Assignment #5

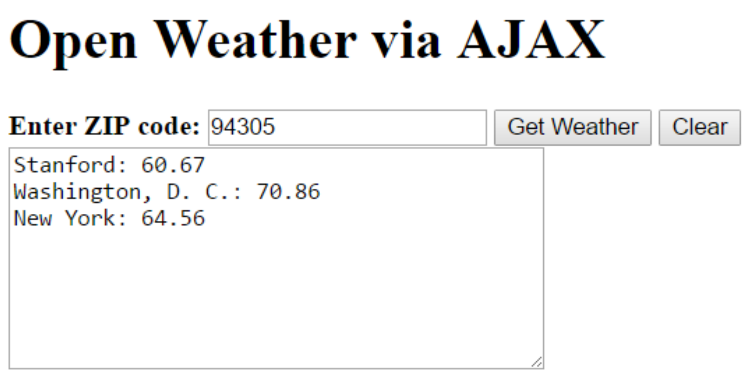
AJAX & JavaScript Libraries and Frameworks

CS193C Summer 2017, Young

For our last assignment we will experiment with AJAX and JavaScript Libraries and Frameworks. This assignment is due Wednesday August 16th at 2:30pm. In order to get all your assignments graded in time to make the end-of-quarter grades deadline, no assignments will be accepted after Thursday the 17th at 11:59pm.

AJAX

For the AJAX section of our assignment we'll retrieve weather information from openweathermap.org. Here is a screenshot showing the AJAX assignment in action:



```
Stanford: 60.67
Washington, D. C.: 70.86
New York: 64.56
```

The user enters in a ZIP code and clicks on "Get Weather". The weather for that ZIP code is retrieved from openweathermap.org and added to a textarea. Weather information listed should include the City corresponding to the ZIP code and the temperature (which is provided to us by openweathermap.org). The textarea should list all requests made. The user can click on the "Clear" button to clear the textarea.

We'll need to teach you a few things in order to get this assignment up and running.

Creating an Account with OpenWeatherMap.org

Go to:

http://openweathermap.org/

Sign Up for an account (upper-right of website). Once you've created an account, go to your account summary and switch to the "API Keys" tab. You'll need to have an API Key to communicate with the server.

Requesting Weather Reports

To make a request to the server for the weather at a particular zipcode the request format is:

http://api.openweathermap.org/data/2.5/weather?zip=*zipcodeDesired*,us&units=imperial&APPID=*yourAPIKey*

where *zipcodeDesired* is replaced by the zipcode you want and *yourAPIKey* is replaced by your account's API key.

Getting Weather Information

You'll pass your URL to OpenWeatherMap.org. After a brief delay, you should get a response providing the weather at the location.

If you use the request above, you'll get your request back as a JSON. This JSON may be found on the XMLHttpRequest object's responseText property. To convert that to an actual object you can do something like this:

```
var result = JSON.parse(requestObj.responseText);
```

If you prefer working with XML you may add the following to the end of your request URL:

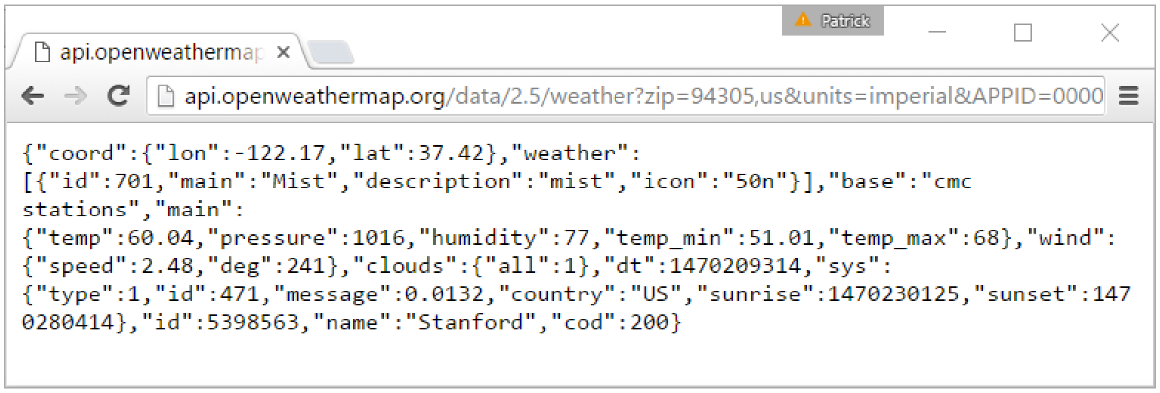
&mode=xml

In this case the resulting XML may be found at:

```
var result = requestObj.responseXML;
```

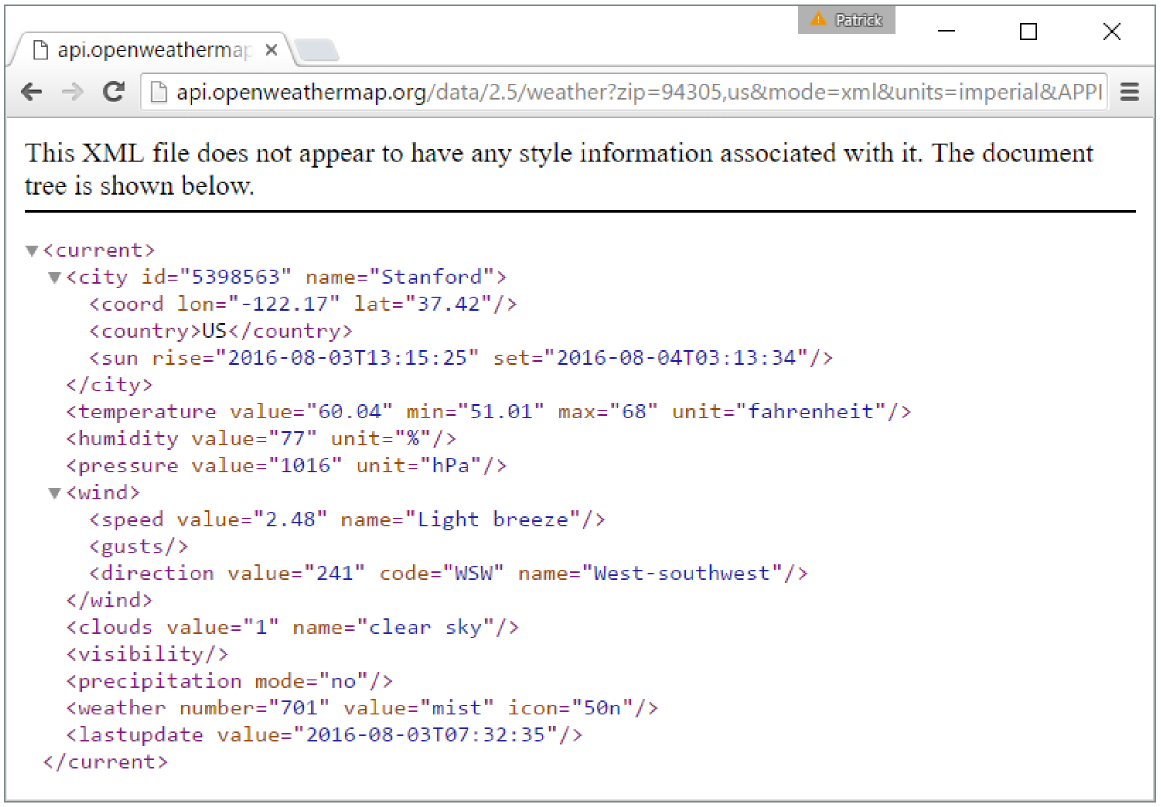
you can then move around the resulting XML tree using the getElementById, getElementsByTagName, and the other standard operations available for moving around the HTML tree.

To take a closer look at what the JSON or the XML look like, instead of using the URL above from JavaScript, instead just enter it directly into the web browser location bar and you should get a result back looking something like this:


Patrick
api.openweathermap
api.openweathermap.org/data/2.5/weather?zip=94305,us&units=imperial&APPID=0000

```
{"coord":{"lon":-122.17,"lat":37.42},"weather":
[{"id":701,"main":"Mist","description":"mist","icon":"50n"}],"base":"cmc
stations","main":
{"temp":60.04,"pressure":1016,"humidity":77,"temp_min":51.01,"temp_max":68},"wind":
{"speed":2.48,"deg":241},"clouds":{"all":1},"dt":1470209314,"sys":
{"type":1,"id":471,"message":0.0132,"country":"US","sunrise":1470230125,"sunset":147
0280414},"id":5398563,"name":"Stanford","cod":200}
```

Or like this:


Patrick
api.openweathermap
api.openweathermap.org/data/2.5/weather?zip=94305,us&mode=xml&units=imperial&APPI

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<current>
  ▼<city id="5398563" name="Stanford">
     <coord lon="-122.17" lat="37.42"/>
     <country>US</country>
     <sun rise="2016-08-03T13:15:25" set="2016-08-04T03:13:34"/>
   </city>
   <temperature value="60.04" min="51.01" max="68" unit="fahrenheit"/>
   <humidity value="77" unit="%"/>
   <pressure value="1016" unit="hPa"/>
  ▼<wind>
     <speed value="2.48" name="Light breeze"/>
     <gusts/>
     <direction value="241" code="WSW" name="West-southwest"/>
   </wind>
   <clouds value="1" name="clear sky"/>
   <visibility/>
   <precipitation mode="no"/>
   <weather number="701" value="mist" icon="50n"/>
   <lastupdate value="2016-08-03T07:32:35"/>
 </current>
```

This is what's in the responseText or responseXML of your XMLHttpRequestObject. You will probably find JSON easier to use, but you may use either for this assignment.

Don't forget Ajax uses a callback because often a server will take a while before it responds. You may have to wait quite a few seconds before your results show up.

JQuery

We’ll keep the JQuery section simple and similar to the in-class examples. Our objective is to just give you just a bit of hands on experience so that you’ll remember what you saw in lecture a bit better.

Start out with the `jquery-practice.html` file provided with this assignment’s downloads. This file contains no JavaScript and no JQuery. It does contain HTML along with some CSS Styles. It also contains several buttons which you’ll need to wire up to carry out various JQuery tasks.

Get JQuery Loaded

First things first, you need to get JQuery loaded. I’ve provided a JQuery file with the assignment downloads. Load it in using a standard <script> tag.

Turn Headings Red

Wire up the first button so that when the user clicks on it, all headings (h1, h2, and h3) turn red. Note that I’ve provided a style rule which you may find helpful for carrying out this task.

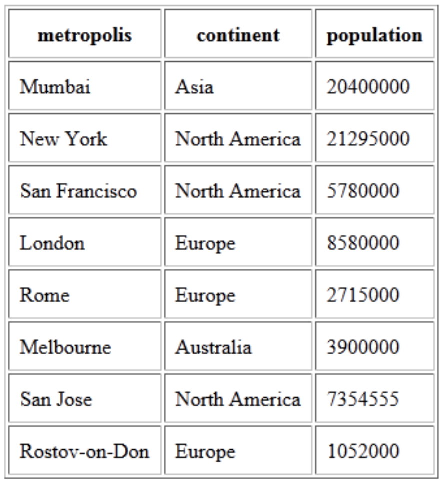
Fading Items

Wire up the second button so that when the user clicks on it the heading “Speakers” fades out over a 1 second (1,000 millisecond) period. Fade just the <h3> tag, not the subsequent paragraph on speakers. Once the heading has completely faded out, it will be removed from the normal text flow and the subsequent paragraph will be bumped up. This is normal and not something you need to correct.

AngularJS

Finally we create an application using AngularJS v1. See the Angular handout for more information. One caveat using the Angular format from the handout, angular directives won’t validate. So you may ignore these validation errors.[1]

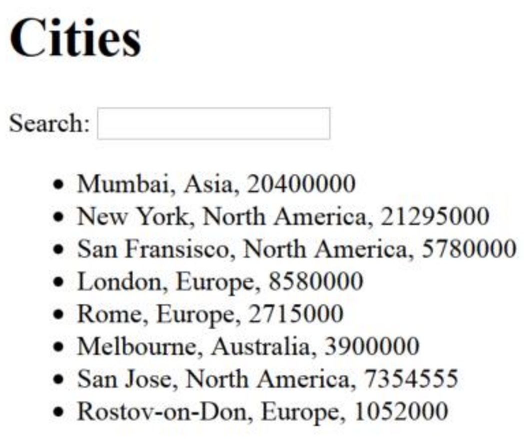
Create an AngularJS application which allows a user to display information on cities. Use the following actual city data:



metropolis	continent	population
Mumbai	Asia	20400000
New York	North America	21295000
San Francisco	North America	5780000
London	Europe	8580000
Rome	Europe	2715000
Melbourne	Australia	3900000
San Jose	North America	7354555
Rostov-on-Don	Europe	1052000

[1] As noted in the AngularJS handout, it is possible to get them to validate by adding a “data-” in front of each directive name. But your format won’t match the examples in the official Angular tutorial.

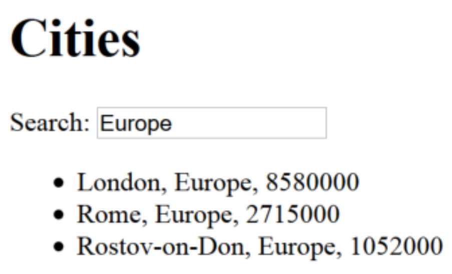
Include a single text field which can be used to filter based on information contained in any of the columns. Here are a few screenshots of the application in action:



Cities

Search:

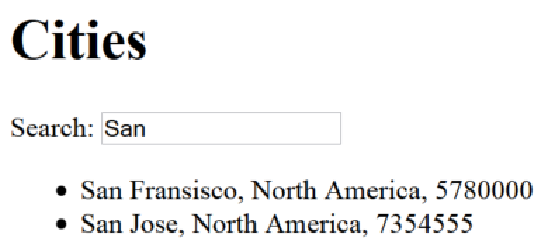
- Mumbai, Asia, 20400000
- New York, North America, 21295000
- San Fransisco, North America, 5780000
- London, Europe, 8580000
- Rome, Europe, 2715000
- Melbourne, Australia, 3900000
- San Jose, North America, 7354555
- Rostov-on-Don, Europe, 1052000



Cities

Search: Europe

- London, Europe, 8580000
- Rome, Europe, 2715000
- Rostov-on-Don, Europe, 1052000



Cities

Search: San

- San Fransisco, North America, 5780000
- San Jose, North America, 7354555

You may either use the more advanced Controller initialization scheme or initialize your data directly in the HTML file.