue`, and then the node is assigned a color from the gradient between `minColor` and `maxColor`.

Configuration Options

Name	
fontColor	The text color. Specify an HTML color value. **Type:** string **Default:** #ffffff
fontFamily	The font family to use for all text. **Type:** string **Default:** auto
fontSize	The font size for all text, in points. **Type:** number **Default:** 12
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
headerColor	The color of the header section for each node. Specify an HTML color value. **Type:** string **Default:** #988f86
headerHeight	The height of the header section for each node, in pixels (can be zero). **Type** number **Default:** 0
headerHighlightColor	The color of the header of a node being hovered over. Specify an HTML color value or null; if null this value will be **`headerColor`** lightened by 35%. **Type:** string **Default:** null
highlightOnMouseOver	Determines if elements should show highlighted effects when moused over. If set to **`true`**, highlighting for different elements can be specified using the various **`highlightColor`** options. **Type:** boolean **Default:** false
hintOpacity	When **`maxPostDepth`** is greater than 1, causing nodes below the current depth to be shown, **`hintOpacity`** specifies how transparent it should be. It should be between 0 and 1; the higher the value, the fainter the node. **Type:** number **Default:** 0.0
maxColor	The color for a rectangle with a column 3 value of **`maxColorValue`**. Specify an HTML color value. **Type:** string **Default:** #00dd00
maxDepth	The maximum number of node levels to show in the current view. Levels will be flattened into the current plane. If your tree has more levels than this, you will have to go up or down to see them. You can additionally see **`maxPostDepth`** levels below this as shaded rectangles within these nodes.

	Type: number **Default:** 1
maxHighlightColor	The highlight color to use for the node with the largest value in column 3. Specify an HTML color value or null; If null, this value will be the value of **`maxColor`** lightened by 35% **Type:** string **Default:** null
maxPostDepth	How many levels of nodes beyond **`maxDepth`** to show in "hinted" fashion. Hinted nodes are shown as shaded rectangles within a node that is within the **`maxDepth`** limit. **Type:** number **Default:** 0
maxColorValue	The maximum value allowed in column 3. All values greater than this will be trimmed to this value. If set to null, it will be set to the max value in the column. **Type:** number **Default:** null
midColor	The color for a rectangle with a column 3 value midway between **`maxColorValue`** and **`minColorValue`**. Specify an HTML color value. **Type:** string **Default:** #000000
midHighlightColor	The highlight color to use for the node with a column 3 value near the median of **`minColorValue`** and **`maxColorValue`**. Specify an HTML color value or null; if null, this value will be the value of **`midColor`** lightened by 35%. **Type:** string **Default:** null
minColor	The color for a rectangle with the column 3 value of **`minColorValue`**. Specify an HTML color value. **Type:** string **Default:** #dd0000
minHighlightColor	The highlight color to use for the node with a column 3 value nearest to **`minColorValue`**. Specify an HTML color value or null; if null, this value will be the value of **`minColor`** lightened by 35% **Type:** string **Default:** null

minColorValue	The minimum value allowed in column 3. All values less than this will be trimmed to this value. If set to null, it will be calculated as the minimum value in the column. **Type:** number **Default:** null
noColor	The color to use for a rectangle when a node has no value for column 3, and that node is a leaf (or contains only leaves). Specify an HTML color value. **Type:** string **Default:** #000000
noHighlightColor	The color to use for a rectangle of "no" color when highlighted. Specify an HTML color value or null; if null, this will be the value of **`noColor`** lightened by 35%. **Type:** string **Default:** null
showScale	Whether or not to show a color gradient scale from **`minColor`** to **`maxColor`** along the top of the chart. Specify true to show the scale. **Type:** boolean **Default:** false
showTooltips	Whether to show tooltips. **Type:** boolean **Default:** true

textStyle	An object that specifies the text style, for certain charts that have text in the content area such as the treemap. 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>}`**

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 **`'#00cc00'`**. Also see **`fontName`** and **`fontSize`**.
	
	Type: object
	Default: **`{color: 'black', fontName: <global-font-name>, fontSize: <global-font-size>}`**

useWeightedAverageForAggregation	Whether to use weighted averages for aggregation. **Type:** boolean **Default:** false

Methods

Method	
`draw(data, options)`	Draws the chart. **Return Type:** none
`getSelection()`	Standard **`getSelection()`** (https://developers.google.com/chart/interactive/docs/reference#visgetselection) implementation. Selected elements are nodes. Only one node can be selected at a time. **Return Type:** Array of selection elements
`setSelection()`	Standard **`setSelection()`** (https://developers.google.com/chart/interactive/docs/reference#vissetselection)

	implementation. Selected elements are nodes. Only one node can be selected at a time. **Return Type:** none
`goUpAndDraw()`	Move up the tree by one level and redraw it. Does not throw an error if the node is the root node. This is fired automatically when the user right-clicks a node. **Return Type:** none
`getMaxPossibleDepth()`	Returns the maximum possible depth for the current view. **Return Type:** number
`clearChart()`	Clears the chart, and releases all of its allocated resources. **Return Type:** none

Events

Name	
`onmouseover`	Fired when the user mouses over a node. The event handler is passed the row index of the corresponding entry in the data table. **Properties:** row
`onmouseout`	Fired when the user mouses out of a node. The event handler is passed the row index of the corresponding entry in the data table. **Properties:** row
`ready`	Fired when 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
`rollup`	Fired when the user navigates back up the tree, typically by right-clicking. The row property passed into the event handler is the row of the node that the user is navigating *from*, not the row the user is navigating *to*. **Properties:** row
`select`	Fired when the user clicks a node. To learn which node was selected,

	call `getSelection()` (#Methods). **Properties:** none

Data Policy

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

上次更新日期：二月 23, 2017