LEAFLETJS

Leaflet is the leading open-source JavaScript library for mobile-friendly interactive maps. Weighing just about 38 KB of JS, it has all the mapping features most developers ever need. Leaflet is designed with *simplicity*, *performance* and *usability* in mind. It works efficiently across all major desktop and mobile platforms, can be extended with lots of plugins, has a beautiful, easy to use and well-documented API and a simple, readable source code that is a joy to contribute to.

Leaflet in angular

1.Get Started:

a. Install Leaflet using npm:

```
npm i leaflet
```

b. In `angular.json` file, Add styles and scripts as follows:

```
"styles": [
            "node_modules/leaflet/dist/leaflet.css",
            "src/styles.css"
          ],
          "scripts": [
            "node_modules/leaflet/dist/leaflet.js"
          ]
```

c. Install mapbox for angular inorder to add a tile layer to add to our map, Creating a tile layer usually involves setting the URL template for the tile images, the attribution text and the maximum zoom level of the layer.

```
npm install --save mapbox
```

d. In order to use tiles from Mapbox, you must also request an access token. and save it in a variable 'accessToken'

2. Create a new component in angular for displaying map using 'ng g c map'

- In **map.component.ts**
  - Declare a variable L to be used by Leaflet, and also save your Access token.
  - Now Let's create a map of the center at a default location with Mapbox Streets tiles. First we'll initialize the map and set its view to our chosen geographical coordinates and a zoom level

```
    var mymap = L.map('mapid').setView([latDefault, lngDefault],
zoomDefault);
```

The setView function call also returns the map object — most Leaflet methods act like this when they don't return an explicit value, which allows convenient jQuery-like method chaining

Next we'll add a tile layer to add to our map, in this case it's a Mapbox Streets tile layer. Creating a tile layer usually involves setting the URL template for the tile images, the attribution text and the maximum zoom level of the layer. Here we'll use the mapbox.streets tiles from Mapbox's "Classic maps" (in order to use tiles from Mapbox, you must also require the access token).

```
L.tileLayer('https://api.tiles.mapbox.com/v4/{id}/{z}/{x}/{y}.png?acc
ess_token={accessToken}', {
    attribution: 'Map data &copy; <a
href="https://www.openstreetmap.org/">OpenStreetMap</a> contributors,
```

```
<a
href="https://creativecommons.org/licenses/by-sa/2.0/">CC-BY-SA</a>,
Imagery (c) <a href="https://www.mapbox.com/">Mapbox</a>',
    maxZoom: 18,
    id: 'mapbox.streets',
    accessToken: 'your.mapbox.access.token.here'
}).addTo(mymap);
```

map.component.ts:

```
import { Component, OnInit } from '@angular/core';

declare var L: any;
var apiToken = 'MAP_BOX_ACCESS_TOKEN_HERE'
@Component({
 selector: 'app-map',
 templateUrl: './map.component.html',
 styleUrls: ['./map.component.css']
})
export class MapComponent implements OnInit {

 constructor() { }

 ngOnInit() {
   var mymap = L.map('mapid').setView([76.505, 10.0229], 12);


L.tileLayer('https://api.tiles.mapbox.com/v4/{id}/{z}/{x}/{y}.png?access_token=
{accessToken}', {
   attribution: 'Map data &copy; <a
href="https://www.openstreetmap.org/">OpenStreetMap</a> contributors, <a
href="https://creativecommons.org/licenses/by-sa/2.0/">CC-BY-SA</a>, Imagery ©
<a href="https://www.mapbox.com/">Mapbox</a>',
   maxZoom: 18,
   id: 'mapbox.streets',
```

```
    accessToken: apiToken
    }).addTo(mymap);
  }
}
```

Now in our **map.component.html** put a `div` element with a certain id where you want your map to be placed.

map.component.html:

```
<div class="container">
    <div id="mapid"></div>
</div>
```

Add Styles to div element that contains our map

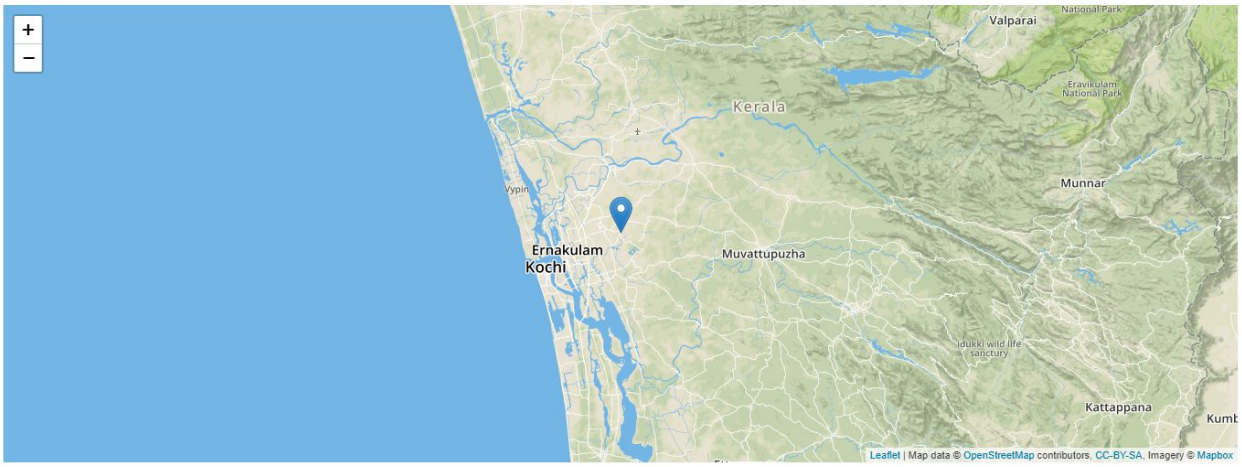
In **map.component.css**

```
#mapid { height: 500px; }
```

Adding Marker to map:

```
var marker = L.marker([51.5, -0.09]).addTo(mymap);
```

Example:



Adding Circle to map:

```
var circle = L.circle([51.508, -0.11], {
    color: 'red',
    fillColor: '#f03',
    fillOpacity: 0.5,
    radius: 500
}).addTo(mymap);
```

Adding Polygon:

```
var polygon = L.polygon([
    [51.509, -0.08],
    [51.503, -0.06],
    [51.51, -0.047]
]).addTo(mymap);
```

Attaching Popups:

Popups are usually used when you want to attach some information to a particular object on a map. Leaflet has a very handy shortcut for this function. To Attach popup to a marker

```
marker.bindPopup("<b>Hello world!</b><br>I am a popup.").openPopup();
//Will Attach popup to marker
circle.bindPopup("I am a circle."); //Will Attach popup to Circle
```

The `bindPopup` method attaches a popup with the specified HTML content to your marker so the popup appears when you click on the object, and the `openPopup` method (for markers only) immediately opens the attached popup.

Dealing with events:

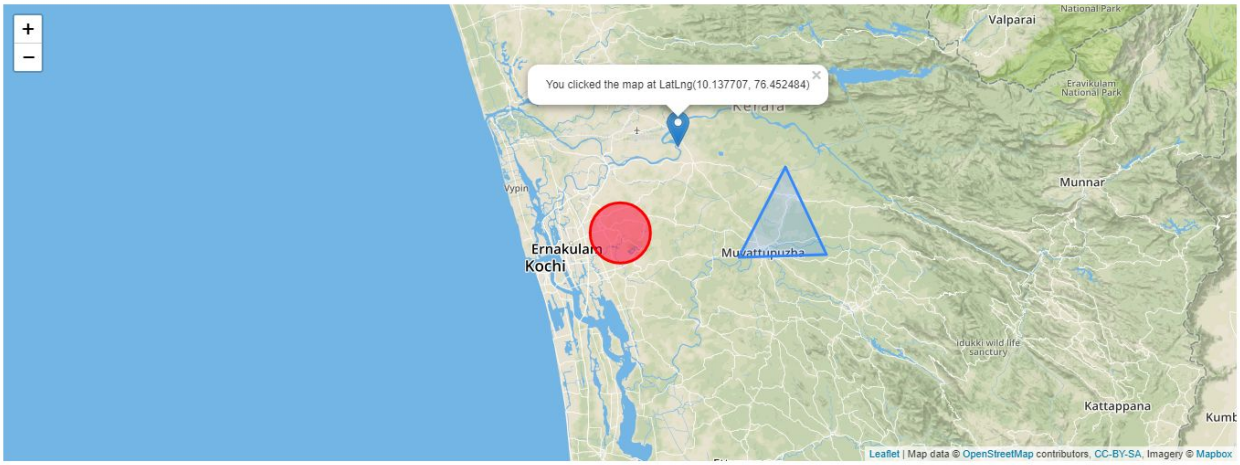
Every time something happens in Leaflet, e.g. user clicks on a marker or map zoom changes, the corresponding object sends an event which you can subscribe to with a function. It allows you to react to user interaction:

```
var popup = L.popup();

function onMapClick(e) {
    popup
        .setLatLng(e.latlng)
        .setContent("You clicked the map at " + e.latlng.toString())
        .openOn(mymap);
}

mymap.on('click', onMapClick);
```

Map Example:



Ref: https://leafletjs.com/examples/quick-start/

https://leafletjs.com/reference-1.4.0.html#url-template

Sample POC : https://github.com/manusree-korewireless/Angular-POCs