Handling Events

This page describes how to listen for and handle events fired by a chart.

Contents

Overview

Google charts can fire *events* that you can listen for. Events can be triggered by user actions, such as when a user clicks on a chart. You can register a Javascript method to be called whenever certain events are fired, possibly with data specific to that event.

Every chart defines its own events, and the documentation for that chart should describe when each event is fired, what it means, and how to get back any information associated with the event. This page describes how to register to receive events from a chart, and how to handle them.

There is one event that any selectable chart should fire: the select event. However, the behavior and meaning of this event is defined by each chart.

It is important to note that the chart events are separate and distinct from the standard DOM events.

Registering For and Handling an Event

To register your event handlers, you call `google.visualization.events.addListener()` or `addOneTimeListener()` with the name of the chart exposing the event, the string name of the event to listen for, and the name of the function to call when that event is fired. Your function should accept a single parameter that is the event that was fired. This event can have custom information about the event, as described in the chart documentation.

Important: If your chart exposes a ready event (#The_Ready_Event), you should always wait for that event to be fired before you try to send methods or receive events from the chart. These charts might work before they throw a ready event, but that behavior is not guaranteed.

The following code snippet shows an alert box every time the user clicks on a table row:

```
// Create a table chart on your page.
var table = new google.visualization.Table(document.getElementById('table_div
table.draw(data, options);

// Every time the table fires the "select" event, it should call your
// selectHandler() function.
google.visualization.events.addListener(table, 'select', selectHandler);

function selectHandler(e) {
  alert('A table row was selected');
}
```

Note that this will only register to listen for events for this specific table object; you can only register to receive events from a specific object.

You can also pass in a function definition, as shown here:

```
// Pass in a function definition.
google.visualization.events.addListener(orgchart, 'select', function() {
  table.setSelection(orgchart.getSelection());
});
```

Retrieving Event Information

Events generally expose information in two ways: by passing information into the handler function as a parameter; or by adding information to a global object. If and how the event exposes information should be described in the documentation for that chart. Here is how to retrieve both types of information:

Event information passed to your handler

If the chart passes data as a parameter to your handling function, you would retrieve it as shown here:

```
// google.visualization.table exposes a 'page' event.
google.visualization.events.addListener(table, 'page', myPageEventHandler);
...
function myPageEventHandler(e) {
  alert('The user is navigating to page ' + e['page']);
}
```

The parameter passed in to your handler will have a property that should be documented for the chart. For an example of a chart that exposes event information this way, see the

Table (https://google-developers.appspot.com/chart/interactive/docs/gallery/table#Events) chart's page event.

Event information passed to a global object

Some events instead change a property of a global object, which you can then request. A common example of that is the "select" event, which is fired when a user selects a part of a chart. In this case, the code must call getSelection() on the chart to learn what the current selection is. More information is given on the select event below.

```
// orgChart is my global orgchart chart variable.
google.visualization.events.addListener(orgChart, 'select', selectHandler);
...
// Notice that e is not used or needed.
function selectHandler(e) {
  alert('The user selected' + orgChart.getSelection().length + ' items.');
```

The *select* Event

As mentioned previously, any chart that can be selected should fire a "select" event that works in a standard way to let you retrieve the values of the selected item in the chart. (However, there is no absolute *requirement* that a chart behave this way; check the documentation for your chart).

In general, charts that expose the "select" event are designed with the following specifications:

- The select event does not pass any properties or objects to the handler (your function handler should not expect any parameters to be passed to it).
- The chart *should* expose the method `getSelection()`, which returns an array of objects describing the selected data elements. These objects have the properties `row` and `column`. `row` and `column` are the row and column indexes of the selected item in the `DataTable`. (Selection events describe the underlying data in the graph, not HTML elements in the chart.) To get the data of the item selected, you'll have to call `DataTable.getValue()` (https://google-developers.appspot.com/chart/interactive/docs/reference#DataTable_getValue) or `getFormattedValue()` (https://google-developers.appspot.com/chart/interactive/docs/reference#DataTable_getFormattedValue) .

If both `row` and `column` are specified, the selected element is a cell. If only `row` is specified, the selected element is a row. If only `column` is specified, the selected element is a column.

- The chart *should* expose the method `setSelection(selection)` to change the selection in the underlying table and select the corresponding data in the chart. The *selection* parameter that is an array similar to the `getSelection()` array, where each element is an object with properties `row` and `column`. The `row` property defines the index of the selected row in the `DataTable`, and the `column` property defines the index of the selected column in the `DataTable`. When this method is called, the chart should visually indicate what the new selection is. The implementation of `setSelection()` *should not* trigger a 'select' event.
  If both `row` and `column` are specified, the selected element is a cell. If only `row` is specified, the selected element is a row. If only `column` is specified, the selected element is a column.

Some caveats:

- Chart might ignore part of the selection. For example a table that can show only selected rows may ignore cell or column elements in their `setSelection` implementation.)
- Some charts might not trigger a 'select' event, and some charts may support only entire row selection or entire column selection. The documentation of each chart defines the events and methods it supports.
- Multiselection is handled differently in different charts (some don't even allow it).
- In order to read the selected data, you'll need to call `DataTable.getValue()` in your handler; the simplest way to enable that is to make the `DataTable` object global.

The following example pops up an alertbox with the selected table elements, when an element of a table chart is selected:

	Name	Manager
1	Mary	
2	John	Mary
3	Steve	Mary
4	Ellen	Steve
5	Robert	Steve

Note that the table chart (https://google-developers.appspot.com/chart/interactive/docs/gallery/table) only fires row selection events; however, the code is generic, and can be used for row, column, and cell selection events.

Here's the handler code for that example:

```
// Create our table.
var table = new google.visualization.Table(document.getElementById('table_div
table.draw(data, options);

// Add our selection handler.
google.visualization.events.addListener(table, 'select', selectHandler);

// The selection handler.
// Loop through all items in the selection and concatenate
// a single message from all of them.
function selectHandler() {
  var selection = table.getSelection();
  var message = '';
  for (var i = 0; i < selection.length; i++) {
    var item = selection[i];
    if (item.row != null && item.column != null) {
      var str = data.getFormattedValue(item.row, item.column);
      message += '{row:' + item.row + ',column:' + item.column + '} = ' + str
    } else if (item.row != null) {
      var str = data.getFormattedValue(item.row, 0);
      message += '{row:' + item.row + ', column:none}; value (col 0) = ' + st
    } else if (item.column != null) {
      var str = data.getFormattedValue(0, item.column);
      message += '{row:none, column:' + item.column + '}; value (row 0) = ' +
    }
  }
  if (message == '') {
```

```
    message = 'nothing';
  }
  alert('You selected ' + message);
}
```

The *ready* Event

Most charts are rendered asynchronously; all Google charts throw a ready event after you call `draw()` on them, indicating that the chart is rendered, and ready to return properties or handle further method calls. You should always listen for the ready event before trying to call methods on it after calling `draw()`.

In general, charts that expose the "ready" event are designed with the following specifications:

- The ready event does not pass any properties to the handler (your function handler should not expect any parameters to be passed to it).
- The chart *should* fire the ready event after the chart is ready for interaction. If the drawing of the chart is asynchronous, it is important that the event is fired when interaction methods can actually be called, and not just when the `draw` method ends.
- Adding a listener to this event should be done before calling the `draw()` method, because otherwise the event might be fired before the listener is set up and you will not catch it.
- By calling interaction methods before the ready event is fired, you risk that these methods will not work properly.

The convention is that charts that do not fire a "ready" event are ready for interaction immediately after the `draw` method ends and returns control to the user. If your chart *does* fire a ready event, you should wait for it to be thrown before calling methods on it, as shown here:

```
google.visualization.events.addListener(tableChart, 'ready', myReadyHandler);
```

Ready Event Handler Syntax

```
function myReadyHandler(){...}
```

The ready event handler is not passed any parameters.

The *error* Event

Charts should throw an error event when they encounter some sort of error, to enable you to handle it gracefully. The event handler is passed a description of the error, as well as custom event properties specific to each chart. You should subscribe to this event right after instantiating the chart to trap any errors that might occur in later steps.

You can use the `goog.visualization.errors` (https://google-developers.appspot.com/chart/interactive/docs/reference#errordisplay) helper functions to help you display any errors gracefully to the user.

Error Event Handler Syntax

```
function myErrorHandler(err){...}
```

The error event handler should be passed an object with the following members:

- **id** [*Required*] - The ID of the DOM element containing the chart, or an error message displayed instead of the chart if it cannot be rendered.
- **message** [*Required*] - A short message string describing the error.
- **detailedMessage** [*Optional*] - A detailed explanation of the error.
- **options** [*Optional*]- An object containing custom parameters appropriate to this error and chart type.

Event Handling Example

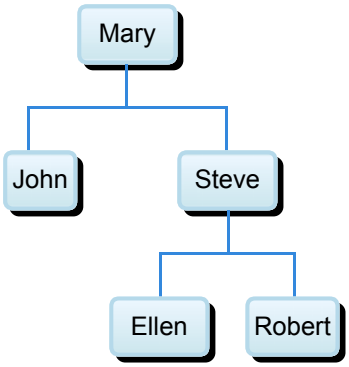
The following example demonstrates both getSelection() and setSelection(). It synchronizes the selection between two charts that use the same data table. Click on either chart to synchronize the selection in the other chart.

```
// Create our two charts.
var table = new google.visualization.Table(document.getElementById('table_div
table.draw(data, {});

var orgchart = new google.visualization.OrgChart(document.getElementById('org
orgchart.draw(data, {});

// When the table is selected, update the orgchart.
google.visualization.events.addListener(table, 'select', function() {
  orgchart.setSelection(table.getSelection());
```

```
});

// When the orgchart is selected, update the table chart.
google.visualization.events.addListener(orgchart, 'select', function() {
  table.setSelection(orgchart.getSelection());
});
```

Click on the charts below on table rows or on chart elements to see the selection in action:

	Name	Manager
1	Mary	
2	John	Mary
3	Steve	Mary
4	Ellen	Steve
5	Robert	Steve



上次更新日期：二月 23, 2017