CSCI 3308 Software Methods & Tools [Fall 2015]
Instructor: Boese
Lab 6

Web Services - REST APIs

Objectives

- Start a local web server
- Create a web page and display on a web server
- Use jQuery's Ajax to get data from a REST API web server
- Parse JSON data and display on a web page

PRE-LAB work:

Ensure you have a web server installed

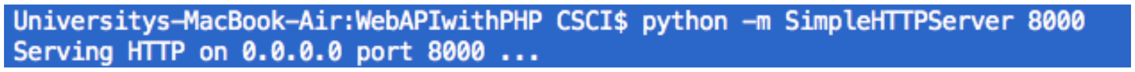
1. You might be all ready to go, check to see what happens if you type in a terminal window:

```
python -m SimpleHTTPServer 8000
```

or

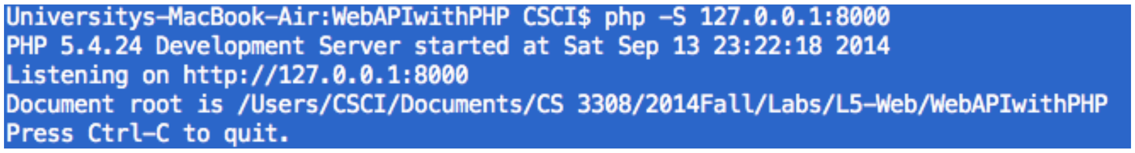
```
php -S 127.0.0.1:8000
```

You should see something like:



```
Universitys-MacBook-Air:WebAPIwithPHP CSCI$ python -m SimpleHTTPServer 8000
Serving HTTP on 0.0.0.0 port 8000 ...
```

or



```
Universitys-MacBook-Air:WebAPIwithPHP CSCI$ php -S 127.0.0.1:8000
PHP 5.4.24 Development Server started at Sat Sep 13 23:22:18 2014
Listening on http://127.0.0.1:8000
Document root is /Users/CSCI/Documents/CS 3308/2014Fall/Labs/L5-Web/WebAPIwithPHP
Press Ctrl-C to quit.
```

Note: the CS department VM does have the python working so you can use that.
Windows users can try installing Cygwin…
Also see slides about MAMP, WAMP, XAMPP as other [more complicated] options…

2. Learn the basics of how HTML and JavaScript work, if you don't know it already. It will help to understand what you are doing. We will walk you through what you need for this lab, it just helps to know more. You may also want to read up on Ajax.

Please pair-program today!

Description

The goal of this project is to set up a client-side web program using JavaScript, the jQuery library, and AJAX to access data on a web server via REST API.
More specifically, you will create a Web application that displays dynamic information on the current weather conditions of different cities.

Setting up your computer

-- This will only work on your computer if you have it set up correctly (did the pre-work to ensure). --

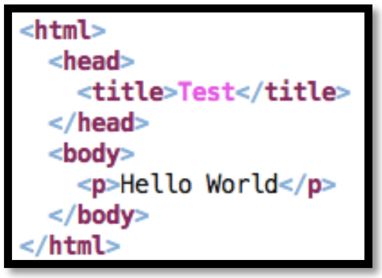
Please work in pairs on this lab.

Step 1: Open a terminal window.

Step 2: Go to a directory (or create a new directory) to work from.

Step 3: Using vim or emacs, create a new web page titled 'hello.html'.

Step 4: Add the following HTML code to the file:



```
<html>
  <head>
    <title>Test</title>
  </head>
  <body>
    <p>Hello World</p>
  </body>
</html>
```

Step 5: Save the file.

Step 6: Open a second terminal window, and go to the same directory. *(On the VM, move the mouse to the gray bar of the terminal window and go to File → Open Terminal).*

Step 7: Run either of the commands below to start the simple web server using the current directory as the base :

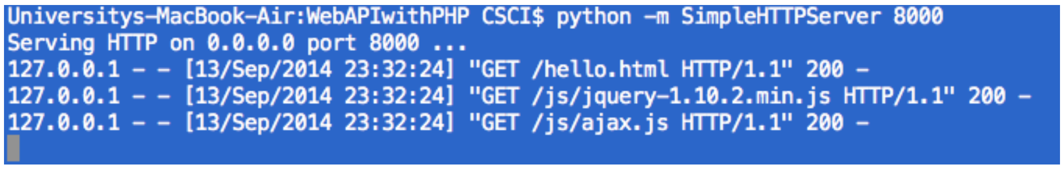
```
python -m SimpleHTTPServer 8000
```

or

```
php -S 127.0.0.1:8000
```

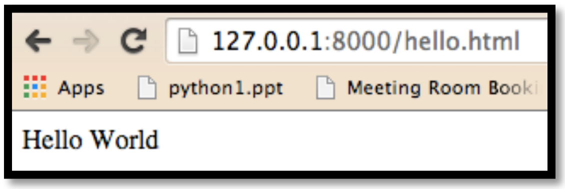
Note: 127.0.0.1 is a.k.a. 'localhost' and is the computer you are currently on. 8000 is the port.

You are going to keep this window open as you will see the logs append to this screen.



```
Universitys-MacBook-Air:WebAPIwithPHP CSCI$ python -m SimpleHTTPServer 8000
Serving HTTP on 0.0.0.0 port 8000 ...
127.0.0.1 - - [13/Sep/2014 23:32:24] "GET /hello.html HTTP/1.1" 200 -
127.0.0.1 - - [13/Sep/2014 23:32:24] "GET /js/jquery-1.10.2.min.js HTTP/1.1" 200 -
127.0.0.1 - - [13/Sep/2014 23:32:24] "GET /js/ajax.js HTTP/1.1" 200 -
```

Step 8: Open your web browser and open the file 'hello.html' via the local web server: http://127.0.0.1:8000/hello.html



Hello World

(If you are using the VM, you need to open a browser within the VM)
<u>Step 9:</u> We are going to use the API from Open Weather Map because it is one of the few sites that do not require authentication (creating an account, adding userid and authentication key, etc).

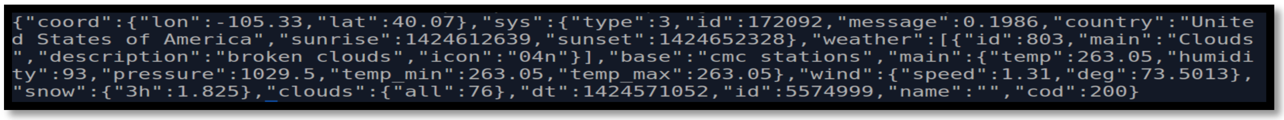
You can see what options they have at: http://openweathermap.org/current

We will start by getting the weather of Boulder.

They show the curl option:

```
curl 'api.openweathermap.org/data/2.5/weather?q=Boulder,CO'
```

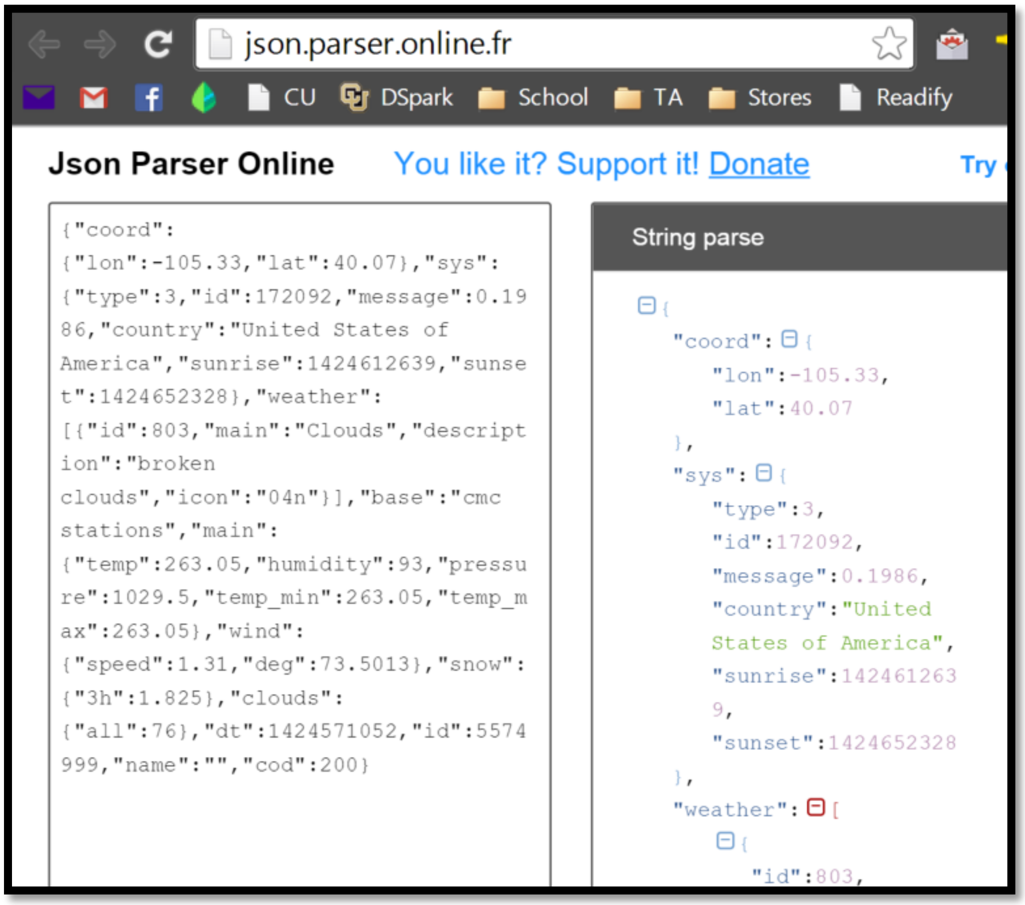
In your terminal window, type this in and press enter. You should see the JSON returned:



```
{"coord":{"lon":-105.33,"lat":40.07},"sys":{"type":3,"id":172092,"message":0.1986,"country":"Unite
d States of America","sunrise":1424612639,"sunset":1424652328},"weather":[{"id":803,"main":"Clouds
","description":"broken clouds","icon":"04n"}],"base":"cmc stations","main":{"temp":263.05,"humidi
ty":93,"pressure":1029.5,"temp_min":263.05,"temp_max":263.05},"wind":{"speed":1.31,"deg":73.5013},
"snow":{"3h":1.825},"clouds":{"all":76},"dt":1424571052,"id":5574999,"name":"","cod":200}
```

One of the tools we use is to 'beautify' the JSON code so we can more easily read it.
An example website that does this for us is at: http://json.parser.online.fr/

Copy-paste the results returned from curl into this website:



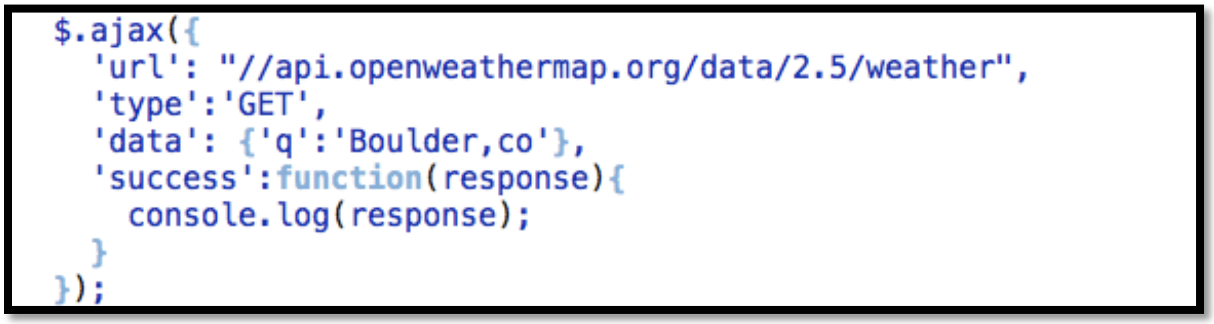
```
{"coord":
{"lon":-105.33,"lat":40.07},"sys":
{"type":3,"id":172092,"message":0.19
86,"country":"United States of
America","sunrise":1424612639,"sunse
t":1424652328},"weather":
[{"id":803,"main":"Clouds","descript
ion":"broken
clouds","icon":"04n"}],"base":"cmc
stations","main":
{"temp":263.05,"humidity":93,"pressu
re":1029.5,"temp_min":263.05,"temp_m
ax":263.05},"wind":
{"speed":1.31,"deg":73.5013},"snow":
{"3h":1.825},"clouds":
{"all":76},"dt":1424571052,"id":5574
999,"name":"","cod":200}
```

```
{
    "coord":{
        "lon":-105.33,
        "lat":40.07
    },
    "sys":{
        "type":3,
        "id":172092,
        "message":0.1986,
        "country":"United
        States of America",
        "sunrise":142461263
        9,
        "sunset":1424652328
    },
    "weather":[
        {
            "id":803,
```

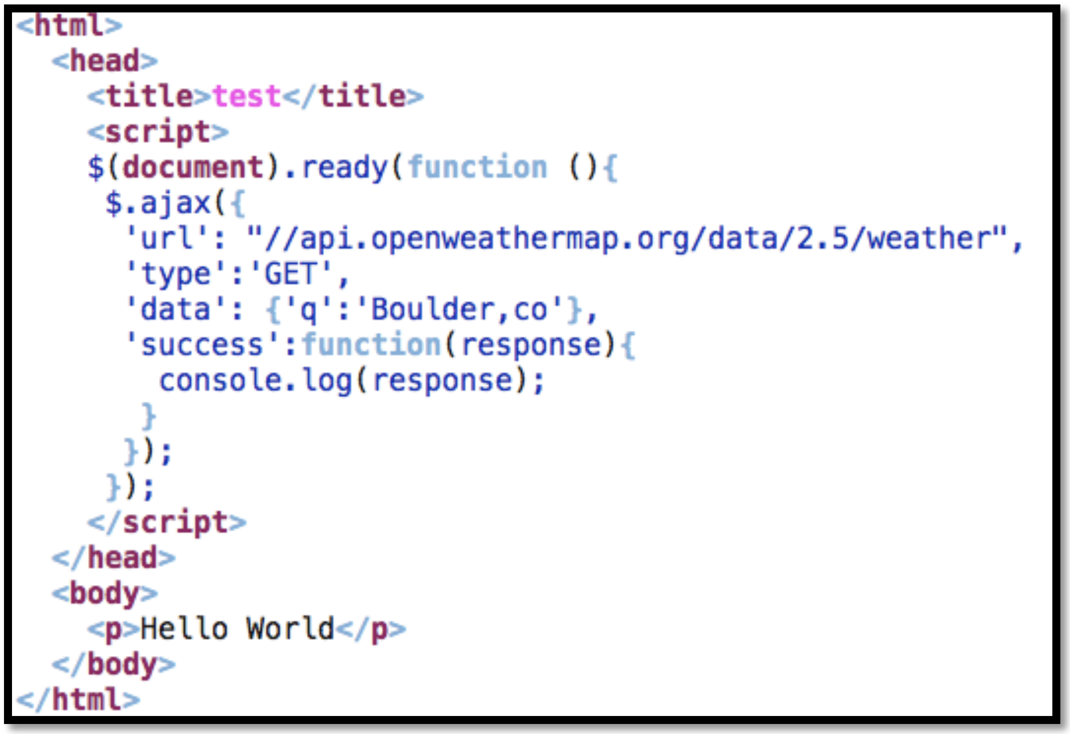
Step 9: We're going to use the jQuery ajax code inside our web page. Open your .html file and add the code from Open Weather Map's website. We are going to need to put this code inside jQuery's document.ready function (http://www.learningjquery.com/2006/09/introducing-document-ready/)

jQuery (http://jquery.com/) is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. (aka Amazing!) *http://learn.jquery.com/about-jquery/how-jquery-works/*
We need jQuery to handle our calls to the REST API that Open Weather Map has provided to us. We do this via Ajax (Asynchronous JavaScript and XML).

The code below is the ajax code that will query Open Weather Map. It specifies the URL for Open Weather Map, the method is GET, and the data is what is passed as the query. If the ajax call is successful, it calls the 'success' function with the results returned from the server into the variable named 'response'.



```
$.ajax({
  'url': "//api.openweathermap.org/data/2.5/weather",
  'type':'GET',
  'data': {'q':'Boulder,co'},
  'success':function(response){
    console.log(response);
  }
});
```



```
<html>
  <head>
    <title>test</title>
    <script>
    $(document).ready(function (){
     $.ajax({
       'url': "//api.openweathermap.org/data/2.5/weather",
       'type':'GET',
       'data': {'q':'Boulder,co'},
       'success':function(response){
         console.log(response);
       }
     });
    });
    </script>
  </head>
  <body>
    <p>Hello World</p>
  </body>
</html>
```

Step 10: If you try to run it, you won't see the results yet.
First thing we need to do is include the jQuery library in our code.

Type the following inside your head tags:

```
<script type="text/javascript" src="//code.jquery.com/jquery-1.11.0.min.js"></script>
```

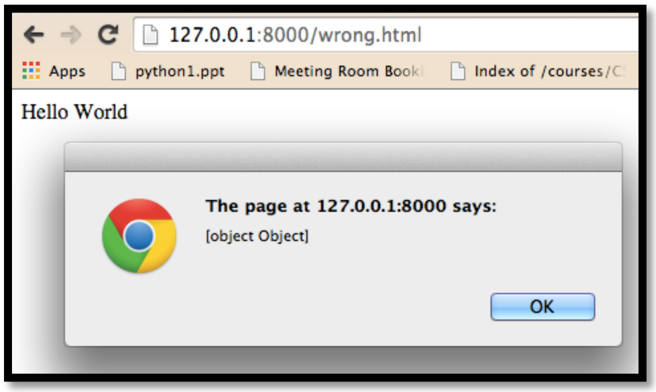
Note: while you can have your web page go access the library on the Internet, most web sites will download the library, put it in a directory named 'js' for JavaScript, and reference that.

Step 11: We're still not seeing the results. So let's add a JavaScript 'alert' function to bring up a pop-up dialog with the value of our 'response' variable.

Add the following line after console.log…

```
alert(response);
```

Now run it. You should see the dialog box:



Hello World

The page at 127.0.0.1:8000 says:
[object Object]

But that doesn't help –
our variable has '[object Object]'!

Step 12: When we receive JSON data, it is represented as an Object. Now we need to be able to parse that object. We do that with dot notation.

To understand this better, check out http://www.w3schools.com/json/tryit.asp?filename=tryjson_parse
Here is the JavaScript from that example:

```
<script>
var text = '{"employees":[' +
'{"firstName":"John","lastName":"Doe" },' +
'{"firstName":"Anna","lastName":"Smith" },' +
'{"firstName":"Peter","lastName":"Jones" }]}';

obj = JSON.parse(text);
document.getElementById("demo").innerHTML =
obj.employees[1].firstName + " " + obj.employees[1].lastName;
</script>
```

Notice how our data is similar, except that you can also simply create your own JSON data right inside your web page!

In the example above, you can see there is a list of employees. Our data is formatted the same, even though we only got one result back.
In this example, you access the data directly from the variable:

```
obj.employees[1].firstName
```

For our example, our variable is named 'response'. So we have:

```
response.results[0].DayCount
```

Change your alert to now read as:

```
alert(response.main.temp);
```

This should give you a proper count!

<u>Step 13:</u> The last step is to take this information that we can now access from the JSON response and add it to the web page.

We will again make use of jQuery to help us.

Remove the alert line (or comment it out).
Instead add the following line:

```
$('#results').text(response.main.temp);
```

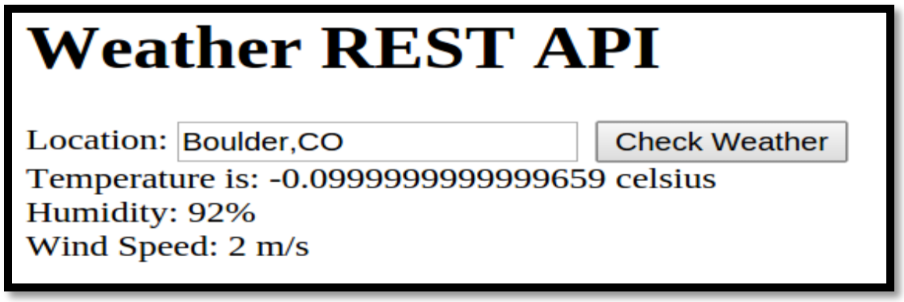
Also add the following code after the 'Hello World' line:

```
<p id="results">original</p>
```

This is jQuery's way of updating the content within an HTML tag. The `#results` references any HTML tag that has the *attribute* `id` equal to `'results'`. The `.text( )` replaces the text inside the tags with the new text.

<u>Step 14:</u> Add the rest of the data from the bicycle JSON object to the web page. Be sure to label each thing that is displayed. *Hint, you can use .append instead of .text to keep adding to the same HTML tag, OR add multiple p tags with different IDs and reference those to add additional fields.*

Your output should look like the following:


Weather REST API

Location: Boulder,CO

Temperature is: -0.0999999999999659 celsius
Humidity: 92%
Wind Speed: 2 m/s

Feel free to add bold or other pretty formatting. If you want to explore adding other information you might want to check out this page to know what units are being used. http://openweathermap.org/weather-data#current

To get credit for this lab exercise, show the TA and sign the lab's completion log.

Here is a list of other free REST APIs that don't require authentication:
Iceland: http://apis.is/
Flickr has a free public REST API: https://www.flickr.com/services/feeds/docs/photos_public/
Google Maps: https://maps.googleapis.com/maps/api/geocode/json?address=Boulder%20Colorado,%20uk&sensor=false
Others: http://shkspr.mobi/blog/2014/04/wanted-simple-apis-without-authentication/

Other REST APIs that require you to set up authentication with an API key…
Movie Database: http://docs.themoviedb.apiary.io/reference
Twitter: https://dev.twitter.com/overview/documentation
Weather: http://openweathermap.org/api