Tableau JavaScript API

Documentation

This documentation contains description of the most important features of this API. It includes information about displaying Tableau Public visualizations, filtering data at visualization load and with use HTML controls, exporting visualization to a PDF file.

Przemysław Rosowski

Tableau JavaScript API

Documentation

Introduction

Tableau JavaScript API is used to integrate Tableau visualizations into web applications. The API lets you tightly control your users' interactions and combine functionality that otherwise couldn't be combined. For example, you can code a single control that filters a group of marks, selects some of those marks etc.

To program with the Tableau JavaScript API you need to have access to Tableau Server, Tableau Online, or Tableau Public, and a published workbook on that server. Moreover, end-users should have one of these web browsers: Chrome, Firefox, Safari 3.2.1 and later, and Internet Explorer 8.0 and later.

It is very important to use JavaScript file with the corresponding version of Tableau Server. Version 9.1, 9.2 and 9.3 use latest version of JS: *tableau-2.js.* The same JS files use Tableau Public and Tableau Online

Initialization

To be able to use this API, a web page have to include the JavaScript API file from the Tableau Server that hosts visualizations:

```
<script src="https://YOUR-SERVER/javascripts/api/tableau-2.js"></script>
```

For Tableau Online and Tableau Public, enter one of the following:

```
<script src="https://online.tableau.com/javascripts/api/tableau-2.js"></script>
<script src="https://public.tableau.com/javascripts/api/tableau-2.js"></script>
```

Create a div element in the page body where Tableau visualization should appear:

```
<div id="vizContainer"></div>
```

Element body has to invoke function that displays visualizations:

```
<body onload="initViz();"></div>
```

Code below present the simplest way to display the visualization. To instantiate a new Viz object, call the Viz constructor, passing a parent element, URL parameters (to visualization) and a set of options. The last parameter is optional.

```
function initViz() {
    var containerDiv = document.getElementById("vizContainer"),
    url = "http://YOUR-SERVER/views/YOUR-VISUALIZATION";
    options = {}
    var viz = new tableau.Viz(containerDiv, url, options);
};
```

The most important fields in *options* are:

hideTabs - indicates whether tabs are hidden or shown,
hideToolbar - indicates whether the toolbar is hidden or shown
height – to set height of view, if not specified defaults to the published height of the view
weight – to set weight of view, if not specified defaults to the published weight of the view
onFirstInteractive – this function is called once, when Viz object become interactive. This is an asynchronous callback, so code can continue to run while the Viz object is instantiating.

Below is code of function *initViz()* with options field:

```
function initViz() {
    var containerDiv = document.getElementById("vizContainer"),
    url = "http://YOUR-SERVER/views/YOUR-VISUALIZATION";
    options = {
      hideTabs: true,
      hideToolbar: true,
      width : “600px”,
      onFirstInteractive : function() (
            // Place to implement functions after viz loaded
            Alert(“Your viz was created succesfully”);
      )

    }
    var viz = new tableau.Viz(containerDiv, url, options);
};
```

Dynamic filtering and changing the appearance

Tableau JavaScript API provides setting the values for a filter prior to load. If values of filters are setting programmatically at load time, this could greatly improve your overall load times.

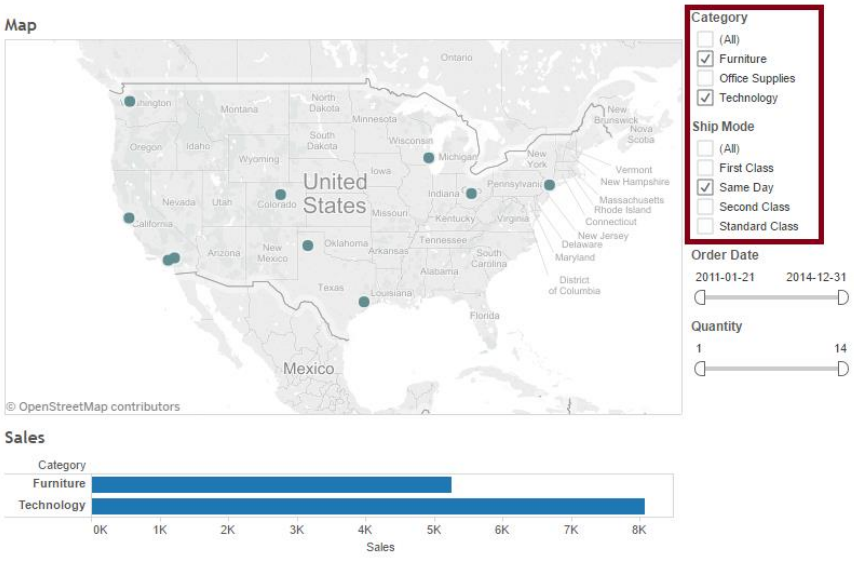
Filtering Discrete Values

The Tableau allows to user to create filters and implement them to worksheets. JavaScript API allows to redefine filter settings. Here have been presented possible methods to change filter operating on discrete values.

First method for apply filter values prior to load is to setting them directly in *option* field of the constructor. For example:

```
function initViz() {
    var containerDiv = document.getElementById("vizContainer"),
    url = "http://YOUR-SERVER/views/YOUR-VISUALIZATION";
    options = {
      "Filter Name" : "Value"
      //FOR EXAMPLE:
      "Category" : ["Furniture", "Technology"]
    }
    var viz = new tableau.Viz(containerDiv, url, options);
};
```

An example above present filtering visualization. From "*Category*" filter have been chosen two values: Furniture and Technology and from "*Ship mode*" one value – Same Day.



Another way to select values to filter is to use *onFirstInteractive* field in *options.* Firstly, there is need to get workbook to which this sheet belongs (*getWorkbook* property) and get currently active sheet (*getActiveSheet* property). In *if* clause has been checked if this sheet is worksheet or dashboard. If it is then filter is applied to worksheet. Otherwise, the next step is to get the collection of worksheets contained in the dashboard (*getWorksheets* property). After that is in loop, filter is implement to all elements in collection *worksheets*. *ApplyRangeFilterAsync* apply replace values for filter named (filterName) with values specified in values element of this function.

```
function initViz() {
    var containerDiv = document.getElementById("vizContainer"),
    url = "http://YOUR-SERVER/views/YOUR-VISUALIZATION";
    options = {
      onFirstInteractive: function() {
            sheet = viz.getWorkbook().getActiveSheet();
            if (sheet.getSheetType() = tableau.SheetType.WORKSHEET){
                  sheet. applyFilterAsync(filterName,
                  values,tableau.FilterUpdateType.REPLACE);
                  }
            else {
                  worksheets = sheet.getWorksheets();
                  for(var i = 0; i < worksheets.length; i++) {
                        worksheets[i]. applyFilterAsync(filterName,
                        values,tableau.FilterUpdateType.REPLACE);
                  }
            }
    }
    var viz = new tableau.Viz(containerDiv, url, options);
};
```

Summarizing both ways to filtering discrete values it should be said, that second way is less efficient, because filter is replace after visualization is loaded. First way is more efficient because filter is defining before visualizations loading.

Filtering Continuous Values

Tableau allows filtering of continuous data like range of dates, value of sales. One of the problem is how to set the initial value or final value on filter. Has been created method to get current time and period time (for example 3 last years from current date). This function is shown in *Supplement 1*.

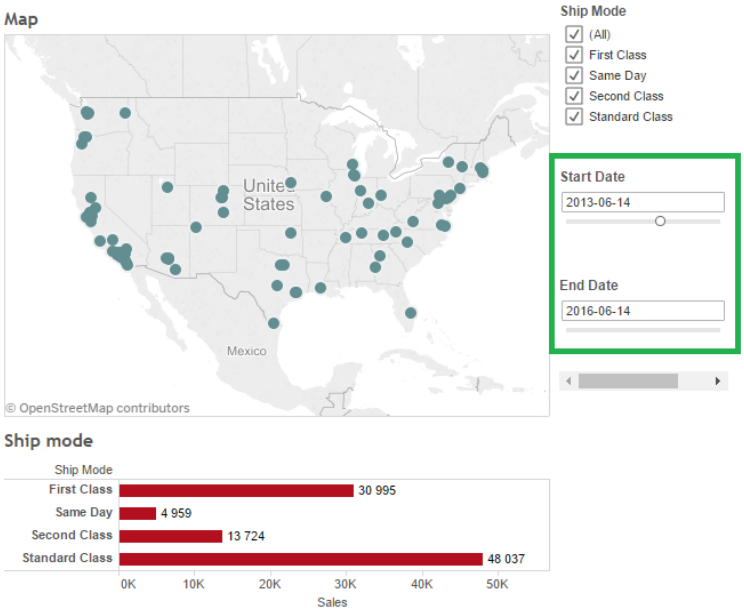
Similar to filtering discrete values, in this case is possibility to filter values with *onFirstInteractive* function and with write directly values of filter in *option* variable. It is needed to set start date or value and end of date or value in parameters. Parameter names need to be distinct from any Field Names, otherwise Tableau will apply value to filter the Field rather than changing the parameter.

Parameters are set up in *option* field in constructor. This method is very similar with method from "Filtering Discrete Values". It is very important to assign to variable dates with the same format as in filter.

```
function initViz() {
    var containerDiv = document.getElementById("vizContainer"),
    url = "http://YOUR-SERVER/views/YOUR-VISUALIZATION";
    options = {
      //Other options
      "Parameter1" = Value,
      "Parameter2" = Value
      //For example
      //"Start Date" = 2013-06-14,
      //"End Date" = 2016-06-14
    }
    var viz = new tableau.Viz(containerDiv, url, options);
};
```

This solution has one problem. When end date is set in Tableau Javascript API end when user change value of this parameter on page then previous date is specified in max attribute of this parameter in Tableau Desktop/Public. Moreover, the ability for parameter values to update automatically when the underlying data changes is not available in Tableau Desktop.

An example on next page present filtering visualization. Parameters *Start Date* and *End Date* have set dates like in code above in comment.



The second idea to filter continuous values use *AppleRangeFilterAsync* function. In this solution is not necessary to define parameters.

```
function initViz() {
    var containerDiv = document.getElementById("vizContainer"),
    url = "http://YOUR-SERVER/views/YOUR-VISUALIZATION";
    options = {
      onFirstInteractive: function() {
            sheet = viz.getWorkbook().getActiveSheet();
            if (sheet.getSheetType() = tableau.SheetType.WORKSHEET){
                  sheet. applyRangeFilterAsync (filterName,
                  { max: value, min : value });
                  }
            else {
                  worksheets = sheet.getWorksheets();
                  for(var i = 0; i < worksheets.length; i++) {
                        worksheets[i]. applyRangeFilterAsync (filterName,
                        { max: value, min : value });
                  }
            }
    }
    var viz = new tableau.Viz(containerDiv, url, options);
};
```

This solution required date range filter. Moreover, this solution is better than solution with parameters. It allows user to set any date or numbers. Filtering with *onFirstInteractive* is slower than filtering by parameters. Second way doesn't support automatically updates of data.

Selecting Marks

Selecting marks is almost identical to filtering. For filtering use one of the *applyFilterAsync()* methods. For selecting marks, you use *selectMarksAsync().* The parameters for mark selection are almost identical to those used for filtering.

```
function initViz() {
    var containerDiv = document.getElementById("vizContainer"),
    url = "http://YOUR-SERVER/views/YOUR-VISUALIZATION";
    options = {
      onFirstInteractive: function() {
            sheet = viz.getWorkbook().getActiveSheet();
            if (sheet.getSheetType() = tableau.SheetType.WORKSHEET){
                  sheet.selectMarksAsync({
                  "Field1": "value",
                  "Field2": ["value1", "value2", "value3"]},
                  tableau.SelectionUpdateType.REPLACE);
                  }
            else {
                  worksheets = sheet.getWorksheets();
                  for(var i = 0; i < worksheets.length; i++) {
                        worksheets[i]. selectMarksAsync({
                        "Field1": value,
                        "Field2": ["value1", "value2", "value3"]},
                        tableau.SelectionUpdateType.REPLACE);
                        }
            }
    }
    var viz = new tableau.Viz(containerDiv, url, options);
};
```

Apart from REPLACE type of mark selection are available:

1) ADD - Adds the values as specified in the call to the current selection. Equivalent to control-clicking in desktop.
2) REMOVE - Removes the values as specified in the call from the current selection. Equivalent to control-clicking an already selected mark in desktop.

Switching between dashboards

Tableau Javascript API provide switching between many dashboards. This solution required two buttons to move to next or previous dashboard. For example:

```
<button style="width:100px;" onclick="javascript:initViz(-1);">Previous</button>
<button style="width:100px;" onclick="javascript:initViz(1);">Next</button>
```

First of all, in array should be specified all dashboards. For them function is checking current position of dashboards in array.

```
function initViz(i) {
var myDashboards = [
          "http://YOUR-SERVER/views/YOUR-VISUALIZATION",
          "http://YOUR-SERVER/views/YOUR-VISUALIZATION2",
           //OTHER VISUALIZATIONS
      ];
      var vizLength= myDashboards.length;
      vizCount = vizCount + i;
      if (vizCount >= vizLength){
         vizCount = 0;
      }
      else if (vizCount < 0){
         vizCount = vizLength - 1;
      }
      if (viz){
         viz.dispose();
       }
      var vizURL = myDashboards[vizCount];
      //Rest of initViz function
}
```

Export to PDF

It is possible to export visualizations to an image or PDF file. For example the web page contains a button with onclick attribute.

```
<button onclick="exportToPDF();">Export</button>
```

Function exportToPDF() invoke for viz – container with visualization – function showExportPDFDialog() - witch shows a dialog allowing the user to select options for the export. It is also possible to export visualizations to an image. It is necessary to use showExportImageDialog(). Tableau JavaScript API does not support to export visualizations to PowerPoint file.

```
function exportToPDF() {
    viz. showExportPDFDialog();};
```

Supplement 1

```
function getdates()
{
	var end = new Date();
	var start = new Date();
	var dd = end.getDate();
	var mm = end.getMonth()+1;
	var yyyy = end.getFullYear();
	if(dd<10) {
		dd='0'+dd
	}
	if(mm<10) {
		mm='0'+mm
	}
	var difference = 3; // Here is defined differential between dates
	start = yyyy-difference +'-'+mm+'-'+dd;
	end = yyyy+'-'+mm+'-'+dd;
	return [start,end];
}
```

Example

```
var vizCount, viz, sheets,worksheet;

function initViz(i)
{
	var containerDiv = document.getElementById("vizContainer"),
	vizCount = 0;
	var dates = getdates();
	start = dates[0];
	end = dates[1];
	var myDashboards = [
		"https://public.tableau.com/views/Projekt_0/Dashboard1",
	    "https://public.tableau.com/views/Projekt_0/Dashboard2",
	];
	var vizLength= myDashboards.length;
    vizCount = vizCount + i;
	if (vizCount >= vizLength){
		vizCount = 0;
	}
	else if (vizCount < 0){
		vizCount = vizLength - 1;
	}
	if (viz){
		viz.dispose();
	}
	var vizURL = myDashboards[vizCount];
    options = {
		/*"Category" : ["Furniture", "Technology"],
		"Ship Mode" : ["Standard Day","First Class","Second Class"],*/
		"Start Date": start,   //Start Date is name of parameter
		"End Date": end,      // End Date is name o parameter
		hideTabs: true,
		hideToolbar: true,
```

```
            onFirstInteractive: function() {
                  sheet = viz.getWorkbook().getActiveSheet();
                  if(sheet.getSheetType() === tableau.SheetType.WORKSHEET) {
                        sheet.applyFilterAsync("Category",["Furniture",
"Technology"],tableau.FilterUpdateType.REPLACE).then();
                        sheet.applyRangeFilterAsync("Order Date",{
                              min : start,
                              max: end});
                        sheet.selectMarksAsync({"State": ["Virginia","New York"]},
tableau.SelectionUpdateType.REPLACE);
                  }
                  else{
                        worksheets = sheet.getWorksheets();
                        for(var i = 0; i < worksheets.length; i++) {

      worksheets[i].applyFilterAsync("Category",["Furniture","Technology"],tableau.FilterUp
dateType.REPLACE);
                              worksheets[i].applyFilterAsync("Ship Mode",["Same
Day","First Class","Second Class"],tableau.FilterUpdateType.REPLACE);
                              worksheets[i].applyRangeFilterAsync("Order Date",{
                                    min : start,
                                    max: end});
                              worksheets[i].selectMarksAsync({"State": ["New York",
"Virginia"]}, tableau.SelectionUpdateType.REPLACE);
                        }
                        worksheets[0].applyRangeFilterAsync("Quantity",{
                                    min : 8,
                                    max: 12});
                  }
            }
      }
    viz = new tableau.Viz(containerDiv, vizURL,options);
};

function exportToPDF()
{
      viz.showExportPDFDialog();
}

function getdates()
{
      var end = new Date();
      var start = new Date();
      var dd = end.getDate();
      var mm = end.getMonth()+1;
      var yyyy = end.getFullYear();
      if(dd<10) {
            dd='0'+dd
      }
      if(mm<10) {
            mm='0'+mm
      }
      start = yyyy-3+'-'+mm+'-'+dd;
      end = yyyy+'-'+mm+'-'+dd;
      return [start,end]; }
```