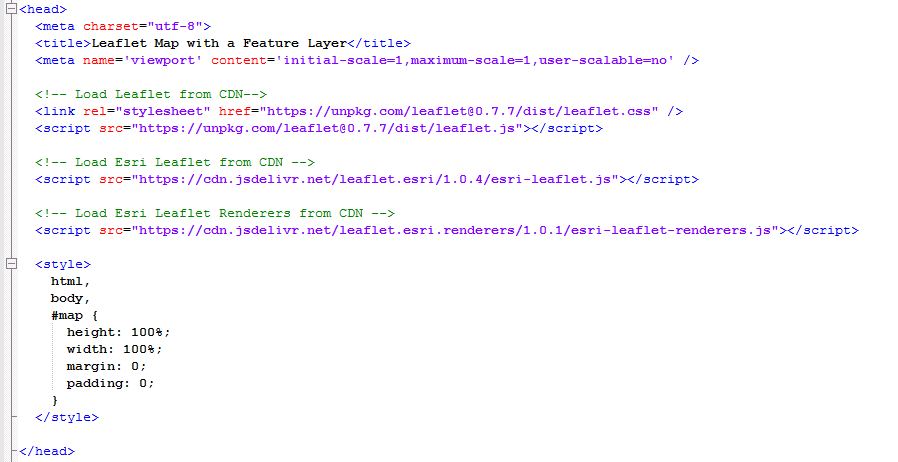
**Introduction**

The goal of this final project was to create an interactive html page from Javascript (JS) code. This interactive webpage features the watersheds of North Carolina and the user would be allowed to click on the watershed of their particular needs and see the different stream segments contained within that basin. Each stream segment would be able to be clicked upon and a popup is provided with general data concerning its water source and recent precipitation. That is the general design of this project and because of the guess and check methods required for using html, some components of the map that were attempted were not successfully implemented. This workflow diagram will walk through the general steps taken to successfully create this interactive map through Leaflet, as well as outline particular difficulties experienced while trying to create a better product.

**Leaflet Library**

Leaflet was the primary tool used in our JS code to make this interactive map. It is a particular package that allows html pages to be coded to provide different geojson’s as interactive, user-friendly maps. The first step in creating this product was to create an html JS file, and then to bring in the Leaflet library. (see top of Figure 1). After bringing in the Leaflet link, this library can provide several functions that allow you to modify your webpage. This interactive map of the watersheds of NC can have several different interface tools added to it such as drop-down menus, maps with user-specified parameters, and D3 graphs incorporated into the popups.

**Figure 1**: Opening lines of code to bring in the Leaflet Library to your JS code and the latter half of code sets up the initial size of your webpage.



**Webpage Setup**

After setting the initial webpage size and length (see bottom of Figure 1), our first step in making sure our html page and Leaflet map were working was to incorporate the feature servers showing our watershed, stream, and gage data. The code was eventually adjusted correctly from our Github files from Advanced GIS to display our 3 different feature servers on our interactive map. Each feature server input required (see Figure 2)

Figure 2: Shows the code getting one of the feature server’s as an input to our interactive html map



The next step in adding user-friendliness to our Leaflet map was to incorporate a drop-down menu of the available watersheds in NC. Actually including the JS code that calls upon each different webpage of that particular webpage became overwhelming. A drop-down menu that is not fully interactive but still displays all of the available watersheds on our interactive map was attempted instead. That code includes calling upon our watershed feature layer and listing each basin name successively within that drop-down menu but the code does not work (see Figure 3). This portion of the project was a little out of the scope for this particular project but should be pursued in the future for more experienced JS users.

Figure 3: This portion of code demonstrates the attempt at adding an interactive drop-down menu for our users to pick a certain watershed that the interactive map would then zoom to.



After the drop-down menu was attempted, our watershed feature names were labeled but the Neuse Watershed’s label in particular was not centered on the basin correctly so an alternate labeling route was taken using the hover function. If the user hovered over a particular watershed, the basin’s name would pop up by the watershed. This method worked and helps the user to see a clear boundary and name of the watershed, helping to eliminate our earlier labeling problem.

Another add-on was applied to the feature server of water gage sites within the Neuse Watershed. The popup function in html was relatively easy to configure to our feature server and this allowed the user to click each specific site and know its description and county. Our JS code also attempted to incorporate D3 graphs into the water gage pop up. ------

Finally, after our product had a great deal of user interactivity and interface functionality, text was added to the webpage for a title, description, and tips on how best to use this interactive map. These map tips would help instruct the user on how to generally manipulate the interactive map to their specific project. They would also provide information concerning the data used in this interactive map so the user could potentially manipulate the code to suit their needs even further.

**GITHUB**

Throughout the coding process, a GitHub account was created to allow us to track our changes and make several different maps. Different add-ons were attempted as entirely different maps. This coding process is shown in the GitHub Table of Contents. We first created a Neuse Watershed map only that allowed minimal user interactivity but still displayed the color-coded streams and stream gage pop-ups.

The next stage in our GitHub process was to attempt adding a dropdown menu of our watersheds for the user to click on, then the map to immediately zoom to that specific watershed. As mentioned in the Problems section, the dropdown had to be simplified in its concept due to time constraints and it did not produce a successful result.

The next map produced was the first NC watershed’s extent map that attempted to produce labels on each one of the watershed polygons. As outlined before, this labeling method did not look very smooth and so another map was produced on our GitHub account to attempt to remedy this stylistic error.

The final map produced added the hover option to eliminate the labeling problem and is stylistically our most advanced interactive map. This map excluded the attempted drop-down menu but included more text and a cleaner interface for the user to navigate.

**Mistakes, Problems, and some Solutions**

Some of these features are very hard to produce, however, and could not be successfully implemented. One problem we had with our hover labeling system is that the watershed stays highlighted after the user hovers over it, eventually resulting in all of the watersheds becoming highlighted. This just requires you to reload the page but the problem could not be remedied in our JS code.

As mentioned above, the drop-down menu is a great tool to implement in an interactive map but ours would not successfully drop down due an error in the JS coding. The button shows and it highlights after you click it but an actual drop-down menu of the NC watersheds is not produced. Maybe each watershed must be linked to its actual data for it to actually show up in the drop-down.

Dojo is a different tool library for JS code and the acquired Dojo code from online could not be implemented into our leaflet map script. It seems relatively easy but because it was a separate library and one we were not familiar with, it could not be properly woven into our Javascript code. Because of this finding, Dojo was not implemented in our final Leaflet product but it could serve as an easy fix for several styling and functionality issues when creating an interactive Leaflet map.

Initially, every time we tried to get each feature to show up on our interactive map, it took a long time just for each feature to load. It is recommended that instead of using Internet Explorer (our default browser) you open the html specifically in Google Chrome. This helped the loading time of each attempted html page by a great deal.