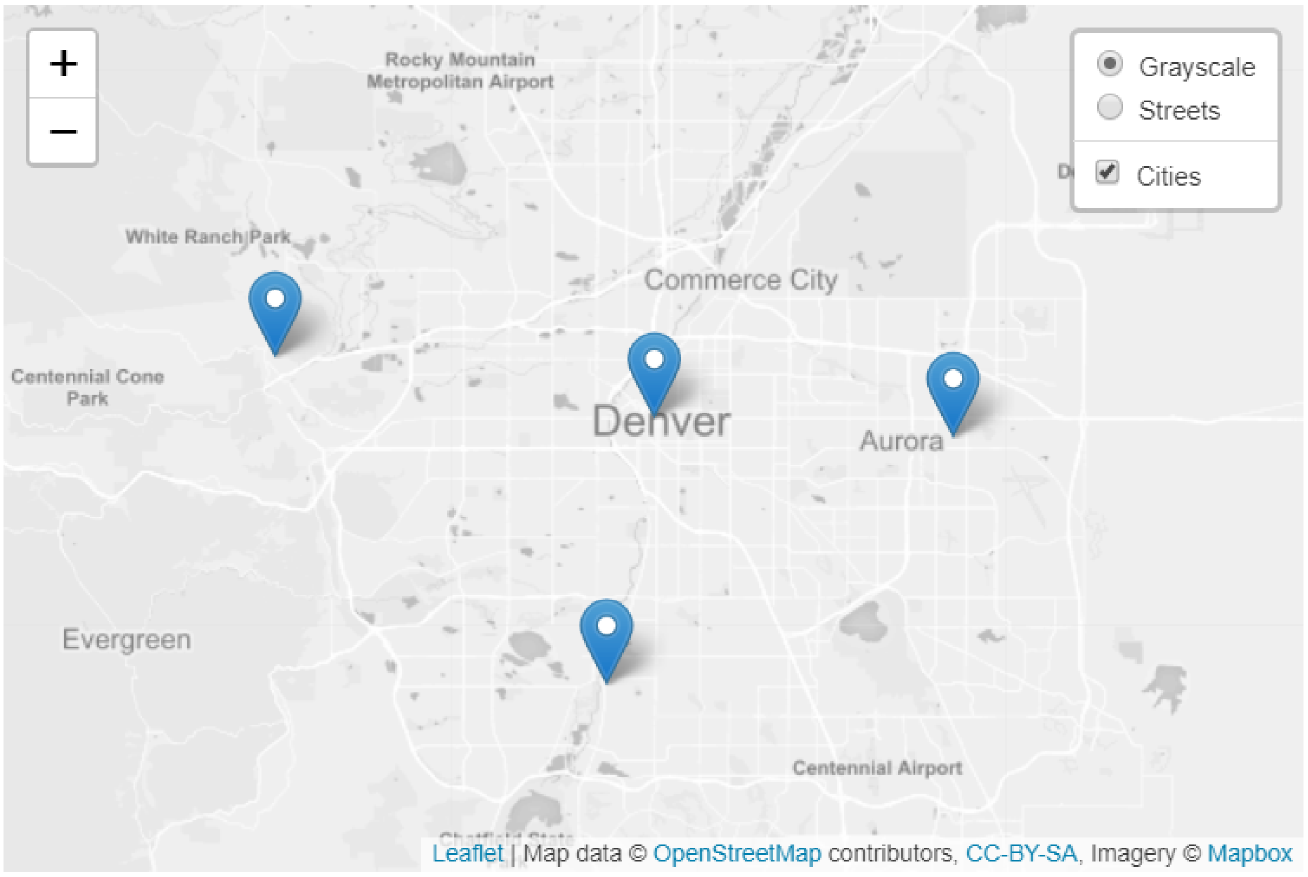
04. LEAFLET Layer Groups & Control.

4.1. Markers with Custom Icons.

This tutorial will show you how to group several layers into one, and how to use the layers control to allow users to easily switch different layers on your map.



4.2. Layer Groups

Let's suppose you have a bunch of layers you want to combine into a group to handle them as one in your code:

```
var littleton = L.marker([39.61, -105.02]).bindPopup('Littleton, CO.'),
    denver    = L.marker([39.74, -104.99]).bindPopup('Denver, CO.'),
    aurora    = L.marker([39.73, -104.8]).bindPopup('Aurora, CO.'),
    golden    = L.marker([39.77, -105.23]).bindPopup('Golden, CO.');
```

Instead of adding them directly to the map, you can do the following, using the LayerGroup class:

```
var cities = L.layerGroup([littleton, denver, aurora, golden]);
```

Easy enough! Now you have a cities layer that combines your city markers into one layer you can add or remove from the map at once.

4.3. Layers Control

Leaflet has a nice little control that allows your users to control which layers they see on your map. In addition to showing you how to use it, we'll also show you another handy use for layer groups.

There are two types of layers: (1) base layers that are mutually exclusive (only one can be visible on your map at a time), e.g. tile layers, and (2) overlays, which are all the other stuff you put over the base layers. In this example, we want to have two base layers (a grayscale and a colored base map) to switch between, and an overlay to switch on and off: the city markers we created earlier.

Now let's create those base layers and add the default ones to the map:

```
var grayscale = L.tileLayer(mapboxUrl, {id: 'MapID', attribution:
mapboxAttribution}),
    streets   = L.tileLayer(mapboxUrl, {id: 'MapID', attribution:
mapboxAttribution});

var map = L.map('map', {
    center: [39.73, -104.99],
    zoom: 10,
    layers: [grayscale, cities]
});
```

Next, we'll create two objects. One will contain our base layers and one will contain our overlays. These are just simple objects with key/value pairs. The key sets the text for the layer in the control (e.g. "Streets"), while the corresponding value is a reference to the layer (e.g. streets).

```
var baseMaps = {
    "Grayscale": grayscale,
    "Streets": streets
};

var overlayMaps = {
    "Cities": cities
};
```

Now, all that's left to do is to create a Layers Control and add it to the map. The first argument passed when creating the layers control is the base layers object. The second argument is the overlays object. Both arguments are optional: you can pass just a base layers object by omitting the second argument, or just an overlays objects by passing null as the first argument. In each case, the omitted layer type will not appear for the user to select.

```
L.control.layers(baseMaps, overlayMaps).addTo(map);
```

Note that we added grayscale and cities layers to the map but didn't add streets. The layers control is smart enough to detect what layers we've already added and have corresponding checkboxes and radioboxes set.

Also note that when using multiple base layers, only one of them should be added to the map at instantiation, but all of them should be present in the base layers object when creating the layers control.

Finally, you can style the keys when you define the objects for the layers. For example, this code will make the label for the grayscale map gray:

Example:

```
var baseMaps = {
    "<span style='color: gray'>Grayscale</span>": grayscale,
    "Streets": streets
};
```