Code Examples

Here are some code samples to demonstrate using the Google Visualization API.

Table Example

	Name	Salary	Full Time
1	John	$10,000	✓
2	Mary	$25,000	✓
3	Steve	$8,000	✗
4	Ellen	$20,000	✓
5	Mike	$12,000	✗

```
function drawTable() {
  var data = new google.visualization.DataTable();
  data.addColumn('string', 'Name');
  data.addColumn('number', 'Salary');
  data.addColumn('boolean', 'Full Time');
  data.addRows(5);
  data.setCell(0, 0, 'John');
  data.setCell(0, 1, 10000, '$10,000');
  data.setCell(0, 2, true);
  data.setCell(1, 0, 'Mary');
  data.setCell(1, 1, 25000, '$25,000');
  data.setCell(1, 2, true);
  data.setCell(2, 0, 'Steve');
  data.setCell(2, 1, 8000, '$8,000');
  data.setCell(2, 2, false);
  data.setCell(3, 0, 'Ellen');
  data.setCell(3, 1, 20000, '$20,000');
  data.setCell(3, 2, true);
  data.setCell(4, 0, 'Mike');
  data.setCell(4, 1, 12000, '$12,000');
  data.setCell(4, 2, false);

  var table = new google.visualization.Table(document.getElementById('table_d
  table.draw(data, {showRowNumber: true, width: '100%', height: '100%'});

  google.visualization.events.addListener(table, 'select', function() {
    var row = table.getSelection()[0].row;
    alert('You selected ' + data.getValue(row, 0));
  });
}
```

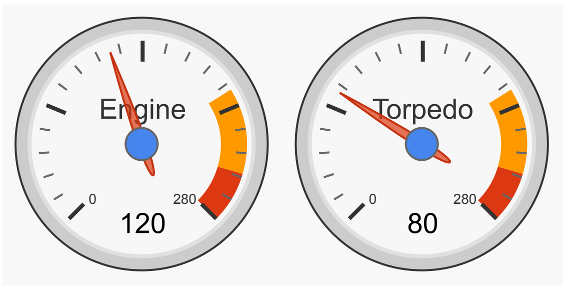
Customized Table Example

	Name	***Salary***	***Full Time***
1	John	**$10,000**	✓
2	**Mary**	**$25,000**	✓
3	Steve	$8,000	✗
4	*Ellen*	$20,000	✓
5	Mike	$12,000	✗

```
<style type='text/css'>
  .bold-green-font {
    font-weight: bold;
    color: green;
  }

  .bold-font {
    font-weight: bold;
  }

  .right-text {
    text-align: right;
  }

  .large-font {
    font-size: 15px;
  }

  .italic-darkblue-font {
    font-style: italic;
    color: darkblue;
  }

  .italic-purple-font {
    font-style: italic;
    color: purple;
  }

  .underline-blue-font {
    text-decoration: underline;
    color: blue;
  }

  .gold-border {
    border: 3px solid gold;
  }

  .deeppink-border {
    border: 3px solid deeppink;
  }

  .orange-background {
    background-color: orange;
  }

  .orchid-background {
    background-color: orchid;
  }

  .beige-background {
    background-color: beige;
  }

</style>

...

function drawTable() {
  var cssClassNames = {
    'headerRow': 'italic-darkblue-font large-font bold-font',
    'tableRow': '',
    'oddTableRow': 'beige-background',
    'selectedTableRow': 'orange-background large-font',
    'hoverTableRow': '',
    'headerCell': 'gold-border',
    'tableCell': '',
    'rowNumberCell': 'underline-blue-font'};

  var options = {'showRowNumber': true, 'allowHtml': true, 'cssClassNames': c

  var data = new google.visualization.DataTable();
  data.addColumn('string', 'Name');
  data.addColumn('number', 'Salary');
  data.addColumn('boolean', 'Full Time');
  data.addRows(5);
  data.setCell(0, 0, 'John');
  data.setCell(0, 1, 10000, '$10,000', {'className': 'bold-green-font large-f
  data.setCell(0, 2, true, {'style': 'background-color: red;'});
  data.setCell(1, 0, 'Mary', null, {'className': 'bold-font'});
  data.setCell(1, 1, 25000, '$25,000', {'className': 'bold-font right-text'})
  data.setCell(1, 2, true, {'className': 'bold-font'});
  data.setCell(2, 0, 'Steve', null, {'className': 'deeppink-border'});
  data.setCell(2, 1, 8000, '$8,000', {'className': 'deeppink-border right-tex
  data.setCell(2, 2, false, null);
  data.setCell(3, 0, 'Ellen', null, {'className': 'italic-purple-font large-f
  data.setCell(3, 1, 20000, '$20,000');
  data.setCell(3, 2, true);
  data.setCell(4, 0, 'Mike');
  data.setCell(4, 1, 12000, '$12,000');
  data.setCell(4, 2, false);
  var container = document.getElementById('table');
  var table = new google.visualization.Table(container);
  table.draw(data, options);
  table.setSelection([{'row': 4}]);
}
```

Gauge Example

Temperature:



GO FASTER

SLOW DOWN

CODE IT YOURSELF ON JSFIDDLE

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

    var gaugeOptions = {min: 0, max: 280, yellowFrom: 200, yellowTo: 250,
      redFrom: 250, redTo: 280, minorTicks: 5};
    var gauge;

    function drawGauge() {
      gaugeData = new google.visualization.DataTable();
      gaugeData.addColumn('number', 'Engine');
      gaugeData.addColumn('number', 'Torpedo');
      gaugeData.addRows(2);
      gaugeData.setCell(0, 0, 120);
      gaugeData.setCell(0, 1, 80);

      gauge = new google.visualization.Gauge(document.getElementById('gauge_d
      gauge.draw(gaugeData, gaugeOptions);
    }

    function changeTemp(dir) {
      gaugeData.setValue(0, 0, gaugeData.getValue(0, 0) + dir * 25);
      gaugeData.setValue(0, 1, gaugeData.getValue(0, 1) + dir * 20);
      gauge.draw(gaugeData, gaugeOptions);
    }
  </script>
 </head>
 <body>
  <div id="gauge_div" style="width:280px; height: 140px;"></div>
  <input type="button" value="Go Faster" onclick="changeTemp(1)" />
  <input type="button" value="Slow down" onclick="changeTemp(-1)" />
 </body>
</html>
```

Interaction example

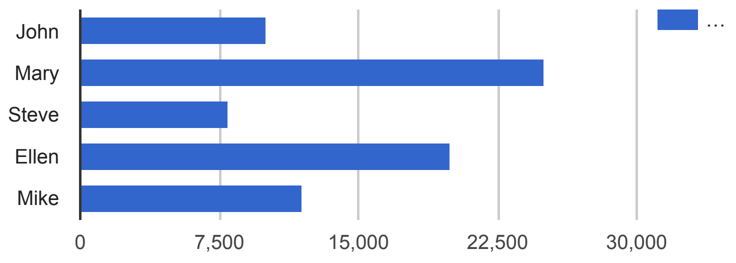
This example demonstrates how to combine visualizations for more complex interactivity. It demonstrates the following features:

- How to use a DataView (https://google-developers.appspot.com/chart/interactive/docs/reference#DataView) object to limit and format data in a DataTable,
- How to link two visualizations to the same data,
- How to use the Table visualization's 'sort' (https://google-developers.appspot.com/chart/interactive/docs/gallery/table#Events) event,
- How to use formatters (https://google-developers.appspot.com/chart/interactive/docs/reference#numberformatter) to

reformat displayed data.

Click on the table's headers to see the column chart getting sorted also.

ame	Salary
ohn	$10,000.00
ary	$25,000.00
teve	$8,000.00
llen	$20,000.00
ike	$12,000.00



```
function drawSort() {
  var data = new google.visualization.DataTable();
  data.addColumn('string', 'Name');
  data.addColumn('number', 'Salary');
  data.addColumn('boolean', 'Full Time');
  data.addRows(5);
  data.setCell(0, 0, 'John');
  data.setCell(0, 1, 10000);
  data.setCell(0, 2, true);
  data.setCell(1, 0, 'Mary');
  data.setCell(1, 1, 25000);
  data.setCell(1, 2, true);
  data.setCell(2, 0, 'Steve');
  data.setCell(2, 1, 8000);
  data.setCell(2, 2, false);
  data.setCell(3, 0, 'Ellen');
  data.setCell(3, 1, 20000);
  data.setCell(3, 2, true);
  data.setCell(4, 0, 'Mike');
  data.setCell(4, 1, 12000);
  data.setCell(4, 2, false);

  var formatter = new google.visualization.NumberFormat({prefix: '$'});
  formatter.format(data, 1); // Apply formatter to second column

  var view = new google.visualization.DataView(data);
  view.setColumns([0, 1]);

  var table = new google.visualization.Table(document.getElementById('table_s
  table.draw(view, {width: '100%', height: '100%'});
  var chart = new google.visualization.BarChart(document.getElementById('char
  chart.draw(view);

  google.visualization.events.addListener(table, 'sort',
      function(event) {
        data.sort([{column: event.column, desc: !event.ascending}]);
        chart.draw(view);
      });
}
```

Full HTML Page Example

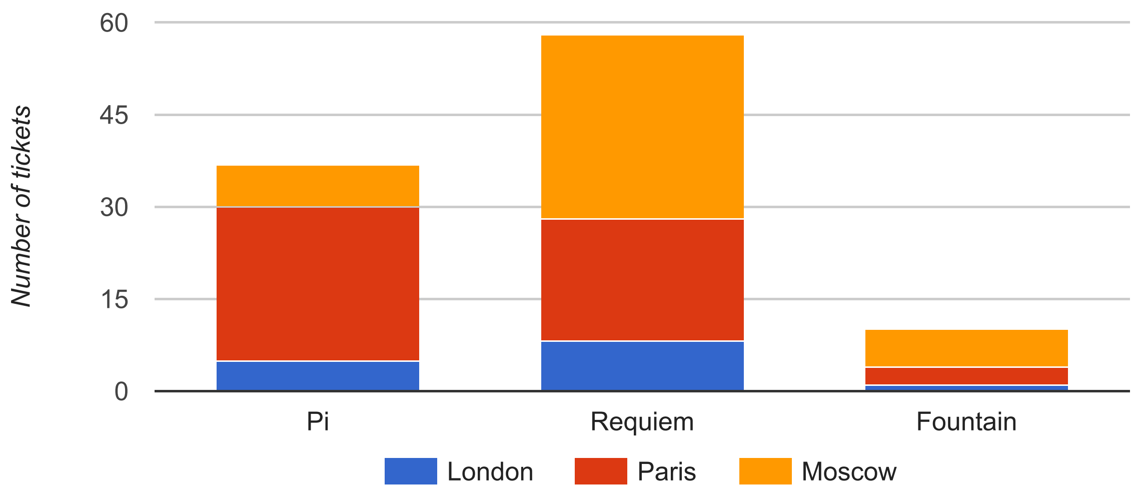
An end-to-end example for creating a web page with visualization charts embedded in it. It also demonstrates a chart connected to Google Spreadsheets (https://google-developers.appspot.com/chart/interactive/docs/spreadsheets) and two charts interacting using visualization Events (https://google-developers.appspot.com/chart/interactive/docs/events).

The example displays a simple statistics page for popular movies and cinema locations of a make-belief cinema chain company.

The page includes a Map (https://google-developers.appspot.com/chart/interactive/docs/gallery/map) and a Table (https://google-developers.appspot.com/chart/interactive/docs/gallery/table) visualization that interact with each other to display the theater locations. The page includes a column chart displaying the number of tickets each movie sold per location. It derives its data from a Google Spreadsheet. The published version of this spreadsheet (https://spreadsheets.google.com/pub?key=pCQbetd-CptF0r8qmCOlZGg) can be viewed for completeness.

Google
地图数据 ©2017 GS(2011)6020 Imagery ©2017 NAS
报告地图错误

Lat	Lon	Name
51.507	-0.128	Cinematics Londo
48.857	2.351	Cinematics Paris
55.75	37.617	Cinematics Mosco



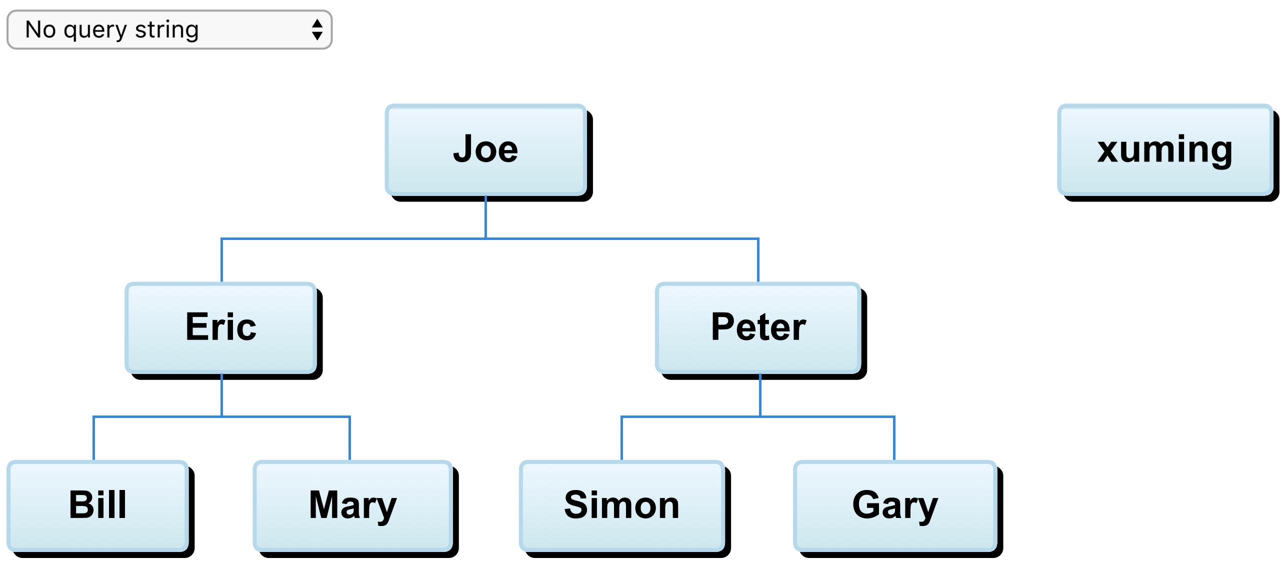
```
<html>
  <head>
    <script type="text/javascript" src="https://www.gstatic.com/charts/loader
    <script type="text/javascript">
      google.charts.load('current', {'packages': ['table', 'map', 'corechart'
      google.charts.setOnLoadCallback(initialize);

      function initialize() {
        // The URL of the spreadsheet to source data from.
        var query = new google.visualization.Query(
            'https://spreadsheets.google.com/pub?key=pCQbetd-CptF0r8qmCOlZGg'
        query.send(draw);
      }

      function draw(response) {
        if (response.isError()) {
          alert('Error in query');
        }

        var ticketsData = response.getDataTable();
        var chart = new google.visualization.ColumnChart(
            document.getElementById('chart_div'));
        chart.draw(ticketsData, {'isStacked': true, 'legend': 'bottom',
            'vAxis': {'title': 'Number of tickets'}});

        var geoData = google.visualization.arrayToDataTable([
          ['Lat', 'Lon', 'Name', 'Food?'],
          [51.5072, -0.1275, 'Cinematics London', true],
          [48.8567, 2.3508, 'Cinematics Paris', true],
          [55.7500, 37.6167, 'Cinematics Moscow', false]]);

        var geoView = new google.visualization.DataView(geoData);
        geoView.setColumns([0, 1]);

        var table =
            new google.visualization.Table(document.getElementById('table_div
        table.draw(geoData, {showRowNumber: false, width: '100%', height: '10

        var map =
            new google.visualization.Map(document.getElementById('map_div'));
        map.draw(geoView, {showTip: true});

        // Set a 'select' event listener for the table.
        // When the table is selected, we set the selection on the map.
        google.visualization.events.addListener(table, 'select',
            function() {
              map.setSelection(table.getSelection());
            });

        // Set a 'select' event listener for the map.
        // When the map is selected, we set the selection on the table.
        google.visualization.events.addListener(map, 'select',
            function() {
              table.setSelection(map.getSelection());
            });
      }
    </script>
  </head>

  <body>
```

```
    <table align="center">
      <tr valign="top">
        <td style="width: 50%;">
          <div id="map_div" style="width: 400px; height: 300;"></div>
        </td>
        <td style="width: 50%;">
          <div id="table_div"></div>
        </td>
      </tr>
      <tr>
        <td colSpan=2>
          <div id="chart_div" style="align: center; width: 700px; height: 300
        </td>
      </tr>
    </table>

  </body>
</html>
```

Query Wrapper Example

This example demonstrates how to create a JavaScript object that wraps many aspects of sending a query for you. This object, called QueryWrapper, is instantiated with a query string, a handle to a visualization, and a set of options for the visualization. It exposes one method to external callers, `sendAndDraw()`, which sends the query string, handles the response, and either calls `draw()` on the visualization, or displays an error message if the query returned an error. The host page here displays the results in an org chart visualization (https://google-developers.appspot.com/chart/interactive/docs/gallery/orgchart).

This example uses the following spreadsheet:
https://spreadsheets.google.com/pub?key=rCaVQNfFDMhOM6ENNYeYZ9Q



Here's the code for the host page:

```
<!DOCTYPE html>
<html>
<head>
  <title>Query Wrapper Example</title>
  <script type="text/javascript" src="https://www.gstatic.com/charts/loader.j
  <script type="text/javascript" src="querywrapper.js"></script>
  <script type="text/javascript">
    google.charts.load('current', {'packages' : ['orgchart']});
    google.charts.setOnLoadCallback(function() { sendAndDraw('') });

    var dataSourceUrl = 'https://spreadsheets.google.com/tq?key=rCaVQNfFDMhOM
    var query;

    function sendAndDraw(queryString) {
      var container = document.getElementById('orgchart');
      var orgChart = new google.visualization.OrgChart(container);
      query && query.abort();
      query = new google.visualization.Query(dataSourceUrl + queryString);
      var queryWrapper = new QueryWrapper(query, orgChart, {'size': 'large'},
      queryWrapper.sendAndDraw();
    }



  </script>
</head>
```

```
<body>
<h1>Query Wrapper Example</h1>
<form action="">
  <span> This example uses the following spreadsheet: <br />
    <a href="https://spreadsheets.google.com/pub?key=rCaVQNfFDMhOM6ENNYeYZ9Q":
      https://spreadsheets.google.com/pub?key=rCaVQNfFDMhOM6ENNYeYZ9Q
    </a></span>
  <br /><br />
  <select onChange="sendAndDraw(this.value)">
    <option value="">No query string</option>
    <option value="&tq=limit 3">query=limit 3</option>
    <option value="&tq=select G,H">(Error) query=select G,H</option>
  </select>
</form>
<br />
<div id="orgchart"></div>
</body>
</html>
```

Here's the JavaScript for the QueryWrapper object.

```
/**
 * A google.visualization.Query Wrapper. Sends a
 * query and draws the visualization with the returned data or outputs an
 * error message.
 *
 * DISCLAIMER: This is an example code which you can copy and change as
 * required. It is used with the google visualization API which is assumed to
 * be loaded to the page. For more info see:
 * https://developers.google.com/chart/interactive/docs/reference#Query
 */


/**
 * Constructs a new query wrapper with the given query, visualization,
 * visualization options, and error message container. The visualization
 * should support the draw(dataTable, options) method.
 * @constructor
 */
var QueryWrapper = function(query, visualization, visOptions, errorContainer)

  this.query = query;
  this.visualization = visualization;
  this.options = visOptions || {};
  this.errorContainer = errorContainer;
  this.currentDataTable = null;
  if (!visualization || !('draw' in visualization) ||
      (typeof(visualization['draw']) != 'function')) {
    throw Error('Visualization must have a draw method.');
  }
};


/** Draws the last returned data table, if no data table exists, does nothing
QueryWrapper.prototype.draw = function() {
  if (!this.currentDataTable) {
    return;
  }
  this.visualization.draw(this.currentDataTable, this.options);
};


/**
 * Sends the query and upon its return draws the visualization.
 * If the query is set to refresh then the visualization will be drawn upon
 * each refresh.
 */
QueryWrapper.prototype.sendAndDraw = function() {
  var query = this.query;
  var self = this;
  query.send(function(response) {self.handleResponse(response)});
};


/** Handles the query response returned by the data source. */
QueryWrapper.prototype.handleResponse = function(response) {
  this.currentDataTable = null;
  if (response.isError()) {
    this.handleErrorResponse(response);
  } else {
    this.currentDataTable = response.getDataTable();
    this.draw();
  }
};


/** Handles a query response error returned by the data source. */
QueryWrapper.prototype.handleErrorResponse = function(response) {
  var message = response.getMessage();
  var detailedMessage = response.getDetailedMessage();
  if (this.errorContainer) {
    google.visualization.errors.addError(this.errorContainer,
        message, detailedMessage, {'showInTooltip': false});
  } else {
    throw Error(message + ' ' + detailedMessage);
  }
};


/** Aborts the sending and drawing. */
QueryWrapper.prototype.abort = function() {
  this.query.abort();
};
```

Table Query Wrapper Example

This is an example demonstrating how to display a large data set in a paged table visualization (https://google-developers.appspot.com/chart/interactive/docs/gallery/table) without fetching all the data in a single request. This is useful when there is a large amount of data and you want to avoid the overhead of requesting or storing all the data at once on your page.

The JavaScript defines an object, TableQueryWrapper, which manages the `google.visualization.Query` (https://google-developers.appspot.com/chart/interactive/docs/reference#Query) object to make the requests, as well as the table visualization, to handle sorting and pagination. The object is instantiated with the `google.visualization.Query` instance, a handle to the page element used to hold the visualization and any errors it throws, and any options to pass into the `draw()` method (including the number of rows to fetch, as the `pageSize` option). This object manages the table visualization by catching and handling paging and sorting events.

When the user pages forward or backward, TableQueryWrapper handles the event by creating and sending a new query with appropriate LIMIT (https://google-developers.appspot.com/chart/interactive/docs/querylanguage#Limit) and OFFSET (https://google-developers.appspot.com/chart/interactive/docs/querylanguage#Offset) values to repopulate the table page. Similarly, when the user clicks a column to change the sort order, the TableQueryWrapper handles the event by creating and sending a new query with an appropriate ORDER BY (https://google-developers.appspot.com/chart/interactive/docs/querylanguage#Order_By) clause, and resets back to offset 0.

In the following example, the dropdown box enables you to select the number of table rows to show. If changed, the host page creates a new TableQueryWrapper instance, then sends a new query with a LIMIT clause reflecting the number of rows selected, and no OFFSET clause (that is, it jumps back to the first row).

This example uses the following spreadsheet:
http://spreadsheets.google.com/pub?key=rh_6pF1K_XsruwVr_doofvw

Number of rows to show: 5

	Name	Dept	Age	Coming
1	John	Men	32	✓
2	Joe	Women	78	✓
3	Bill	Women	25	✓
4	Mary	Kids	29	✗
5	Peter	Shoes	34	✓

◄ ► 1 2

Here's the code for the host page:

```
<!DOCTYPE html>
<html>
<head>
<title>Table Query Wrapper Example</title>
<script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"
<script type="text/javascript" src="tablequerywrapper.js"></script>
<script type="text/javascript">
    google.charts.load('current', {'packages' : ['table']});
    google.charts.setOnLoadCallback(init);

    var dataSourceUrl = 'https://spreadsheets.google.com/tq?key=rh_6pF1K_Xsru
    var query, options, container;

    function init() {
      query = new google.visualization.Query(dataSourceUrl);
      container = document.getElementById("table");
      options = {'pageSize': 5};
      sendAndDraw();
    }

    function sendAndDraw() {
      query.abort();
      var tableQueryWrapper = new TableQueryWrapper(query, container, options
      tableQueryWrapper.sendAndDraw();
    }
    function setOption(prop, value) {
      options[prop] = value;
      sendAndDraw();
    }

  </script>
</head>
<body>
<p>This example uses the following spreadsheet: <br />
  <a href="https://spreadsheets.google.com/pub?key=rh_6pF1K_XsruwVr_doofvw">
    https://spreadsheets.google.com/pub?key=rh_6pF1K_XsruwVr_doofvw
  </a>
</p>
<form action="">
  Number of rows to show:
  <select onChange="setOption('pageSize', parseInt(this.value, 10))">
    <option value="0">0</option>
    <option value="3">3</option>
    <option selected=selected value="5">5</option>
    <option value="8">8</option>
    <option value="-1">-1</option>
  </select>
</form>
<br />
<div id="table"></div>
</body>
</html>
```

Here is the JavaScript code for the TableQueryWrapper object:

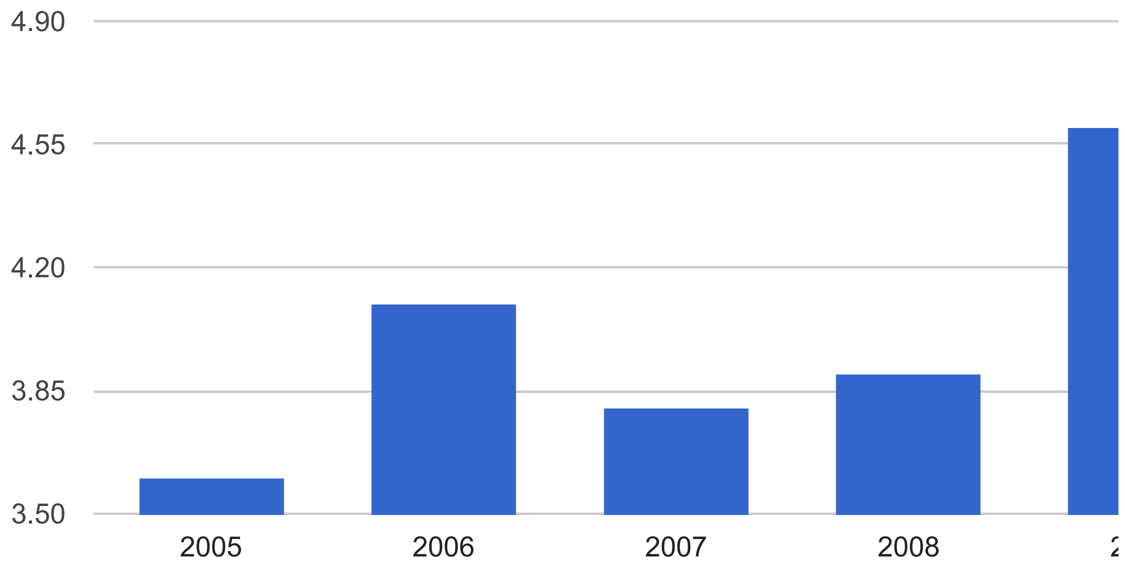
```
/**
 * A wrapper for a query and a table visualization.
 * The object only requests 1 page + 1 row at a time, by default, in order
 * to minimize the amount of data held locally.
 * Table sorting and pagination is executed by issuing
 * additional requests with appropriate query parameters.
 * E.g., for getting the data sorted by column 'A' the following query is
 * attached to the request: 'tq=order by A'.
 *
 * Note: Discards query strings set by the user on the query object using
 * google.visualization.Query#setQuery.
 *
 * DISCLAIMER: This is an example code which you can copy and change as
 * required. It is used with the google visualization API table visualization
 * which is assumed to be loaded to the page. For more info see:
```

```
 * https://developers.google.com/chart/interactive/docs/gallery/table
 * https://developers.google.com/chart/interactive/docs/reference#Query
 */


/**
 * Constructs a new table query wrapper for the specified query, container
 * and tableOptions.
 *
 * Note: The wrapper clones the options object to adjust some of its properti
 * In particular:
 *         sort {string} set to 'event'.
 *         page {string} set to 'event'.
 *         pageSize {Number} If number <= 0 set to 10.
 *         showRowNumber {boolean} set to true.
 *         firstRowNumber {number} set according to the current page.
 *         sortAscending {boolean} set according to the current sort.
 *         sortColumn {number} set according to the given sort.
 * @constructor
 */
var TableQueryWrapper = function(query, container, options) {

  this.table = new google.visualization.Table(container);
  this.query = query;
  this.sortQueryClause = '';
  this.pageQueryClause = '';
  this.container = container;
  this.currentDataTable = null;

  var self = this;
  var addListener = google.visualization.events.addListener;
  addListener(this.table, 'page', function(e) {self.handlePage(e)});
  addListener(this.table, 'sort', function(e) {self.handleSort(e)});

  options = options || {};
  options = TableQueryWrapper.clone(options);

  options['sort'] = 'event';
  options['page'] = 'event';
  options['showRowNumber'] = true;
  var buttonConfig = 'pagingButtonsConfiguration';
  options[buttonConfig] = options[buttonConfig] || 'both';
  options['pageSize'] = (options['pageSize'] > 0) ? options['pageSize'] : 10;
  this.pageSize = options['pageSize'];
  this.tableOptions = options;
  this.currentPageIndex = 0;
  this.setPageQueryClause(0);
};
```

```
/**
 * Sends the query and upon its return draws the Table visualization in the
 * container. If the query refresh interval is set then the visualization wil
 * be redrawn upon each refresh.
 */
TableQueryWrapper.prototype.sendAndDraw = function() {
  this.query.abort();
  var queryClause = this.sortQueryClause + ' ' + this.pageQueryClause;
  this.query.setQuery(queryClause);
  this.table.setSelection([]);
  var self = this;
  this.query.send(function(response) {self.handleResponse(response)});
};


/** Handles the query response after a send returned by the data source. */
TableQueryWrapper.prototype.handleResponse = function(response) {
  this.currentDataTable = null;
  if (response.isError()) {
    google.visualization.errors.addError(this.container, response.getMessage(
        response.getDetailedMessage(), {'showInTooltip': false});
  } else {
    this.currentDataTable = response.getDataTable();
    this.table.draw(this.currentDataTable, this.tableOptions);
  }
};


/** Handles a sort event with the given properties. Will page to page=0. */
TableQueryWrapper.prototype.handleSort = function(properties) {
  var columnIndex = properties['column'];
  var isAscending = properties['ascending'];
  this.tableOptions['sortColumn'] = columnIndex;
  this.tableOptions['sortAscending'] = isAscending;
  // dataTable exists since the user clicked the table.
  var colID = this.currentDataTable.getColumnId(columnIndex);
  this.sortQueryClause = 'order by `' + colID + (!isAscending ? '` desc' : '`
  // Calls sendAndDraw internally.
  this.handlePage({'page': 0});
};


/** Handles a page event with the given properties. */
TableQueryWrapper.prototype.handlePage = function(properties) {
  var localTableNewPage = properties['page']; // 1, -1 or 0
  var newPage = 0;
  if (localTableNewPage != 0) {
    newPage = this.currentPageIndex + localTableNewPage;
  }
  if (this.setPageQueryClause(newPage)) {
    this.sendAndDraw();
  }
};


/**
 * Sets the pageQueryClause and table options for a new page request.
 * In case the next page is requested - checks that another page exists
 * based on the previous request.
 * Returns true if a new page query clause was set, false otherwise.
 */
TableQueryWrapper.prototype.setPageQueryClause = function(pageIndex) {
  var pageSize = this.pageSize;

  if (pageIndex < 0) {
    return false;
  }
  var dataTable = this.currentDataTable;
  if ((pageIndex == this.currentPageIndex + 1) && dataTable) {
    if (dataTable.getNumberOfRows() <= pageSize) {
      return false;
    }
  }
  this.currentPageIndex = pageIndex;
  var newStartRow = this.currentPageIndex * pageSize;
  // Get the pageSize + 1 so that we can know when the last page is reached.
  this.pageQueryClause = 'limit ' + (pageSize + 1) + ' offset ' + newStartRow
  // Note: row numbers are 1-based yet dataTable rows are 0-based.
  this.tableOptions['firstRowNumber'] = newStartRow + 1;
  return true;
};


/** Performs a shallow clone of the given object. */
TableQueryWrapper.clone = function(obj) {
  var newObj = {};
  for (var key in obj) {
    newObj[key] = obj[key];
  }
  return newObj;
};
```

Mouseover Tooltip Example

This example demonstrates how to listen for mouseover events to display tooltips in your chart.



```
<script type="text/javascript">
  // barsVisualization must be global in our script tag to be able
  // to get and set selection.
  var barsVisualization;

  function drawMouseoverVisualization() {
    var data = new google.visualization.DataTable();
    data.addColumn('string', 'Year');
    data.addColumn('number', 'Score');
    data.addRows([
      ['2005',3.6],
      ['2006',4.1],
      ['2007',3.8],
      ['2008',3.9],
      ['2009',4.6]
    ]);

    barsVisualization = new google.visualization.ColumnChart(document.getElem
    barsVisualization.draw(data, null);
    // Add our over/out handlers.
    google.visualization.events.addListener(barsVisualization, 'onmouseover', 
    google.visualization.events.addListener(barsVisualization, 'onmouseout', 
  }

  function barMouseOver(e) {
    barsVisualization.setSelection([e]);
  }

  function barMouseOut(e) {
    barsVisualization.setSelection([{'row': null, 'column': null}]);
  }

</script>
```

上次更新日期：二月 23, 2017