Assignment #4

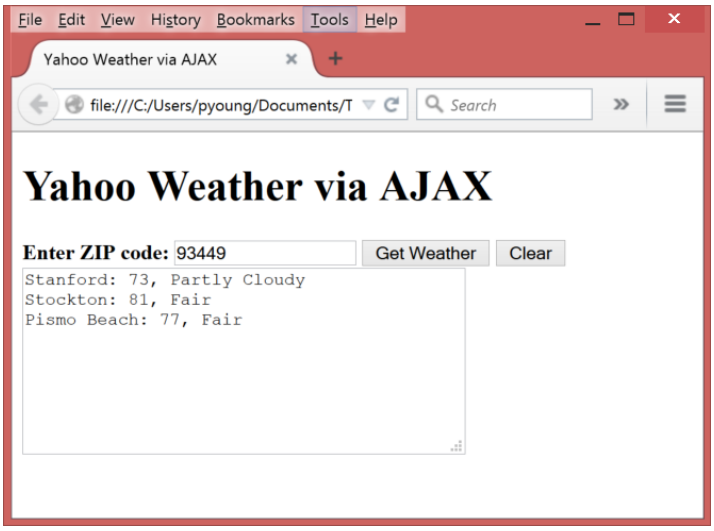
AJAX & JavaScript Libraries and Frameworks

CS193C Summer 2015, Young

For our last assignment we will experiment with AJAX and JavaScript Libraries and Frameworks. This assignment is due Thursday August 13th at 3:15pm. In order to get all your assignments graded in time to make the end-of-quarter grades deadline, no assignments will be accepted after Friday the 14th at 11:59pm.

AJAX

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



The user enters in a ZIP code and clicks on "Get Weather". The weather for that ZIP code is retrieved from Yahoo and added to a textarea. Weather information listed should include the City corresponding to the ZIP code, the temperature and a brief condition description (which is provided to us by Yahoo Weather). 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.

Requesting Weather Reports

You'll need to access weather reports from our proxy server. The URL for our proxy server is:

http://web.stanford.edu/~psyoung/cgi-bin/a3.php?yws_path=***encodedRequest***

where *encodedRequest* is a request to the Yahoo Weather webserver and is encoded using a URI encoding technique. This technique translates symbols to special encoded characters—for example the space character is encoded as %20. The request we're most interested in is weather

for a particular ZIP code. This request is specified as "p=*XXXXX*" where *XXXXX* is the actual ZIP code.[1] The '=' gets encoded as %3D.

Here is an example accessing the weather at Stanford (ZIP code 93405):

http://web.stanford.edu/~psyoung/cgi-bin/a3.php?yws_path=p%3D94305

You can encode your request programmatically using something like this:

```
var weatherURL = "http://web.stanford.edu/~psyoung/cgi-bin/a3.php?yws_path="
                        + encodeURIComponent("p=" + zipcode);
```

encodeURIComponent is a standard JavaScript function to encode a string.

Getting Weather Information

You'll pass your URL including the encoded request for the Yahoo Weather server on to our proxy via AJAX. After a brief delay, you should get an XML RSS response providing the weather at the location. Try typing this directly into the web browser and then using "View Page Source" in order to see what the XML looks like:

http://web.stanford.edu/~psyoung/cgi-bin/a3.php?yws_path=p%3D94305

Once you've gotten the XML back, you'll need to get to specific parts of the XML returned. For the most part, you can move around in the returned XML just as you would within your own XHTML document (e.g., using getElementsByTagName). There are two things you need to know about working with the XML.

- The XML elements used something called an XML namespace. For example, instead of using a location tag, Yahoo returns a yweather:location tag in with the actual location name corresponding to the ZIP code entered. The yweather indicates that in this case location is part of the yweather XML namespace. Here is the actual tag:

  ```
  <yweather:location city="Stanford" region="CA" country="US" />
  ```

  We can access this element by using the getElementsByTagName method on the XMLHttpRequest object's responseXML. Unfortunately Firefox and IE both expect the tag name to be preceded by yweather, i.e., getElementsByTagName("yweather:location") whereas Chrome just wants "location". So you'll need to try retrieving both. This call returns an array of elements, if you access using the "yweather:location" and the array returned has length zero, try accessing using just "location".

- Once you get to the element, you'll need to access the attribute value. We haven't discussed this before, but it's actually rather simple. Just call getAttribute(*attrName*) and the value of the attribute will be returned.

- Use the same procedure with the yweather:condition tag to get the temperature and condition information.

[1] There are a few fancier things you can do as well (see http://developer.yahoo.com/weather/) for more information.

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

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

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.

Note: You may get a copy of the angular.js file from the lecture examples.