**Lab Assignment 5: APIs, external JavaScript libraries**

GUS 8068, GUS/ES 4000: Web Mapping

Spring 2016

**DUE via Blackboard Fri, April 15th, 12 a.m.**

Purpose of the lab: To learn how to access APIs and external JavaScript libraries, and to thus extend the skills we’ve been learning with Leaflet.

**1. Walkthrough**

Go through these two tutorials:

1. How to Use APIs with JavaScript. <http://bit.ly/201a4X8>

2. Showing Instagram photos and videos on a Leaflet map. <http://bit.ly/1TwWEkU>

3. Look at the options available for the plugin: <https://github.com/turban/Leaflet.Instagram>

4. Load the following web page, as you’ll need it to complete the assignment: http://jelled.com/instagram/access-token

The first tutorial is crucial for understanding how to use JavaScript for API access. Here are some very important things to note as you go through this tutorial:

1. In the Codecademy site, you see the results by using “console.log”, but you can just as easily use the “alert()” function or assign the results to a div object by using “document.getElementById” as in previous labs.
2. Sometimes APIs return XML, and sometimes JSON (GeoJSON is a structured JSON file). In Web Mapping we’ll be working almost entirely with JSON.

Try experimenting with the XMLHttpRequest functions by loading a GeoJSON file that I’ve uploaded to my Github site: <http://burnsr77.github.io/GH_demo/PACo.geojson>. After you request the URL, your XMLHttpRequest object will have a “responseText” assigned to it. That is, if you have initiatlized your XMLHttpRequest like:

var xhr = new XMLHttpRequest();

then you can access the responseText with:

var APIresponse = xhr.responseText;

“responseText” is an attribute of xhr, and we have assigned it to a new variable, APIresponse. We can access that result by placing it in an alert like, “alert(APIresponse)” or by putting it in a div object like, “document.getElementById(‘yourDIVnamehere’).innerHTML = APIresponse;”. Any data that is on the web, you can scrape using this method.

When your XMLHttpRequest object receives the response from the server, it is stored as a string data type. Since JSON files are highly structured, you can also *parse* them in order to convert them to actual JSON dictionaries rather than strings. By calling the following method:

JSON.parse(xhr.responseText);

You convert a GeoJSON *string*, which is very limited, to a GeoJSON *dictionary*, which then allows you to process in a much more efficient way. For example, with my GeoJSON file listed above, you can access the first PA county’s name by calling the following attribute:

myJsonDictionary.features[0].properties.COUNTY\_NAM;

You can also upload your own GeoJSON file, and access it using XMLHttpRequest. If you would like to try this, you can isolate your GeoJSON files from previous labs, cutting-and-pasting them into a new .js file on Github, and then calling them with the above methods.

In the **second** tutorial, you learn a bit about displaying Instagram images on a Leaflet map. In order to do this, you need to download the “reqwest” library (linked in the sites above) and store it in the same folder as your website HTML file. You can then embed it in your HTML the same way you link to the Leaflet library, only you will be linking to your local folder. If, for example, in the same folder as your website you have the reqwest and Leaflet.Instagram libraries, in the <body> of your HTML you will need to put:

<script src="reqwest.min.js"></script>

<script src="Leaflet.Instagram/dist/Leaflet.Instagram.js"></script>

To use the Leaflet.Instagram library, you will need to have your own Instagram access token. To get one, first register a new client on the Instagram API site (<https://www.instagram.com/developer/clients/manage/>). You can put anything for your Website URL, and for the Redirect URL you will need to put “http://localhost”. With your Client ID, go to the site listed in #4 above, and generate an access token. Note that it will appear only in the URL bar.

Experiment a bit with displaying Instagram photos on a Leaflet map by using the following code:

L.instagram(“https://api.instagram.com/v1/media/search?lat=40.0237&lng=-75.1723&access\_token=[ACCESS\_TOKEN]”).addTo(map);

And make sure you replace the square brackets and ACCESS\_TOKEN with your access token. You can modify those lat/long values to change which data shows up. The default search distance is 5km.

**2. Other scraping resources**

When it comes to your final group projects, you may find lots of other scraping tools and APIs useful. I recommend sniffing around to find one that suits your needs, but you can certainly also check out this GUI-based interface for scraping: <https://www.import.io/>

**3. Assignment**

Using the tutorials above, create an Instagram overlay in a Leaflet map. For 5 extra points, display a GeoJSON file that you’ve created in the same map. You can choose anywhere in the world, and any narrative structure you wish. Upload this map and a lab report to the Lab 5 folder on Github.

**4. Steps**

In terms of practical steps to complete this lab, I am making some suggestions below. Depending on how you want to take the assignment, you may find that some or most of these steps do not apply in your case.

1. Go through the tutorials.
2. Create your own folder on Github into which you will place your code.
3. Download the Leaflet.Instagram and reqwest libraries to your folder on Github.
4. Create an Instagram access token.
5. Use this access token and the libraries above to overlay data on a Leaflet map.

**5. Submission**

Submit your map *and* your lab report in your own folder within Lab 5 on Github.