Chart Drawing Techniques

This page lists the different ways that you can instantiate and draw a chart on the page. Each method has advantages and disadvantages, as listed below.

Contents

chart.draw()

This is the basic method covered in the Hello Chart! example (https://developers.google.com/chart/interactive/docs/quick_start) in this documentation. Here are the basic steps:

1. Load the gstatic library loader, Google Visualization, and chart libraries
2. Prepare your data
3. Prepare any chart options
4. Instantiate the chart class, passing in a handle to the page container element.
5. Optionally register to receive any chart events. If you intend to call methods on the chart, you should listen for the "ready" event.
6. Call `chart.draw()`, passing in the data and options.

Advantages:

- You have complete control over every step of the process.
- You can register to listen for all events thrown by the chart.

Disadvantages:

- Verbose
- You must explicitly load all required chart libraries.
- If you send queries, you must manually implement for the callback, check for success, extract the returned `DataTable`, and pass it to the chart.

Example:

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

```
  <!--Load the AJAX API-->
  <script type="text/javascript" src="https://www.gstatic.com/charts/loader
  <script type="text/javascript">
   var data;
   var chart;

    // Load the Visualization API and the piechart package.
    google.charts.load('current', {'packages':['corechart']});

    // Set a callback to run when the Google Visualization API is loaded.
    google.charts.setOnLoadCallback(drawChart);

    // Callback that creates and populates a data table,
    // instantiates the pie chart, passes in the data and
    // draws it.
    function drawChart() {

      // Create our data table.
      data = new google.visualization.DataTable();
      data.addColumn('string', 'Topping');
      data.addColumn('number', 'Slices');
      data.addRows([
        ['Mushrooms', 3],
        ['Onions', 1],
        ['Olives', 1],
        ['Zucchini', 1],
        ['Pepperoni', 2]
      ]);

      // Set chart options
      var options = {'title':'How Much Pizza I Ate Last Night',
                     'width':400,
                     'height':300};

      // Instantiate and draw our chart, passing in some options.
      chart = new google.visualization.PieChart(document.getElementById('cha
      google.visualization.events.addListener(chart, 'select', selectHandle
      chart.draw(data, options);
    }

    function selectHandler() {
      var selectedItem = chart.getSelection()[0];
      var value = data.getValue(selectedItem.row, 0);
      alert('The user selected ' + value);
    }

  </script>
</head>
```

```
  <body>
    <!--Div that will hold the pie chart-->
    <div id="chart_div" style="width:400; height:300"></div>
  </body>
</html>
```

ChartWrapper

`ChartWrapper` (https://developers.google.com/chart/interactive/docs/reference#chartwrapperobject) is a convenience class that handles loading all the appropriate chart libraries for you and also simplifies sending queries to Chart Tools Datasources.

Advantages:

- Much less code
- Loads all the required chart libraries for you
- Makes querying Datasources much easier by creating the `Query` object and handling the callback for you
- Pass in the container element ID, and it will call getElementByID for you.
- Data can be submitted in a variety of formats: as an array of values, as a JSON literal string, or as a `DataTable` handle

Disadvantages:

- `ChartWrapper` currently propagates only the select, ready, and error events. To get other events, you must get a handle to the wrapped chart and subscribe to events there. See the `ChartWrapper` documentation (https://developers.google.com/chart/interactive/docs/reference#chartwrapperobject) for examples.

Examples:

Here's an example of a column chart with locally constructed data specified as an array. You cannot specify chart labels or datetime values using the array syntax, but you could manually create a `DataTable` object with those values and pass that to the `dataTable` property.

```
<html>
  <head>
    <script type="text/javascript" src="https://www.gstatic.com/charts/loader
    <script type="text/javascript">
```

```
      google.charts.load('current');   // Don't need to specify chart librari
      google.charts.setOnLoadCallback(drawVisualization);

      function drawVisualization() {
        var wrapper = new google.visualization.ChartWrapper({
          chartType: 'ColumnChart',
          dataTable: [['', 'Germany', 'USA', 'Brazil', 'Canada', 'France', 'RU
                      ['', 700, 300, 400, 500, 600, 800]],
          options: {'title': 'Countries'},
          containerId: 'vis_div'
        });
        wrapper.draw();
      }
    </script>
  </head>
  <body style="font-family: Arial;border: 0 none;">
    <div id="vis_div" style="width: 600px; height: 400px;"></div>
  </body>
</html>
```

Here's an example of a line chart that gets its data by querying a Google Spreadsheet. Note that the code doesn't need to handle the callback.

```
<html>
  <head>
    <script type="text/javascript" src="https://www.gstatic.com/charts/loader
    <script type="text/javascript">
      google.charts.load('current');
      google.charts.setOnLoadCallback(drawVisualization);

      function drawVisualization() {
        var wrapper = new google.visualization.ChartWrapper({
          chartType: 'LineChart',
          dataSourceUrl: 'http://spreadsheets.google.com/tq?key=pCQbetd-CptGX
          query: 'SELECT A,D WHERE D > 100 ORDER BY D',
          options: {'title': 'Countries'},
          containerId: 'vis_div'
        });
        wrapper.draw()

        // No query callback handler needed!
      }
    </script>
  </head>
  <body style="font-family: Arial;border: 0 none;">
    <div id="vis_div" style="width: 600px; height: 400px;"></div>
```

```
  </body>
</html>
```

Combined with autoloading (https://developers.google.com/chart/interactive/docs/library_loading_enhancements#enhancedloading), this can make for very compact code:

```
<html>
  <head>
    <script type="text/javascript" src="https://www.gstatic.com/charts/loader
    <script type="text/javascript">
      google.charts.load('current');   // Don't need to specify chart librari
      google.charts.setOnLoadCallback(drawVisualization);

      function drawVisualization() {
        var wrapper = new google.visualization.ChartWrapper({
          chartType: 'LineChart',
          dataSourceUrl: 'http://spreadsheets.google.com/tq?key=pCQbetd-CptGX
          query: 'SELECT A,D WHERE D > 100 ORDER BY D',
          options: {'title': 'Countries'},
          containerId: 'vis_div'
        });
        wrapper.draw()
      }
    </script>
  </head>
  <body style="font-family: Arial;border: 0 none;">
    <div id="vis_div" style="width: 600px; height: 400px;"></div>
  </body>
</html>
```

Using the Chart Editor with ChartWrapper

You can use the Chart Editor dialog built into Google Spreadsheets to design a chart and then request the serialized **`ChartWrapper`** string that represents the chart. You can then copy and paste this string and use it as described above in **`ChartWrapper`** (#chartwrapper).

You can embed a chart editor on your own page and expose methods for users to connect to other data sources and return the **`ChartWrapper`** string. See the **`ChartEditor`** reference documentation (https://developers.google.com/chart/interactive/docs/reference#google_visualization_charteditor) for more information.

DrawChart()

`DrawChart` is a global static method that wraps a `ChartWrapper`.

Advantages:

- Same as `ChartWrapper`, but slightly shorter to use.

Disadvantages:

- Does not return a handle to the wrapper, so you cannot handle any events.

```
<DOCTYPE html>
<html>
  <head>
    <script type="text/javascript" src="https://www.gstatic.com/charts/loader
    <script type="text/javascript">
      google.charts.load('current');   // Don't need to specify chart librari
      google.charts.setOnLoadCallback(drawVisualization);
      function drawVisualization() {
        google.visualization.drawChart({
         "containerId": "visualization_div",
         "dataSourceUrl": "https://spreadsheets.google.com/a/google.com/tq?ke
         "query":"SELECT A,D WHERE D > 100 ORDER BY D",
         "refreshInterval": 5,
         "chartType": "Table",
         "options": {
            "alternatingRowStyle": true,
            "showRowNumber" : true
         }
       });
      }
    google.charts.setOnLoadCallback(drawVisualization);
    </script>
  </head>
  <body style="font-family: Arial;border: 0 none;">
    <div id="visualization_div" style="width: 600px; height: 400px;"></div>
  </body>
</html>
```

More Information

- `ChartEditor` reference documentation (https://developers.google.com/chart/interactive/docs/reference#google_visualization_chareditor)
- Generic `chart.draw()` (https://developers.google.com/chart/interactive/docs/reference#visdraw) documentation
- `ChartWrapper` reference documentation (https://developers.google.com/chart/interactive/docs/reference#chartwrapperobject)
- `DrawChart` reference documentation (https://developers.google.com/chart/interactive/docs/reference#google.visualization.drawchart)

上次更新日期：一月 28, 2016