Mapping API's: Leaflet - Content Styled Circle Markers

Welcome to the Essential ArcGIS Task Sheet Series. This series supplements the Iowa State University GIS Geospatial Technology Training Program short course series. The task sheets are designed to provide quick, easy instructions for performing mapping tasks.

This task sheet demonstrates how to display a GeoJSON data set in Leaflet as standard circle markers and as circle markers that can be styled based on the data value. This task sheet builds off of the *Mapping API's: Leaflet - Circles and Circle Markers* **PM2082-15b**. The code for this task sheet (**contentStyledMarkersLeaflet.html**) and the previous task sheets can be found on the ISU Geospatial Technology Program GitHub page at https://github.com/ISUEOGTP/GISTaskSheets.

1. Introduction

a. First, you will need a basic leaflet map setup. Reference the task sheet: *Mapping API's: Leaflet - Getting Started* **PM2082-14r** to learn how to get this set up, or get the set up code from our GitHub page at https://github.com/ISUEOGTP/GISTaskSheets/blob/master/Leaflet-Tutorials/helloLeaflet.htm.

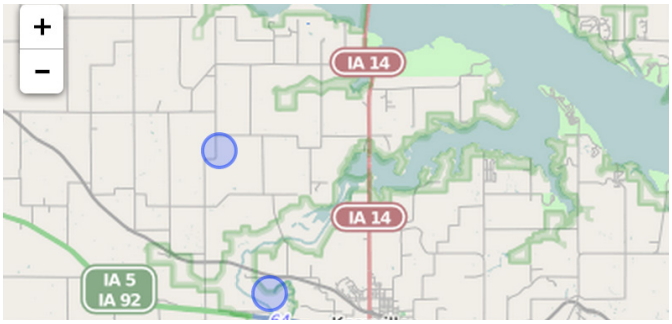
b. A sample of the GeoJSON data used in this task sheet can be seen to the right, for the full data set, copy the data from **contentStyledMarkersLeaflet.html** on GitHub.

c. In this demonstration, you will use a simple dataset in the GeoJSON format to display circle marker. This data includes three features that have two properties each – **specialLabel** and **specialValue**. These properties will be used to style the markers in **step 5** and **step 6**.

```
//define data
myJSON={
    "type": "FeatureCollection"
    "features": [
        {
            "type": "Feature",
            "properties": {
                "specialValue":20,
                "specialLabel": "rock"
            },
            "geometry": {
                "type": "Point"
                "coordinates": [
                        -93.1,
                        41.3
                ]
            }
        },
        {
            "type": "Feature",
            "properties": {
                "specialValue": 30,
                "specialLabel": "tree"
            },
            "geometry": {
                "type": "Point",
                "coordinates": [
```

2. Adding Data

a. First, create a variable called **myJSON** at the top of the JavaScript code and set it equal to the GeoJSON data. *Note: typically it is best practice to have data in an external file, but for the purpose of testing it is sometimes advantageous to start with a subset of the data. This allows you to modify the values and see the impact on the map.*



3. Standard Points

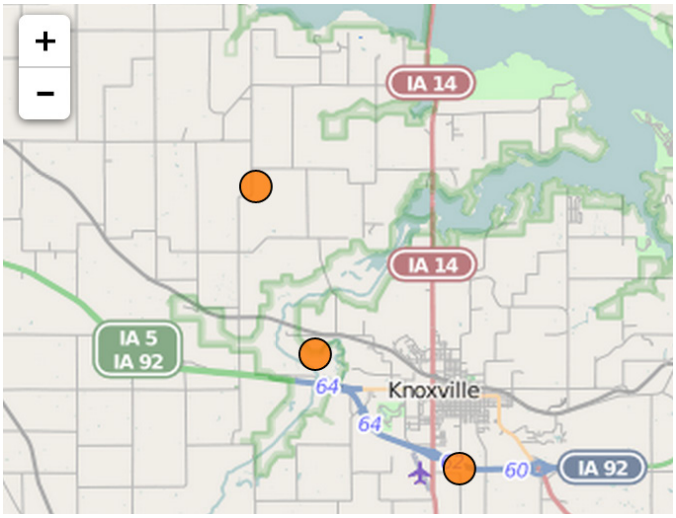
a. Once the GeoJSON data is added, the basic map constructor can be written. This is followed by the code necessary to read the data and display it as markers on the map. Use the code to the right to display the default circle markers.

```
//load data
L.geoJson(myJSON, {
    pointToLayer: function (feature,
latlng) {
        return L.circleMarker(latlng,{});
    }
}).addTo(map);
```

4. Marker Options

a. To change the marker styling from the default, add the marker options as shown below after **return L.circleMarker(latlng,{** . The markers will now be a transparent orange with a black outline.

```
return L.circleMarker(latlng,{
  radius:8, //expressed in pixels
  fillColor: "#ff7800",
  color: "#000", //black outline
  weight: 1, //outline width
  opacity: 1, //line opacity
  fillOpacity: 0.8
});
```



5. Custom Fill Color Based on Data Value

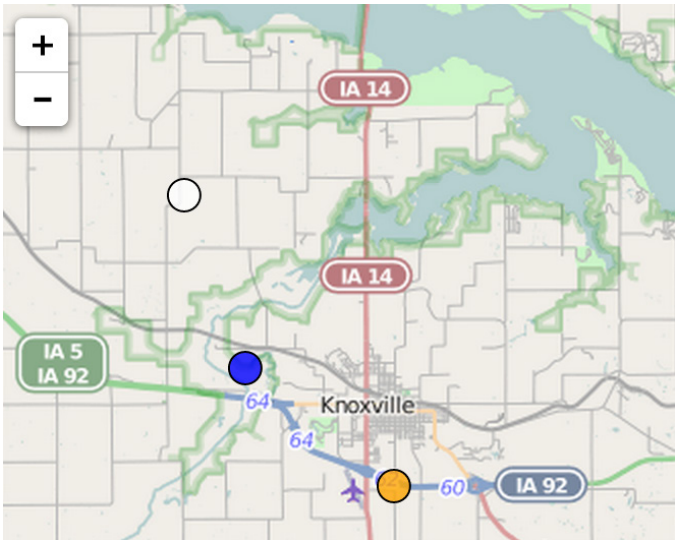
a. To customize the fill color based on a data property value, change the **fillColor** option to call a function defined as **getColor**. *Note that the function is called getColor and is passed the feature value of the specialLabel property.*

```
fillColor: getColor(feature.properties.specialLabel),
```

b. The getColor function needs to be written before the data is loaded. Place the **getColor** function shown below, between the map constructor and where the data is added to the map.

```
//set color of marker based on feature value
function getColor(d) {
    return   d == 'rock' ? 'orange' :
             d == 'tree' ? "#0000ff" : //blue
             "#FFFFFF"; //white
}
```

c. The **getColor** function receives the **specialValue** property (in this case **rock** or **tree**) and names it **d**. Then it tests to see if d is equal (**==**) to a string value. If it matches, the defined color is returned. If the string does not match, it tests the next value for a match. This process continues until there is a match. If there are no matches, a default color is used. *Note: colors can be provided as hexadecimal or as a color name. For possible color names see: http://www.w3schools.com/html/html_colornames.asp.*



6. Other Options

a. If desired the radius of the circle (in pixels), opacity, line color and line weight could also be based on a data value. The following example would set the radius size of the circle.

```
radius: feature.properties.specialValue,
```

Contact:

Bailey Hanson bahanson@iastate.edu, 515-520-1436 or Professor Christopher J. Seeger, ASLA, GISP cjseeger@iastate.edu, 515-509-0651 for more information about the Geospatial Technology Program. This task sheet and more are available at www.extension.iastate.edu/communities/gis