THE UNIVERSITY OF SYDNEY

School of Information Technologies
Dr. Ying Zhou

COMP5347: Web Application Development **1.Sem./2017**

Week 10 jQuery And Google Charting

Learning Objectives

- Get familiar with jQuery selectors and syntax
- Practice Ajax requests
- Practice basic Google Charting features

Question 1: Simple Charting Application

Download `week10-src.zip` from ELearning and extract the content. The zip file contains a client side script `main.js`, a serve side script `plot.js` and a server side template `entry.pug`.

You can reuse the previous project and put the three files in their respective folders. As an example: `plot.js` should be put under project's root folder; `entry.pug` should be put under `app/views`; the client side `main.js` should be put under `public/js` folder. We consider client side javascript files as static resources similar to CSS files. You will need to create a `js` folder under `public` folder.

You can run `plot.js` as `node.js` application. Once the server is started, you can access it from browser with the address:`localhost:3000/`. This would display a simple page with two links. If you click the first link "Pie Chart", a pie chart would display. If you click the second link, you will get an error message.

Question 2: Understand the code

The simple application does not use MVC structure. All server side processing logics are implemented in `plot.js`. Open `plot.js` in Eclipse and inspect the content. The script sets view and static folder as usual; it also defines two route methods. The method mapped to `GET /` renders the view `entry.pug`. The method mapped to `GET /data` demonstrates a slightly different response handling method. It does not send a text or render any view, instead it sends back a hard coded json object using `res.json()` function. The json object response is meant for client side script. But you can view it from browser by sending a request to `localhost:3000/data` from your browser.

ExpressJS has predefined functions to configure the response in various ways, for a full list see reference document `https://expressjs.com/en/api.html#res`.

Now open `entry.pug` and inspect its content in Eclipse. The `body` element of `entry.pug` contains two anchor links. You may notice that there is no mapping route method defined for any in `plot.js`. Both links are meant to be processed by client side javascript code. The `head` element of `entry.pug` contains links to three scripts. The first two are third party libraries: jQuery and Google Charts loader. The last one is our own script: `main.js`

Majority of the processing in this application happens on client side and is implemented in `main.js`. Charting related functions and statements in this script are adapted from Google Charts Tutorials.(`https://developers.google.com/chart/interactive/docs/quick_start`)

The first statement in this script calls the `load()` function to load the `corechart` package. This statement is required for any web page using Google Charts. Two global variables: `options` and `data` are declared in the next two statements. A global function `drawPie()` is declared to draw a pie chart based on the chart data stored in `data` variable. The chart data is stored as javascript object, the next few statements follow google Charts tutorial (`https://developers.google.com/chart/interactive/docs/basic_load_libs`) to prepare the `DataTable`. jQuery's `$.each()` function is used to iterate all key value pairs in the javascript object. After `DataTable` is prepared, a pie chart is created on the designed container element: `$("#myChart")[0]`. This is equivalent to the raw JavaScript expression `document.getElementById("myChart")`. The chart.draw() statement will update the element with the chart content.

The next statement registers an even handler on the `document.ready` event:

```
$(document).ready(function() {
...
}
```

Inside the event handler, an ajax request `$.getJSON(url,data, callback)` is sent to get the chart data from server. This request is similar to `$.get()` and `$.post()` except that it receives JSON data as response. The response is assigned to the global variable `data`, so that it is available to all scripts on this page.

It also registers an event handler to the click event of the link "Pie Chart". The event handler disables the default action of the link and calls the `drawPie()` function.

Question 3: Write your own code

Now write your own code to display a bar charts when a user clicks the "Bar Chart" link. You can find tutorial on bar chart here: `https://developers.google.com/chart/interactive/docs/gallery/columnchart`. Note that Google Charts uses different terminologies for vertical and horizontal bar charts.