JavaScript API – Hands On Training

Confused? Miss a step? Your computer contains 9 **htdocs** **step x** folders you can use to catch up. Just take one of these folders, rename it to htdocs, and use it to replace the htdocs folder stored in **c:\tsi.portal.**

You will be working with two files:

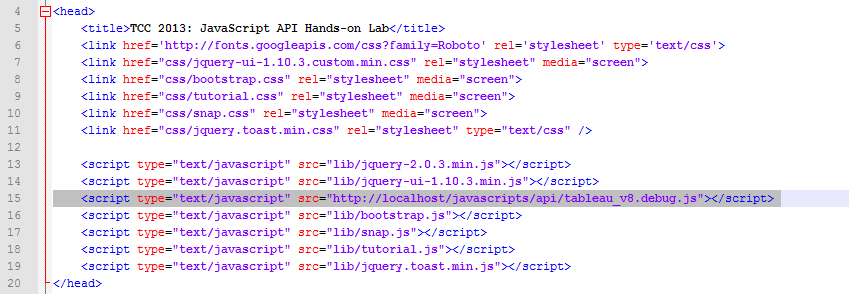
* **index.html**, stored in c:\tsi.portal\htdocs
* **tutorial.js**, stored in c:\tsi.portal\htdocs\lib

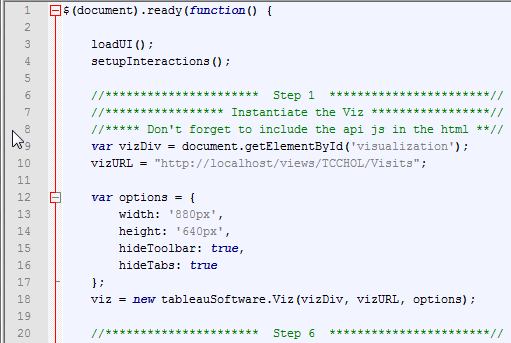
Instantiating a Tableau Viz – Step 1

To instantiate a viz using the Tableau JavaScript API, you must:

* Add a reference to the Tableau JavaScript API js file in your HTML page (index.html)
* Modify **/lib/tutorial.js** Create a new tableauSoftware.Viz with the code below

NOTE: Why are screenshots inserted below instead of code? – Because we’re mean and want you to **type the code**. If you really need to catch up with a copy-and-paste job, use the “saved” htdocs folders mentioned above.





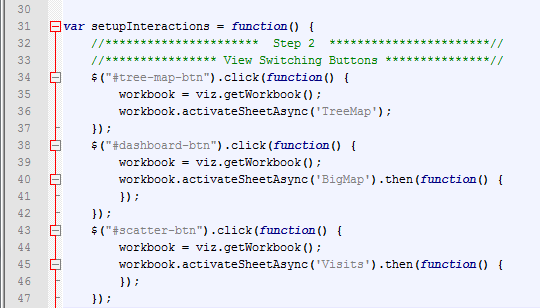
After completing this step, you should be able to instantiate a single scatterplot viz in the web page. Don’t be concerned about the map, combo-boxes, and calendar controls which appear, then disappear as the viz renders – you’ll fix that behavior later.

Switching Views with Buttons – Step 2

Index.html contains three buttons:

* tree-map-btn
* dashboard-btn
* scatter-btn

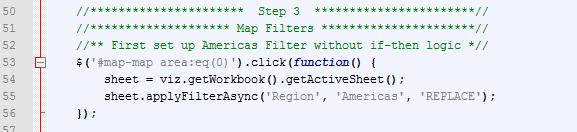
In section **Step 2**, wire up each button so that it activates the appropriate worksheet or dashboard inside the Tableau workbook when the button is clicked.



After adding a handler for each button’s click event, you should be able to switch back and forth between views using the buttons in the top menu bar.

Filtering Views with a Map Control – Step 3

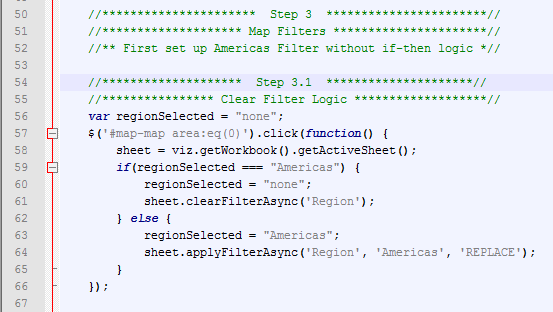
Next, you will write code so that clicking on the **Americas** region of the **Map** **Control** in the **Menu** pane will filter the **scatterplot** or **tree map** viz appropriately.



Once you have added the handler to deal with clicks on the **Americas** region, you should be able to filter the scatterplot or tree map viz. You will **not** be able to filter the map-based dashboard connected to the second button on your menu bar yet.

Removing the Americas filter – Step 3.1

You’ll want the ability to remove the filter, as well. Write an **if** statement that filters on **Americas** if the Americas region was selected and **removes** the filter if it is clicked a second time:



Dealing with sheets in a Dashboard – Step 4

The second visualization on the toolbar is a Dashboard. A **Dashboard** is itself a worksheet of type “dashboard”, and it contains other worksheets of type “worksheet”. In order to filter the nested worksheets inside a dashboard, you must reference them specifically.

Write code to test to see if the sheet being filtered or cleared is of type **worksheet** or **dashboard**. If the type is **not** worksheet, you must get the collection of worksheets nested inside the dashboard, and apply or clear the filter for each worksheet in the collection.

After you complete this step, you should be able to filter either the scatterplot, “map dashboard”, or treemap.



Repeat for other EMEA, Asia – Step 4.1

In the previous few steps you built up a solution which allows you to apply or clear a filter for a specific region (Americas) against a worksheet or dashboard. **Repeat** step 4 for the other two regions – EMEA (*#map-map area:eq(1)*) and Asia (*#map-map area:eq(2)*). Feel free to copy and paste!:



Implement Calendar Control filtering – Step 5

The **Menu** area contains two calendar controls which allow you to select a begin and end date for filtering. Write code which leverages **applyRangeFilterAsync** to apply a range filter to either a **worksheet** OR the sheets inside a **dashboard**:

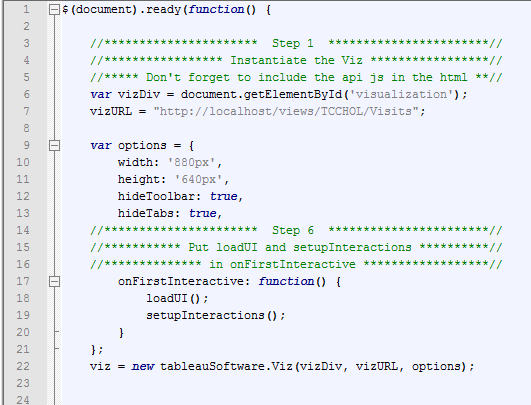


One your code is done; you should be able to filter all three vizzes by time.

Add onFirstInteractive instructions to viz rendering– Step 6

You’ve probably noticed that the user interface components in the **Menu** area appear while vizzes are rendering, then disappear. Let’s fix that by calling the functions which load the UI only after the visualization has been created.

* Delete references to loadUI() and setupInteractions() at the beginning of function **$(document).ready()**
* Move them to the **onFirstInteractive** option



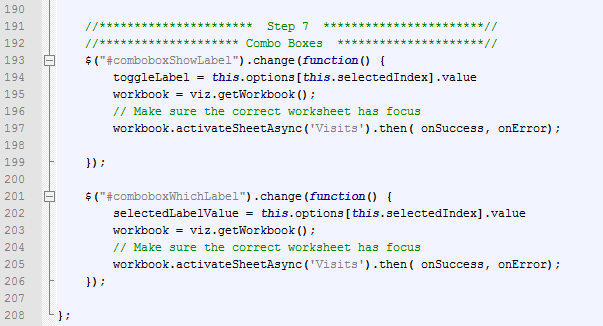
Once the code above is added, the web page should look much more “clean” when you initially run it.

Attach code to Drop-down-boxes – Step 7

Add code so that the two combo-boxes are able to change the behavior of the scatterplot viz:

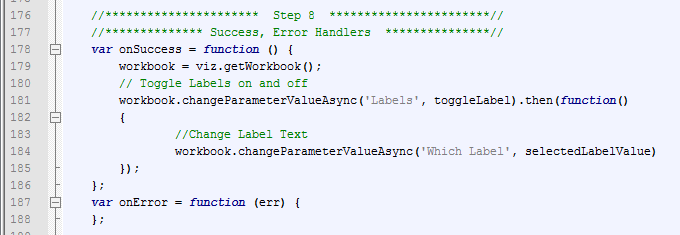
* Use the first combo-box to change a viz-specific parameter which controls whether labels are displayed or hidden
* Use the second combo-box to select which field should be displayed as the label

The code below will **call** fulfilled and error handlers (onSuccess and onError) which have not yet been defined. There is no way to test for success in this step – wait till step 8.



Define fulfilled and error handlers – Step 8

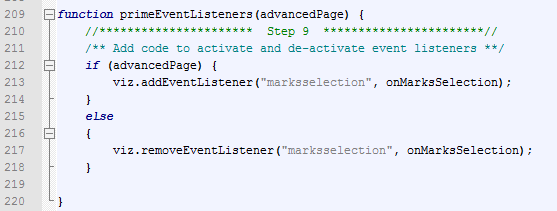
The code you implemented in step 7 calls handlers that must be added directly *before* step 8. The handlers will actually make calls to **changeParamterValueAsync()** in order to change the parameters which control viz label behavior.



Now that handlers which call changeParameterValueAsync() are in place, you should be able to change the label behavior of the scatterplot viz.

Turn Event Listeners On and Off – Step 9

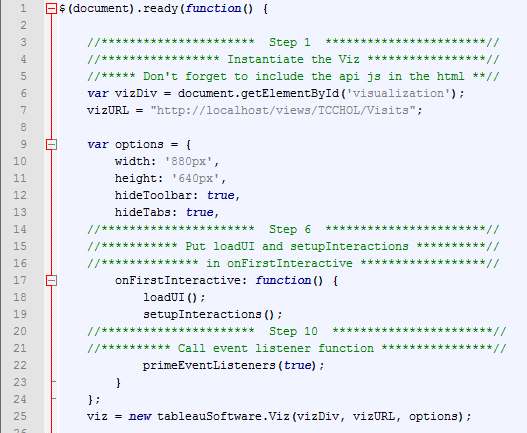
Your page requires a mechanism to enable and disable event listeners. Add it to towards the bottom of tutorial.js:



Continue to steps 10 and 11 before trying to test your code.

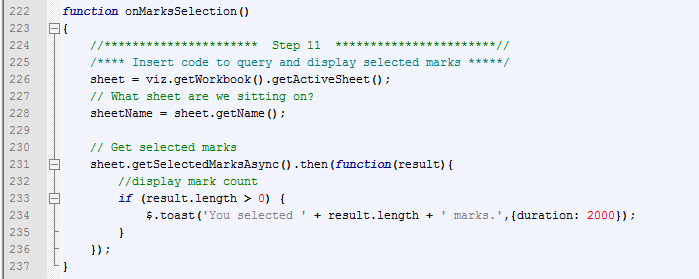
Call primeEventListeners() during onFirstInteractive function – Step 10

Add another line of code to onFirstInteractive which turns event listeners on:



Add event handling logic – Step 11

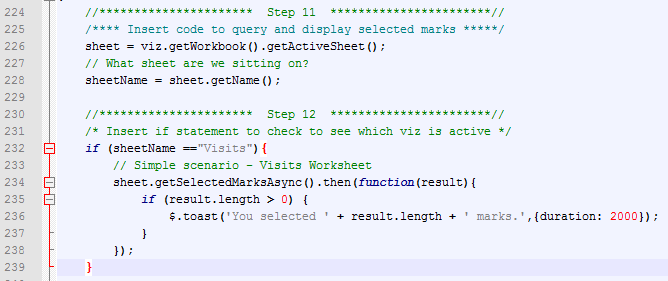
Finally, add logic to the **onMarksSelection** function to getSelectedMarksAsync() from your scatterplot viz and display the result if at least one mark has been selected.



At this point, you should be able to select marks in the scatterplot and cause a toast alert to appear in the upper-left side of your browser.

Advanced Event Listeners, Part I – Step 12

We want to handle mark selection event differently based on whether the **Visits** scatterplot view or the **Big Map** dashboard is in play. Wrap the code you just wrote in an if statement so that it only fires when **Visits** is selected.

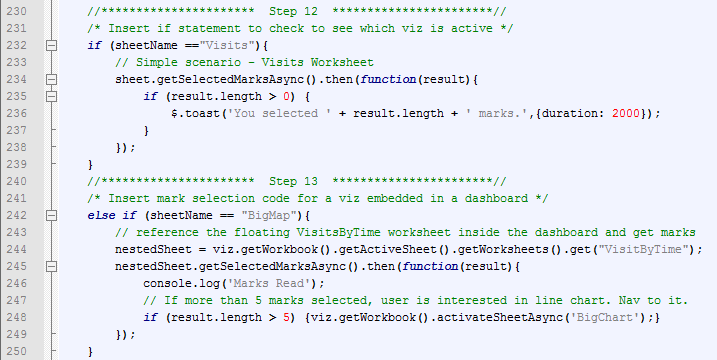


Advanced Event Listeners, Part II – Step 13

You’re almost there.

Continue the if statement above by adding an else-part so which checks to see if the **Big Map** dashboard is active. If it is, your code will need toL

* Select the small nested **VisitsByTime** viz (it’s a small line chart)
* Check how many marks have been selected
* Navigate to a different dashboard if over 5 marks are selected.



That’s it. Navigate to the Big Map dashboard and select a single mark in the line chart. The viz should filter to a single region.

Select over 5 marks, and you should navigate to a “clone” of this dashboard in which the time series is large, and the map is small. Click on the small map to return.