South King County Food Coalition Google Maps Specifications (map for individual food bank)

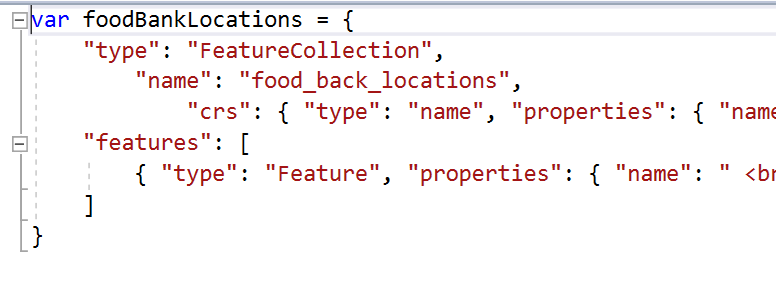
GEOG 469 David Lee 4/25/2019

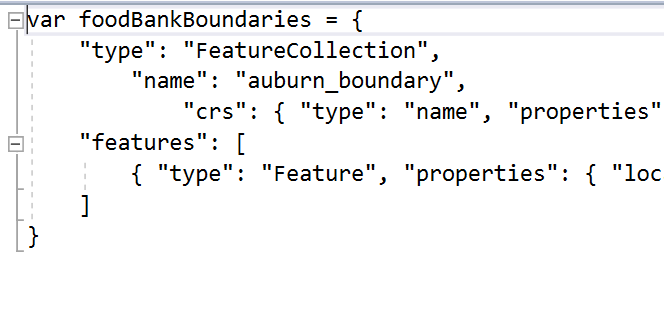
1. Preparing necessary shapefiles

The food bank location should be in one shapefile as point data. The food bank boundary should be saved into its own shapefile (ensure that the boundaries are accurate). Once both shapefiles are correctly made, user a conversion tool such as an online website to convert from SHP file to GeoJSON file. GeoJSON is the format that will be used within the JS file when creating the html page. The food bank locations should be formatted similar to this:



This file can be then turned into a JS file by saving as a new file with the .js extension. The boundary GeoJSON file should look similar and the file can be turned into a JS file the same way.

Once the JS file is created, you must initiate the food bank point data into a variable by adding ‘var foodbankName’. Also, similarly for the boundary.

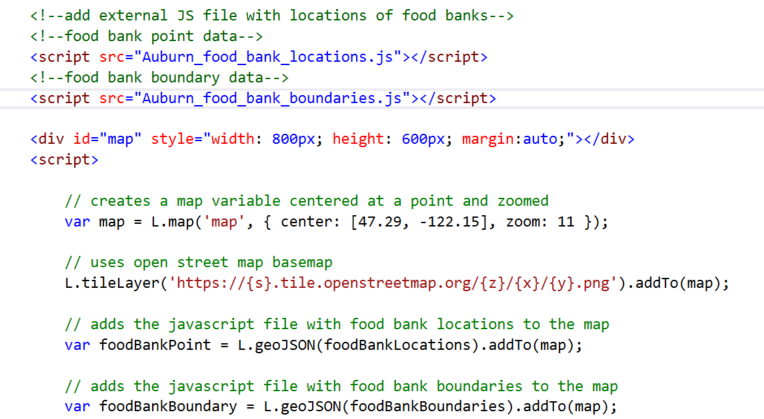


Setting the data to a variable will allow the html code to access that data and to use it.

1. Creating the html page



This is the basic mapping shell. The code that is already written here creates a map object using the Google API and the Leaflet plugin. The tile layer can be changed to choose a different basemap.



Once the JS files are connected within the <script src> tag, A new variable called foodbankPoint can be created and the food bank point variable that was made within the actual JS file can be called.

‘var foodBankPoint = L.geoJSON(foodBankLocations).addTo(map);’

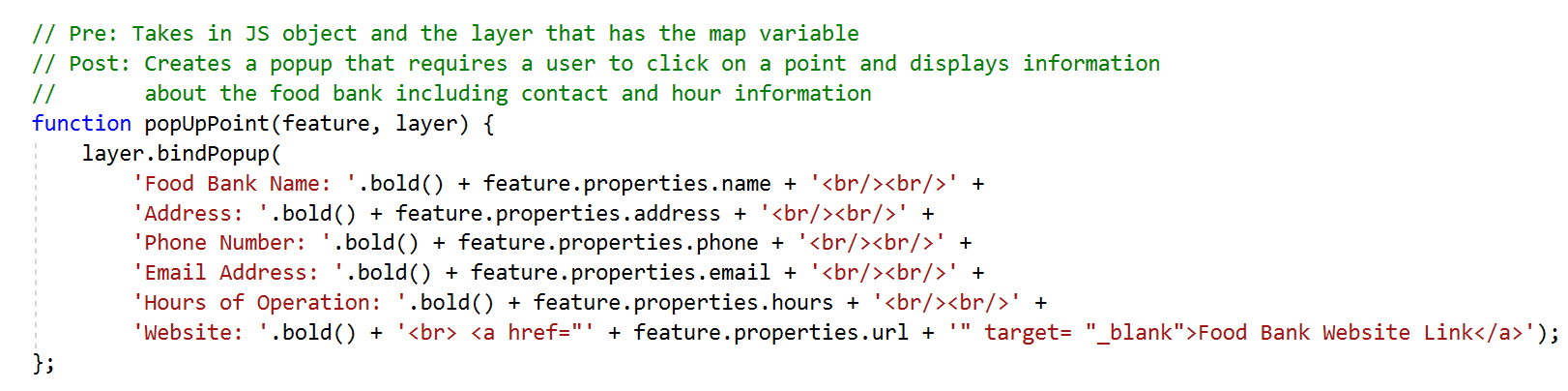
This will add the point to the map. Similar method for adding the boundary.

1. Adding description and interaction to the food bank point

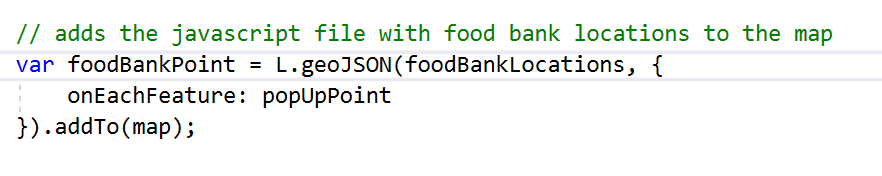
This point displayed on the map by itself is not very informative. Information about the actual food bank location was manually added as properties within the JS file for that food bank. As shown below:



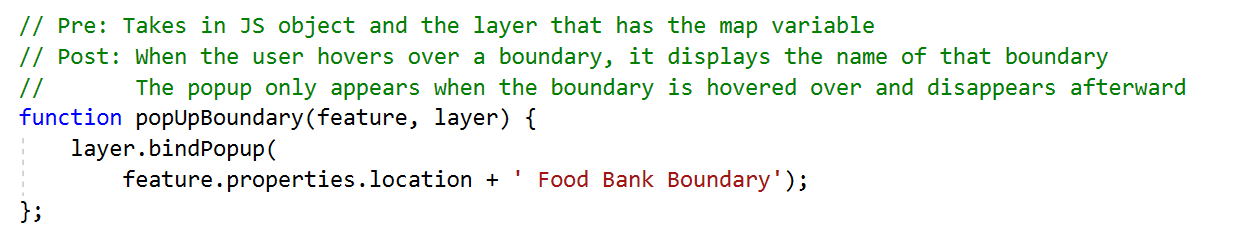
Adding these attributes will allow for the html page to get these elements to display on the map. Each property is identified using a variable name such as ‘name’ or ‘address’. These names will be used to call the variable. This code creates a popup that will appear when users click on a food bank point. Each variable in the JS file is called with ‘feature.properties.’.



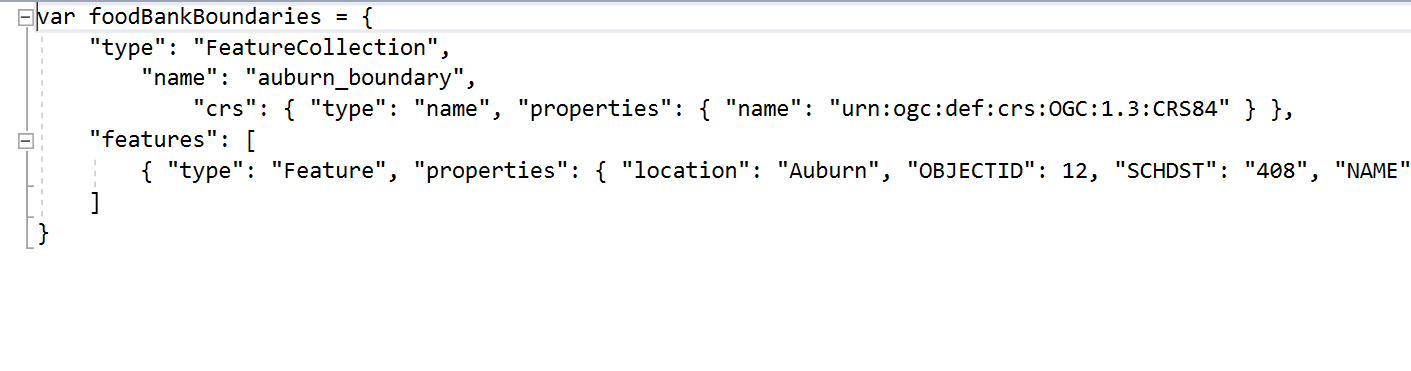
After creating this function, it must be connected to our foodBankPoint object to actually use it. This we add the follow code to our variable initiation:



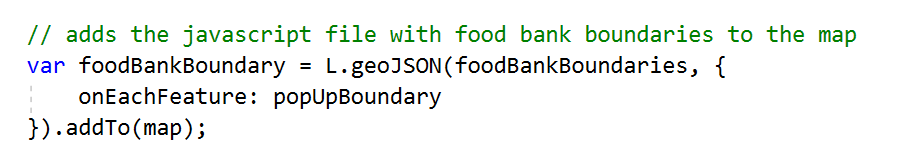
As a result, on each feature click, the popup we created will open.



We can also create a similar function for our boundary which only displays the boundary name. The name is also added as a property in the boundary JS file.



Since the function was created, it must also be connected to our foodBankBoundary variable.



Since the food bank boundary name can be shown when click on, it would be ideal to have that selected boundary display as a different color to identify it. 2 functions were created to display the color that is shown: one shows the default before the user clicks on a boundary and the other function shows a color after an event occurs (event meaning a click or hover). Thus those two functions need to be hooked up to the boundary function based on ‘mouseover’ (hover).

