Due: Monday July 18 (11:59PM)

Submission

Put your assignment in a folder named assign3_*userid* (e.g., assign3_ealmasri). Put all resources used by your assignment into this folder. Compress this folder into a zip file (call it *assign3_userid.zip*). Submit a copy of this compressed file via Canvas (Assignments → Assignment 3).

Overview

The intent of this assignment is to demonstrate your understanding of AJAX and jQuery. In this assignment, you will use jQuery AJAX to consume and display JSON data. The assignment will also make use of a third-party JavaScript library to display charts of that data. The required starting files to complete the assignment can be found in a compressed file called ***assign3-start.zip*** which can be downloaded via Canvas (Assignments → Assignment 3 → assign3-start.zip).

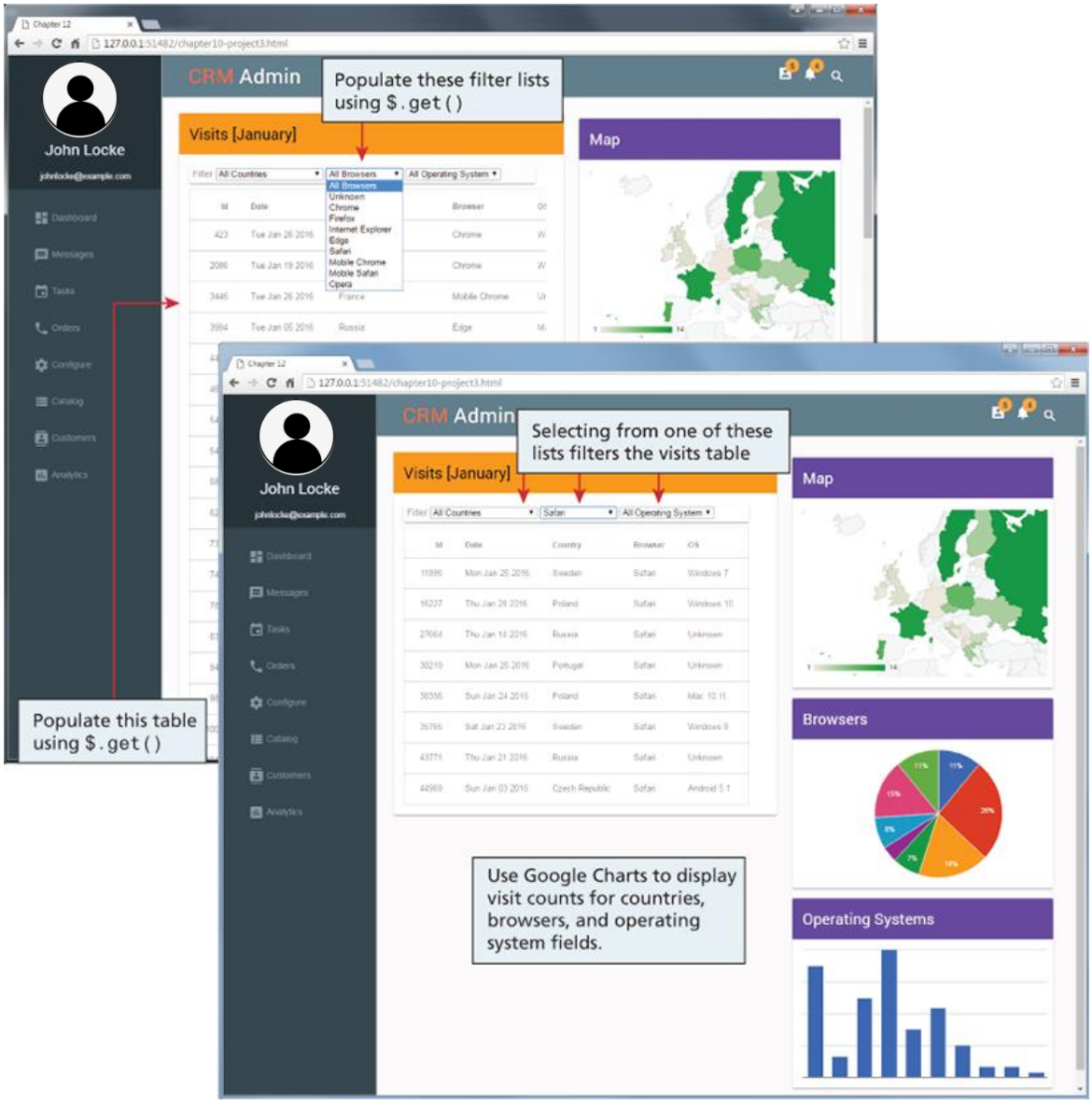

Figure 1: Mockup of Assignment 3

Instructions

Some of the subsequent steps have already been implemented in the start-up code. However, they are provided to guide you on what the startup code does.

1. Examine assign3.html in the browser and then editor. You have been supplied with the appropriate CSS files as well. This assignment can be a bit overwhelming, so we advise breaking it down into smaller steps: at each step below, test to ensure it works. To assist you in completing the assignment, some functionality has already been coded and you are asked to complete the sections (or methods) that require completion. That is, use the instructions and the provided code to complete the sections that have the following comment (in /js/assign3.js).

   ```
   //TO DO : complete this function ....
   ```

2. First, you will populate the #filterBrowser <select> list with a list of browsers. The data for this list is going to be retrieved using the $.get() method. The URL for the web service is as follows:

   http://www.randyconnolly.com/funwebdev/services/visits/browsers.php

   You may want to first examine the JSON that is returned (simply by entering this URL into a browser window). You will notice it returns an array of objects: each object contains the browser id and name.

   Now you want to programmatically retrieve this information using the $.get() method. When the data is retrieved (i.e., within the .done() handler) you will loop through the returned data and add an <option> element to the #filterBrowser <select> list for each browser in the returned data. Be sure to set the value attribute for each <option> to the id property (e.g., <option value="2">Chrome</option>).

3. Do the same for the countries and operating system <select> lists. The URL for the operating system list web service is as follows:

   http://www.randyconnolly.com/funwebdev/services/visits/os.php

   The URL for the countries is as follows:

   http://www.randyconnolly.com/funwebdev/services/visits/countries.php?continent=EU

   Notice that the countries service is expecting a query string parameter. If you don't include it, your country list will have all the countries in the world. We want just the countries from Europe.

4. Now you are ready to display the visits table. This data will also be retrieved from an external service. The URL is as follows:

   http://www.randyconnolly.com/funwebdev/services/visits/visits.php

We will also add the following querystring to this URL:

```
continent=EU&month=1&limit=100
```

This ensures the visits data includes only those from Europe, in the month of January, and returns 100 records.

Using the same techniques as with the above <select> lists, you will programmatically add <tr> and <td> elements to the provided <tbody id="visitsBody"> element.

5. Add event handlers for the change events of the three filter <select> lists. When the user selects an item in one of these lists, empty the <tbody> element, and redisplay the table showing only those visits that match the selected filter. Hint: the $.grep() method provides an easy way to create a new array of objects based on a filter condition.

   Your code shouldn't need to perform any more external data retrievals for this (or subsequent) steps. You just need to maintain the retrieved visit data in a variable that persists after the retrieval.

6. Add the three charts using Google Charts (the <script> tags for these libraries are already provided in the assign3.html file). The documentation for these charts can be found at https://developers.google.com/chart/. The API for these charts is pretty straightforward. Much of the code can be copy-and-pasted from the online documentation into your code. You will need to replace the hard-coded example data in the documentation with data you will construct from the visits data you have already downloaded.

   For the map/geo chart, you will need to construct an array of visit counts for each country. For instance, in the visits data, there are 15 visit records with country=Russia, 14 records with country=Sweden, and so on. This array of country names and visit counts will be passed into the google.visualization.arrayToDataTable() method.

   You will need to do the same thing for the pie chart (visit counts for each browser type) and column chart (visit counts for each OS type).

   Note: Google Charts is an external API, and as such, it could change from time to time (May 2018) and when you are reading it. As a result you may need to make adjustments to your code that are not mentioned here.

Testing

We strongly suggest testing each step before moving on to the next step.