Dave Strohmeier

Intro to Web Mapping

Final Project

5-4-2018

Working for an archaeological firm for the past 15 years has shown me how much data can be accumulated during the course of an archaeological project let alone a few hundred projects in a single year. Most of the data created or collected is spatial in nature or associated to some spatial element such as a test pit, trench or excavation unit. For decades, if not centuries, archaeology has been synonymous with map making in that site plans are always being drawn and site locations carefully recorded. Within the last 20 years GIS has greatly filled a need for archaeologists to more accurately map archaeological features and resources and more quickly conduct spatial analyses for better site interpretation. With all these technological innovations most archaeologists, however; are still hindered by the use of static maps created for report production or fieldwork to review past projects or phases of projects. Review of these maps involve huge printing costs or cumbersome scrolling back and forth between PDFs on a computer to get the entire picture.

For my final project, my goal is to use the power of web mapping through html, css and javascript to create an HTML console by which archaeologists at the company I work for may use to review the data collected and created for past projects.

Though this will continue as an on-going project, for the sake of time, I decided to limit myself to 10 completed projects the company has conducted in the state of New Jersey in the past year. The first step was to normalize all the spatial data (shapefiles) used for these projects so that they all contained the same fields and attributes respective of the data type and individual feature so that they may be merged by data type. I also needed to add a field to each that would act as a key specific to the project so that each feature of a project could be called independently from other project. For this I used the 7-digit company job code. Once the data was normalized I merged the specific data types together in ArcGIS. They consisted of Areas of Potential Effects or Project Location (Polygon), Shovel Test Pits (Point), Excavation Units (Polygon) and Features (Polygon).

Each of these four shapefiles was converted to geoJson using Mapshaper to be called by my HTML document.

The first thing to do was create the map so a <div> tag was created for it to fit into that referenced the style id called ‘map.’ I wanted to make the map utilize the left half of the page and use the right half for some description, list of available project codes and a functional button. I set the width to 48% and height to 900px to leave some room at the top for a header, set position to fixed and had it float left for any tags that might come after it.

A variable was created for the map called map and using the leaflet method of L.map() calling to the id created for map and another variable called options that set the options for the map to be centered over the state of New Jersey at a zoom level to view the entire state. For this project I wanted the ability to use a variety of basemaps that would be useful in the context of these archaeological projects. A common basemap in used in the production of static maps for the company is the 2015 aerial tile layer provided in the web mapping service by the New Jersey Geographical Information Network. In order to use this basemap I needed to use an extension of the L.tileLayer() method L,tileLayer.wms(). This allows you to call tile layers from web mapping services with a call to the url and providing the name of specific layer as an option. I also wanted to be able to zoom in as far as possible so set the max zoom to 22. Other useful layers are the esri world imagery and world street map using the standard call to the leaflet basemap API, but also for some historical context to the data I wanted to pull a 1930 Aerial Imagery layer and 1977 Tidelands Aerial Imagery layer from the NJGIN web mapping service. Each base map was assighned a variable. I was able to add all of these base maps as options by using the leaflet L.control.layers() method and creating a variable of the list of basemaps and add that variable as an object in the method. Now the user could click through a series of five separate basemap under which to view the data.

The next thing to do was create a variable for each object to be added to the map. These objects would come from data provided by the geojson files. First, within each geojson a variable had to be created to refer to all the data in the file. A search call to the file was written within a script tag for each geojson allowing the variable within each file to be referred to throughout the HTML document.

Using the leaflet L.geoJson() method variables were created for each file by calling the variable within each file and converting them to objects that could be added to the map- Excavation Units, Features, STPs and APEs. onEachFeature options were added to the geoJsons that allowed popups to added to the layers that would summon desired attributes of the object using feature.properties.<attribute>.

I didn’t want my map, though, to just bring in all the objects at once and have the user have to navigate to the desired project location. I added a Button that through an event listener instantiated a prompt that asks the user to enter a project number. The input is then stored as a variable “proNum” within the event listener along with all the other geoJson variables. Within each geoJson variable a filter option was added that stated if “feature.properties.project”== “proNum”, return the feature. I also wanted the map to zoom to the desired project location and so added the combo method of map.fitBounds(ape.getBounds()) at end of the area of potential effects variable. Now when the user enters the input code the map zooms to that particular layer.

I also wanted to add different symbology for the varying result attributes for the shovel test pit layer. To do this I created icons for each of the possible results and set them to variables of appropriate size using the L.icon() method. I tried to use a system of if else statements to return the appropriate symbol for each result, but ran into conflict with “proNum” variable and so ended up creating a geoJson variable for every possible result. This is something that I believe can be worked out under a single variable but decided to go with what worked for the sake of time.

The style was rendered by using the setStyle() method on each geoJson variable except for the STP layer which only used the options of their icons. I wanted to add these features to the control layer so created a layer group for each type of feature and a variable for all the layer groups which was added to the same control layer as the basemaps.

To top the map off, a full screen option was added by using a mapbox plugin that required a call to the map box API and setting fullscreenControl to true in the map options. One problem I noticed with this full screen was that my collapsed control layer was not visible in his view setting, perhaps something that might be corrected if a leaflet full screen plugin could be found.

Finally, a background image was added that was created in Adobe illustrator that incorporated an archaeology photo blended with a gradient of the company color scheme. An unordered list was made of the available projects codes and carefully (I thought) positioned to align right of the map with all elements evenly distributed. During the final presentation it was noted that the elements were arranged differently on a different device with the map covering the button negating the functionality of the map. This will have to be corrected and hopefully done so before final submission.

Over the course of the semester, I faced many challenges. A lot of them involved functions and where certain variables can be placed either inside or outside the function for other parts of the script to recognize them. Once the concept of being able to pile more than just the initial action of the function or method inside was realized, the possibilities opened up. Such as in lab 4, at the start, it never occurred to me that the majority of the document’s script lived inside the button. Naturally, for whatever reason, I would tend to think that the script for a button is independent of the rest of the document. Realizing that the event listener is parent to all functionality that follows it was an important step in figuring out how a workable document should be structured.

I also struggled and still am struggling with html/css positioning of elements, especially on different computers or devices. Though it wasn’t really a required task in the course I was always thinking about how to change the layout of the assignment templates. For the final project presentation, it’s quite the letdown to find that all the work into tweaking the position of all your elements is for naught when opened on a device that’s not the one on which the document was developed.

My greatest success was not in what I did, per se, but in realizing what I didn’t do. Trying so many different ways and approaches to get a map to function properly you will be left with a lot of junk that doesn’t work. When being bold you may end up with a lot of code that you need to get rid of when you find that it fails. Good housekeeping of your document can save you lots of time in the long run. If you miss one little piece it can cause your script to malfunction, thinking something doesn’t work when it actually does. Unnecessary setbacks. I found that once you have a part working to save the document as something new when you proceed with a new idea, that way you can easily return to a document that works without spending time figuring out which bracket needs to be added or omitted.

I can imagine building web maps like these quickly and easily to great fanfare. It seems like most people want to see relatively easy concepts displayed in an interactive way. Making a simple web map can provide exactly what people are looking for and knowing how to stylize them and add fun widgets are a bonus. Knowing how to deal with tougher statistical data and concepts seems like a fun challenge. As a fan of puzzles I could see myself, professionally, trying to interpret other people’s ideas into functional web maps.

If there were a web mapping 2 class, I of course, would love to learn about the use of other common functions in mapping with javascript. I would also like to learn about more plugins in and out of the leaflet API. Perhaps even some jquery plugins that don’t necessarily have to do with web mapping, but could easily be incorporated. I feel like learning some more basics of html/css and how to layout elements of a web page would be helpful. It is for sure a growing field, and with every added plugin and capability more closely rivals the leading industry software with the added bonus of interactivity.