CS 1100: Web Development: Client-Side Coding / Fall 2016
Lab 13: AJAX (extra credit)

Description

This is an extra credit *semi-formal* lab. It's goal is to give you a taste of using AJAX. We'll also use the Flickr API.

Semi-formal means that it is based on one brief class period where we covered the basics of AJAX. This is also the last class period of the semester, which is why I want all of you to get this code running. For these reasons, the final code is provided. However, I *strongly recommend* that you type your own code and modify it here and there. Make sure you understand everything it does; if something is confusing - please ask! You will get credit for this exercise either way - so make it count!

AJAX stands for Asynchronous JavaScript and XML. In a nutshell, we use AJAX to send requests to the server and modify parts of the web page based on the server response ***asynchronously*** - without reloading the entire page. There are plenty of online resources and tutorials on ALAX (here's one, and here's another). However, implementing AJAX with raw JavaScript can be challenging for a novice. jQuery simplifies this quite a bit. If you look at AJAX outside of class, I recommend you start with this jQuery documentation: https://learn.jquery.com/ajax/

Today you will do the following:

1. You will write a simple AJAX-enabled program that fetches content from files and loads it into a div element on your web page.
2. ..But who needs text files when we can mess with the Internet! After you are done with loading your text files, you will use the Flickr API to load photos into your page tapping into Flickr's immense collection.

Part 0. Setup

We'll use a simplified version of the example we used in class. Download the code: lab13.zip (from elearning or the course website). The folder contains everything you need - even the js file with the code commented out.

You can see a live version here: http://weblab.uni.edu/sergey/lab13/ This page will be removed after the lab because it exposes a personal API key (more on that later).

AJAX can only be used when your code resides on a server. That server may be a local server setup on your computer; however, our lab computers are not setup that way this semester, so you will have to upload your code to the remote weblab server. Thus, today your mode of operation will be as follows:
- edit the file
- upload to weblab.uni.edu
- refresh browser
- repeat

As a reminder, here are your settings for your FTP/SFTP client:
- host: weblab.uni.edu
- protocol: SFTP
- logon type: ask for password
- user: your CatID user name

You must place your files inside the *web* directory. I suggest you create a lab13 directory inside the web directory and place your files there. That way, your webpage will be availabe at http://weblab.uni.edu/your-username/lab13/

Part 1. The basics: loading text files

Let's start with loading simple text files.

- In your lab13.js file, uncomment PART 1 (leave the title "PART 1" commented out)
- The first line attaches the loadContent event handler (which is a function name) to the "click" event triggered on any of the images in the "profiles" div.
- Next comes the loadContent function definition. The function does the following:
    - builds up the path to the text file (content/*id-of-clicked-image*.html)
    - calls the .get method on the jQuery object, passing the constructed url and an anonymous callback function. The callback is executed once the .get method retrieves the content: it gets our target div and loads it up with the retrieved content (stored in the function's parameter "response").

Save the file, upload all the lab13 files it to the weblab server. In your browser, go to the location to which you have uploaded the files. Now each time you click a profile image, the corresponding content will be displayed WITHOUT the entire page being reloaded.

Part 2. Loading JSON format.

Loading text or HTML is not always what we want. In fact, usually we would prefer to retrieve specific data and only then generate the appropriate HTML. However, for this we need a format that, ideally, both we and the computer can understand. There are several options, one of them is JSON.

JSON stands for JavaScript Object Notation. It is a format used for sending/receiving structured text (data). It is machine- and human-readable. Here are some good resources:

http://www.json.org/
http://www.w3schools.com/js/js_json_intro.asp

In our case, we need JSON to explore the Flickr API (or almost any other API we might have chosen to play with). First, let's try to load a simple JSON file: content/test.json.

- First, open the test.json file in your text editor. As you see, it's a very simple data set consisting of 3 key-value pairs; the 3 keys are "one", "two", "three"; and the 3 values are "this is value 1", "this is value 2", "this is value 3". Let's say, we want to display the value of ley "two".
- Uncomment PART 2 in your lab13.js file. It does almost the same thing as the code you used before, except this time we call the .getJSON method. The callback function's parameter contains the retrieved data; however, this time the data is not just text, but structured text. To access the data we need, we simply use "data.two" (with "two" being the key to data "this is value 2")

Save the lab13.js file and upload it to the server. Check your work by clicking the Load local JSON button.

Part 3. Working with the Flickr API

An application programming interface (API) is a set of subroutine definitions, protocols, and tools for building application software. (from Wikipedia). Essentially, API is a set of rules on how to access and use a system's functionality. For example, you may be writing a program A that needs to talk to program B. Program B will have some rules that make such communication possible. These rules are called an API.

Companies like Flickr (and Twitter, and Google, and Amazon, ant lots of others) provide APIs so that developers like you and I can tap into their services and data while building our own applications, mashups and whatnots.

From flickr.com: "With over 5 billion photos (many with valuable metadata such as tags, geolocation, and Exif data), the Flickr community creates wonderfully rich data. The Flickr API is how you can access that data. In fact, almost all the functionality that runs flickr.com is available through the API. And the API is completely free to use, as a service to our members as well as developers and other integrators, so they can create even more ways to interact with photos beyond flickr.com." (https://www.flickr.com/services/developer/)

Learning how to use the Flickr API (or any API for that matter) is beyond the scope of our class (If you are curious, here's a good place to start: https://www.flickr.com/services/api/). However, that doesn't mean we can't try it out!

Here's how this works:

1. You apply for an API key (usually takes a minute). The key is then included in each method call you make using the API (it's like an extra argument in a function call)

2. You read the API specification (in our case, this: https://www.flickr.com/services/api/)

3. You construct a URL that includes a query string:
   http://sergey.cs.uni.edu/somefile**?parameter1=value1&parameter2=value2&parameter3=value3**
   ? marks the beginning of the query string
   parameter1=value1 is a key/value pair
   & separates key value pairs

4. You build your query string following the API specs

5. You use the constructed URL to send a request to the server (in our case, that would be the url we use in our AJAX call

6. You use the retrieved data the same way use did in Part! Of course, the structure of the data will be different and will depend on what API you use and what method you called.

So, last step in today's lab:

- Go to https://www.flickr.com/services/ and get an API key (choose non-commercial of course)
- Uncomment PART 3. in your lab13.js file.
- Copy and paste your new API key into the fcode (find var key = "YOUR-API-KEY-GOES-HERE";)

We'll discuss the code that follows in class. Essentially, we build up a url, then make the AJAX call, and then process the data. The data comes in a format that can be understood through (a) reading the API specs; or (b) looking at the data and comparing it to the URL of any photo on the Flickr website.Finally, we iterate over the retrieved data and construct an HTML string that we will then load into our div.

Save your file and upload it to the server. Now your page displays the 20 most recent photos postyed to Flickr!

Important note. Your API is private and should be protected. You CANNOT protect it if you use it in your JavaScript code. The right way to do it is to use it in a server-side program and call that program with your JavaScript AJAX call. This will be covered in CS 2100 and CS 3110. For now, get your code running, enjoy it, and then take it offline as soon as you are done.