13.2.4 Create a Simple Map

Now that you have added stylesheet links to your HTML page and CSS file, Sadhana will help you add the last two files you'll need to create a simple map—the `config.js` file for your API token and the `logic.js` file for your JavaScript and Leaflet code.

To create a map, we'll need to add two final pieces to our Simple_Map folder:

- The `config.js` file, which will hold our Mapbox API key.
- The `logic.js` file, which will contain all the JavaScript and Leaflet code to create the map and add data to the map.

Add the config.js File

In our Simple_Map folder, add a new folder in the static folder called "js." In the js folder, create a new file and name it `config.js`. This file will hold our Mapbox API key.

REWIND

The `config.js` file is like the `config.py` file we used to hold the OpenWeatherMap API and Google API keys.

On the first line of the `config.js` file, add the following code:

```
// API key
const API_KEY = "";
```

Add your Mapbox API key between the quotations and save the file.

Add the logic.js File

Next, in the js folder, create a new file and name it, `logic.js`. Now we'll add some boilerplate code to the `logic.js` file. Most of this code can be reused for many of the maps we'll create later on in this module.

On the first line, add the following code:

```
// Add console.log to check to see if our code is working.
console.log("working");
```

The `console.log()` function with the phrase "working" inside the parentheses will help us confirm that our `logic.js` file is being accessed in the console on Chrome.

Add a Map Object

Next, we'll add the map object, as shown on the Leaflet Quick Start Guide page with some slight modifications. We'll change the geographical center of the map to the approximate geographical center of the United States.

```
// Create the map object with a center and zoom level.
let map = L.map('mapid').setView([40.7, -94.5], 4);
```

In the code block above:

1. We're assigning the variable map to the object `L.map()`, and we'll instantiate the object with the given string `'mapid'`.

2. The `mapid` will reference the `id` tag in our `<div>` element on the `index.html` file.
3. The `setView()` method sets the view of the map with a geographical center, where the first coordinate is latitude (40.7) and the second is longitude (-94.5). We set the zoom level of "4" on a scale 0–18.

An alternative to using the `setView()` method is to modify each attribute in the map object using the curly braces notation as follows:

```
// Create the map object with a center and zoom level.
let map = L.map("mapid", {
  center: [
    40.7, -94.5
  ],
  zoom: 4
});
```

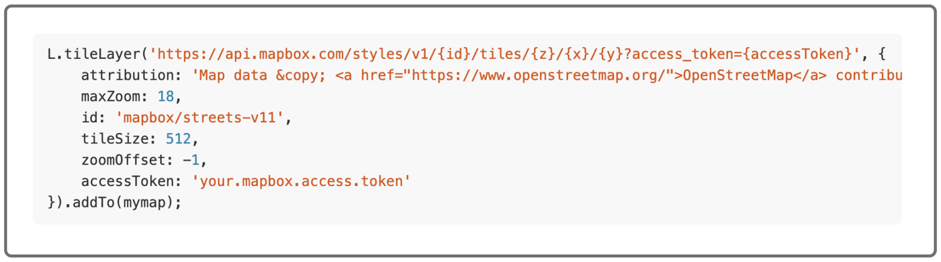
This method is useful when we need to add multiple tile layers, or a background image of our map(s), which we will do later in this module.

Add a Tile Layer for Our Map

Now, we will add the map tile layer method to the `logic.js` file. The tile layer is used to load and display a tile layer on the map. We have two options to create a tile layer.

Use the Leaflet Documentation

The **Leaflet Quick Start Guide** **(https://leafletjs.com/examples/quick-start/)** provides the `tileLayer()` code:



```
L.tileLayer('https://api.mapbox.com/styles/v1/{id}/tiles/{z}/{x}/{y}?access_token={accessToken}', {
    attribution: 'Map data &copy; <a href="https://www.openstreetmap.org/">OpenStreetMap</a> contribu
    maxZoom: 18,
    id: 'mapbox/streets-v11',
    tileSize: 512,
    zoomOffset: -1,
    accessToken: 'your.mapbox.access.token'
}).addTo(mymap);
```

We can copy this tile layer code and assign it to the `streets` variable, since the tile layer will create a street-level map. Add the following code block to your `logic.js` file:

```
// We create the tile layer that will be the background of our map.
let streets = L.tileLayer('https://api.mapbox.com/styles/v1/{id}/tiles/{z}/{
    attribution: 'Map data &copy; <a href="https://www.openstreetmap.org/">O
    maxZoom: 18,
    id: 'mapbox/streets-v11',
    tileSize: 512,
    zoomOffset: -1,
    accessToken: API_KEY
});
// Then we add our 'graymap' tile layer to the map.
streets.addTo(map);
```

Let's break down what's happening in this code block:

1. We assign the `tileLayer()` method, as shown in the Quick Start Guide's "Setting up the map" section to the variable `streets`. Leaflet doesn't provide a tile layer. Instead, it offers various tile layer APIs.
2. The following URLs appear in the parentheses of our `tileLayer()` method:
   - The API URL with a reference to the `accessToken`
   - The `OpenStreetMap` URL inside the curly braces of our `tileLayer()` method
3. We add the `maxZoom` attribute and assign it a value of 18.
4. We add the `id` attribute and assign it `mapbox/streets-v11`, which will show the streets on the map.
5. We add the `accessToken` attribute and assign it the value of our `API_KEY`.
6. Finally, we call the `addTo()` function with our map object, `map` on our graymap object tile layer. The `addTo()` function will add the graymap object tile layer to our `let map`.

To change the map's style, change the map `id` using the list of Mapbox ids below:

- mapbox/streets-v11
- mapbox/outdoors-v11
- mapbox/light-v10
- mapbox/dark-v10
- mapbox/satellite-v9

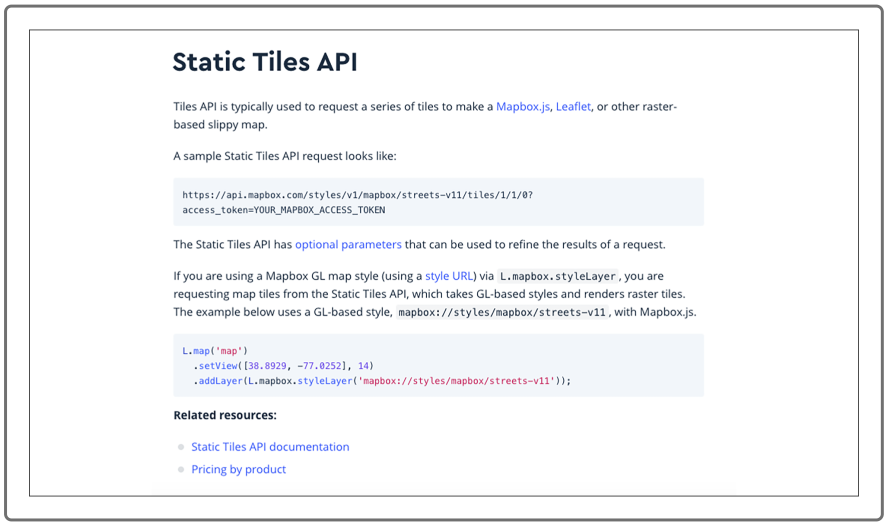
- mapbox/satellite-streets-v11

Use the Mapbox Styles API

To use the Mapbox Styles API, edit the URL in the Leaflet `tilelayer()` method, as detailed on the Leaflet website:

1. First, navigate to the **Mapbox Glossary (https://docs.mapbox.com/help/glossary/)**.

2. Search the **Static Tiles API (https://docs.mapbox.com/help/glossary/static-tiles-api/)**.



Static Tiles API

Tiles API is typically used to request a series of tiles to make a Mapbox.js, Leaflet, or other raster-based slippy map.

A sample Static Tiles API request looks like:

```
https://api.mapbox.com/styles/v1/mapbox/streets-v11/tiles/1/1/0?
access_token=YOUR_MAPBOX_ACCESS_TOKEN
```

The Static Tiles API has optional parameters that can be used to refine the results of a request.

If you are using a Mapbox GL map style (using a style URL) via `L.mapbox.styleLayer`, you are requesting map tiles from the Static Tiles API, which takes GL-based styles and renders raster tiles. The example below uses a GL-based style, `mapbox://styles/mapbox/streets-v11`, with Mapbox.js.

```
L.map('map')
  .setView([38.8929, -77.0252], 14)
  .addLayer(L.mapbox.styleLayer('mapbox://styles/mapbox/streets-v11'));
```

Related resources:

- Static Tiles API documentation
- Pricing by product

3. Copy this part of the URL: `https://api.mapbox.com/styles/v1/mapbox/streets-v11/tiles/`.

4. In your `tileLayer()` code, replace this part of the URL, `https://api.tiles.mapbox.com/v4/{id}/`, with the Mapbox Styles API URL you copied.

5. Remove the `.png` from the URL.

6. Remove the `id` attribute and the map style reference.

7. The code for the `tileLayer()` should look like the following:

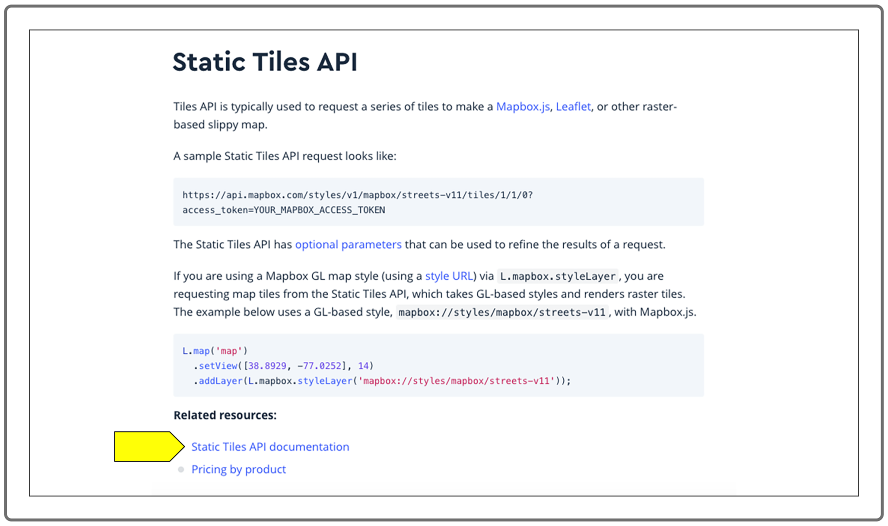
```
// We create the tile layer that will be the background of our map.
let streets = L.tileLayer('https://api.mapbox.com/styles/v1/mapbox/stree
attribution: 'Map data © <a href="https://www.openstreetmap.org/">OpenSt
    maxZoom: 18,
    accessToken: API_KEY
});

// Then we add our 'graymap' tile layer to the map.
streets.addTo(map);
```

NOTE

> If you have trouble editing the URL, copy the URL above.

8. To change your map style, click on the **Static Tiles API documentation (https://docs.mapbox.com/api/maps/#static-tiles)** link on the Static Tiles API page.



Static Tiles API

Tiles API is typically used to request a series of tiles to make a Mapbox.js, Leaflet, or other raster-based slippy map.

A sample Static Tiles API request looks like:

```
https://api.mapbox.com/styles/v1/mapbox/streets-v11/tiles/1/1/0?
access_token=YOUR_MAPBOX_ACCESS_TOKEN
```

The Static Tiles API has optional parameters that can be used to refine the results of a request.

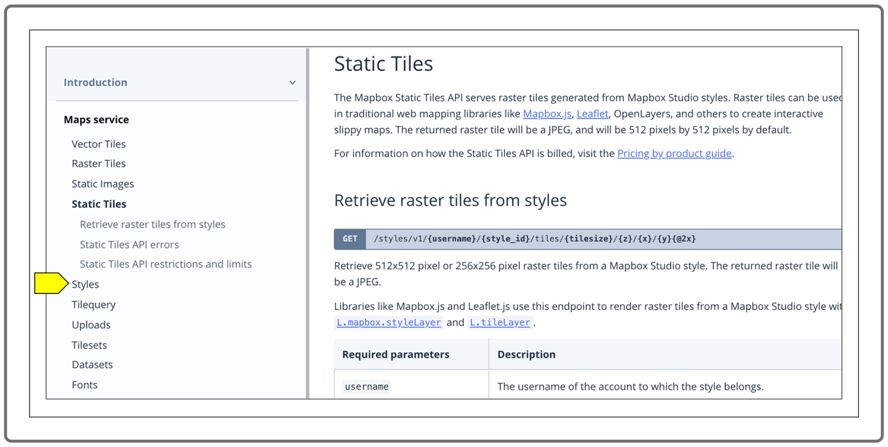
If you are using a Mapbox GL map style (using a style URL) via `L.mapbox.styleLayer`, you are requesting map tiles from the Static Tiles API, which takes GL-based styles and renders raster tiles. The example below uses a GL-based style, `mapbox://styles/mapbox/streets-v11`, with Mapbox.js.

```
L.map('map')
  .setView([38.8929, -77.0252], 14)
  .addLayer(L.mapbox.styleLayer('mapbox://styles/mapbox/streets-v11'));
```

Related resources:

- Static Tiles API documentation
- Pricing by product

9. On the left sidebar, click on the **Styles (https://docs.mapbox.com/api/maps/#styles)** link.



Introduction

Maps service

- Vector Tiles
- Raster Tiles
- Static Images
- **Static Tiles**
  - Retrieve raster tiles from styles
  - Static Tiles API errors
  - Static Tiles API restrictions and limits
- Styles
- Tilequery
- Uploads
- Tilesets
- Datasets
- Fonts

Static Tiles

The Mapbox Static Tiles API serves raster tiles generated from Mapbox Studio styles. Raster tiles can be used in traditional web mapping libraries like Mapbox.js, Leaflet, OpenLayers, and others to create interactive slippy maps. The returned raster tile will be a JPEG, and will be 512 pixels by 512 pixels by default.

For information on how the Static Tiles API is billed, visit the Pricing by product guide.

Retrieve raster tiles from styles

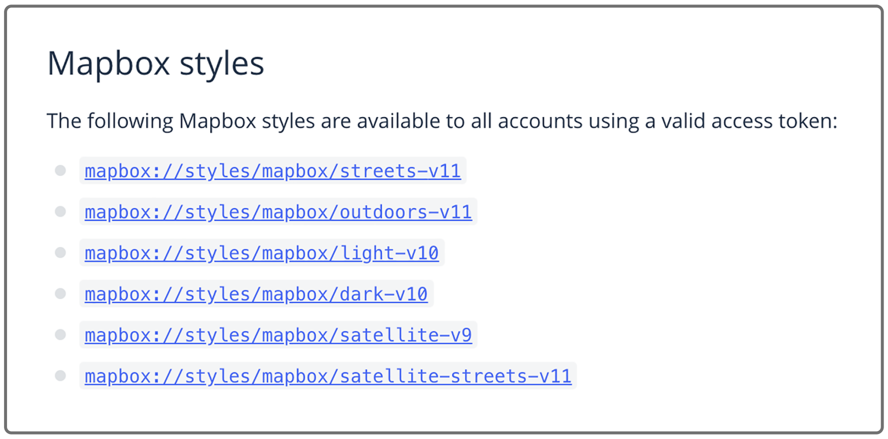
```
GET /styles/v1/{username}/{style_id}/tiles/{tilesize}/{z}/{x}/{y}{@2x}
```

Retrieve 512x512 pixel or 256x256 pixel raster tiles from a Mapbox Studio style. The returned raster tile will be a JPEG.

Libraries like Mapbox.js and Leaflet.js use this endpoint to render raster tiles from a Mapbox Studio style wit `L.mapbox.styleLayer` and `L.tileLayer`.

Required parameters	Description
`username`	The username of the account to which the style belongs.

10. Below the Styles subheading, find a list of different Mapbox styles.



Mapbox styles

The following Mapbox styles are available to all accounts using a valid access token:

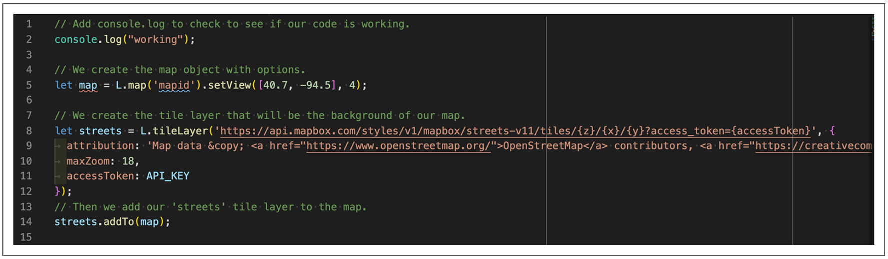
- mapbox://styles/mapbox/streets-v11
- mapbox://styles/mapbox/outdoors-v11
- mapbox://styles/mapbox/light-v10
- mapbox://styles/mapbox/dark-v10
- mapbox://styles/mapbox/satellite-v9
- mapbox://styles/mapbox/satellite-streets-v11

11. To change the map style, use the style given in the URLs (e.g., "streets-v11," "dark-v10," etc.).

NOTE

> For the rest of this module, we'll use the Static Tiles API format in the Leaflet `tileLayer()` method.

After adding all of the code, your `logic.js` file should look like the following:



```
// Add console.log to check to see if our code is working.
console.log("working");

// We create the map object with options.
let map = L.map('mapid').setView([40.7, -94.5], 4);

// We create the tile layer that will be the background of our map.
let streets = L.tileLayer('https://api.mapbox.com/styles/v1/mapbox/streets-v11/tiles/{z}/{x}/{y}?access_token={accessToken}', {
    attribution: 'Map data &copy; <a href="https://www.openstreetmap.org/">OpenStreetMap</a> contributors, <a href="https://creativecom
    maxZoom: 18,
    accessToken: API_KEY
});
// Then we add our 'streets' tile layer to the map.
streets.addTo(map);

```

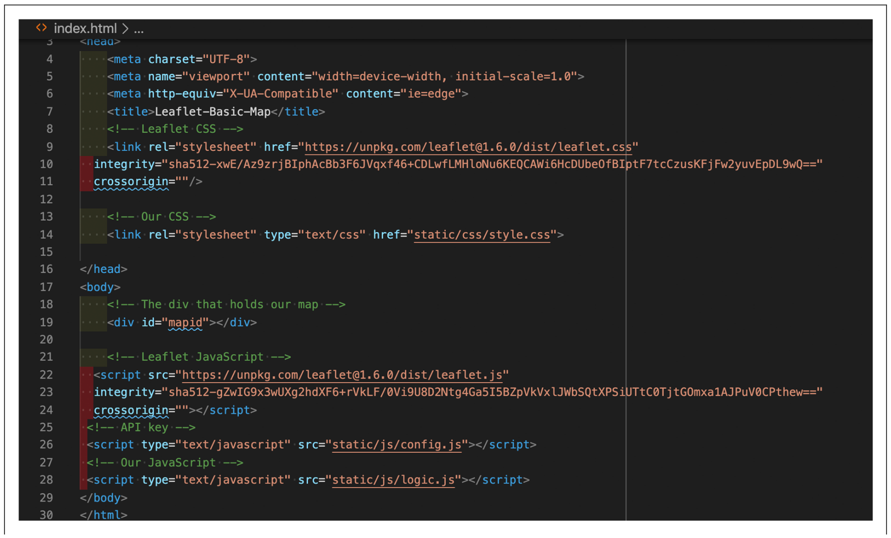
Next, we'll add the `logic.js` and `config.js` scripts to the HTML page.

Add JavaScript Tags to HTML Page

The last step is to allow our `index.html` file to use the `logic.js` and `config.js` scripts. To do this, open up the `index.html` file and add the following `<script>` elements below the Leaflet JavaScript link script and above the closing `</body>` element, and save the file.

```
<!-- API key -->
<script type="text/javascript" src="static/js/config.js"></script>
<!-- Our JavaScript -->
<script type="text/javascript" src="static/js/logic.js"></script>
```

Your `index.html` file should look like the following after adding all the elements to the head and body sections:



```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Leaflet-Basic-Map</title>
    <!-- Leaflet CSS -->
    <link rel="stylesheet" href="https://unpkg.com/leaflet@1.6.0/dist/leaflet.css"
  integrity="sha512-xwE/Az9zrjBIphAcBb3F6JVqxf46+CDLwfLMHloNu6KEQCAWi6HcDUbeOfBIptF7tcCzusKFjFw2yuvEpDL9wQ=="
  crossorigin=""/>

    <!-- Our CSS -->
    <link rel="stylesheet" type="text/css" href="static/css/style.css">

</head>
<body>
    <!-- The div that holds our map -->
    <div id="mapid"></div>

    <!-- Leaflet JavaScript -->
  <script src="https://unpkg.com/leaflet@1.6.0/dist/leaflet.js"
  integrity="sha512-gZwIG9x3wUXg2hdXF6+rVkLF/0Vi9U8D2Ntg4Ga5I5BZpVkVxlJWbSQtXPSiUTtC0TjtGOmxa1AJPuV0CPthew=="
  crossorigin=""></script>
 <!-- API key -->
 <script type="text/javascript" src="static/js/config.js"></script>
 <!-- Our JavaScript -->
 <script type="text/javascript" src="static/js/logic.js"></script>
</body>
</html>
```

Open the Map in the Browser

Now, open the `index.html` file.

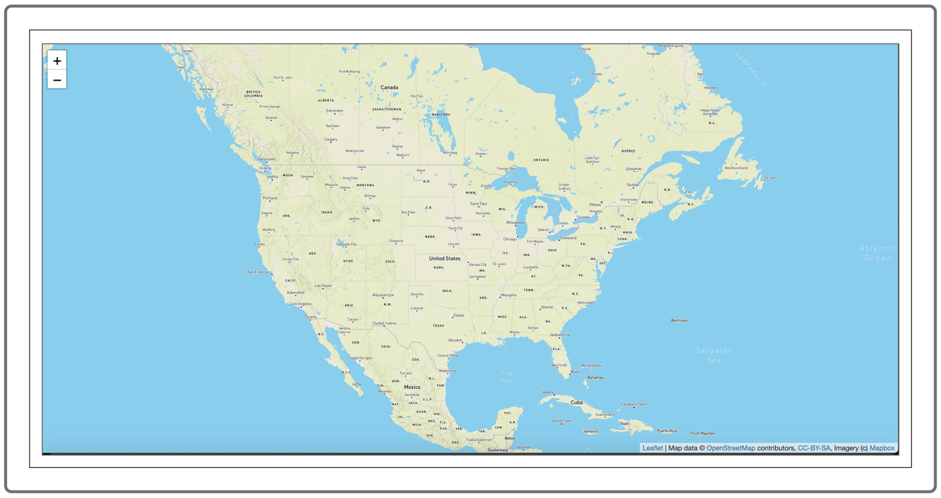
REWIND

To open the `index.html` file in the browser:

1. Open the command line and navigate to your Mapping_Earthquakes folder. The `index.html` file should be in the top-level of that folder.

2. On the command line, type `python -m http.server` and press Enter.

The basic map should look like the following in your web browser:


+
−
Canada
United States
Mexico
Cuba
Atlantic Ocean
Sargasso Sea
Leaflet | Map data © OpenStreetMap contributors, CC-BY-SA, Imagery (c) Mapbox

Congratulations on creating your first map using Leaflet and the Mapbox API!

NOTE

> For more information, see the **Leaflet documentation on methods using the map object and tile layer (https://leafletjs.com/reference-1.6.0.html#map-example)**.