KATHMANDU UNIVERSITY

Dhulikhel, Kavre

DOCUMENTATION FOR ASSIGNMENT-2

SUBMITTED TO:
Suman Baral

SUBMITTED BY:
Nabin Poudel
Roll no:42
KUGE 2020

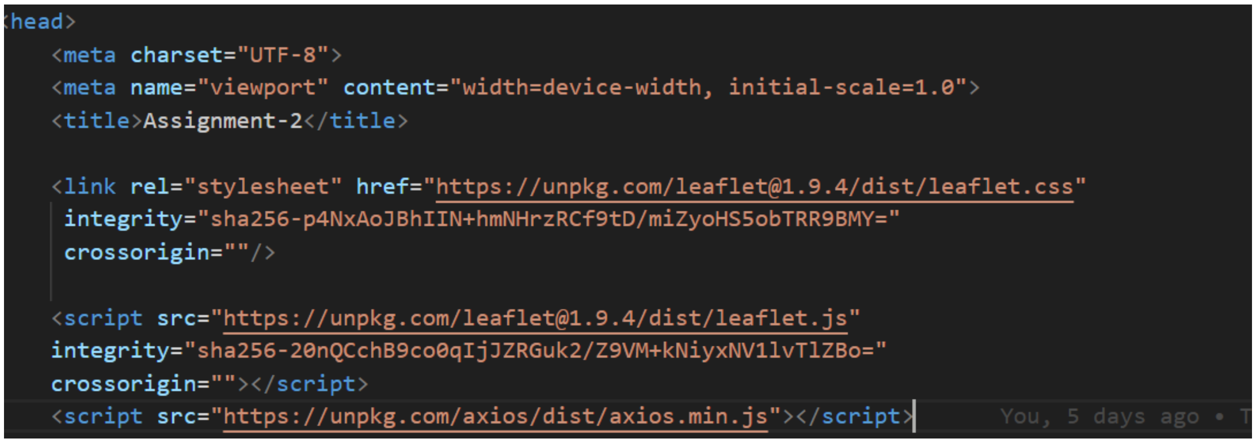
Date:2024/12/10

This is the documentation for the website with features like BaseMap, Marker, geojson loaded in the map.

The GitHub repository link for this project is here

Section1: Preparing the Map

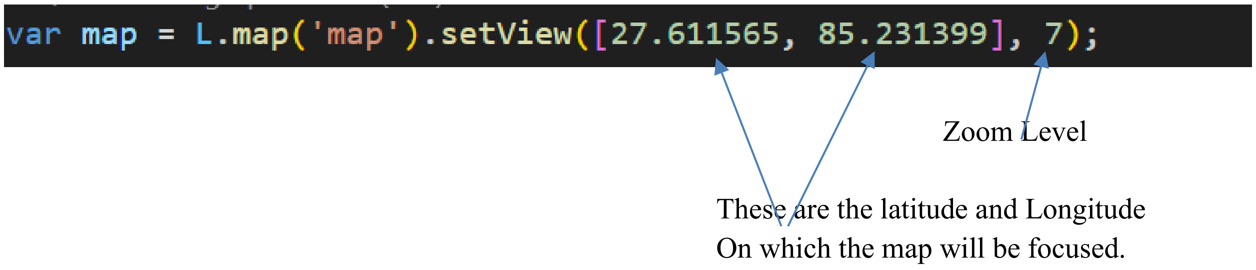
1. First of all HTML file is created. In this HTML file. The necessary scripts are added inside the head section of the html files. The appropriate tilte should be given to the website.



```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Assignment-2</title>

    <link rel="stylesheet" href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css"
     integrity="sha256-p4NxAoJBhIIN+hmNHrzRCf9tD/miZyoHS5obTRR9BMY="
     crossorigin=""/>

    <script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js"
    integrity="sha256-20nQCchB9co0qIjJZRGuk2/Z9VM+kNiyxNV1lvTlZBo="
    crossorigin=""></script>
    <script src="https://unpkg.com/axios/dist/axios.min.js"></script>        You, 5 days ago • T
```

(IMPORTANT) The leaflet Javascript and leaflet stylesheet are added in the head section. Both can be found in the leaflet's official documentation. Here is the link.

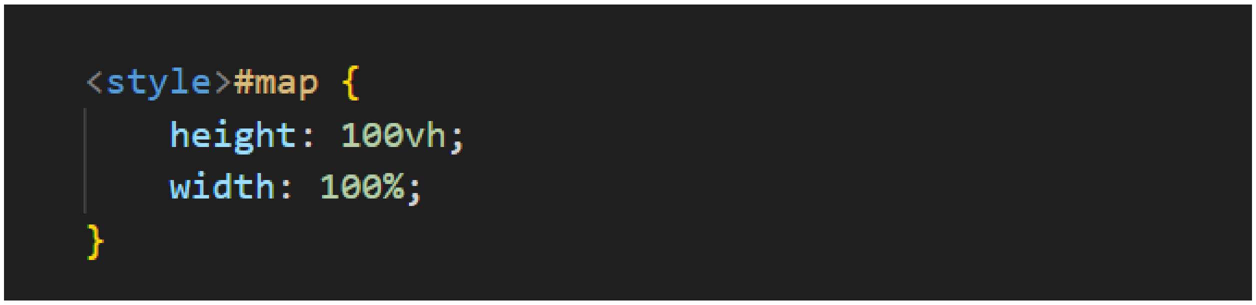
2. Now we'll initialize the map and set its view to our chosen geographical coordinates and a zoom level:



```
var map = L.map('map').setView([27.611565, 85.231399], 7);
```

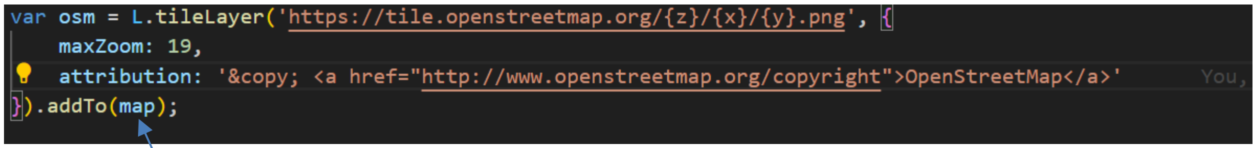
Zoom Level
These are the latitude and Longitude
On which the map will be focused.

For this to work we have to make a division for the map in the body section of the HTML file. And the height and width of this division are made 100vh and 100% respectively in the style section. (IMPORTANT)



```
<style>#map {
    height: 100vh;
    width: 100%;
}
```

3. Now the tileLayer is added to the map. Which will be the baseLayer for this assignment. The others tilelayers can also be added. Here

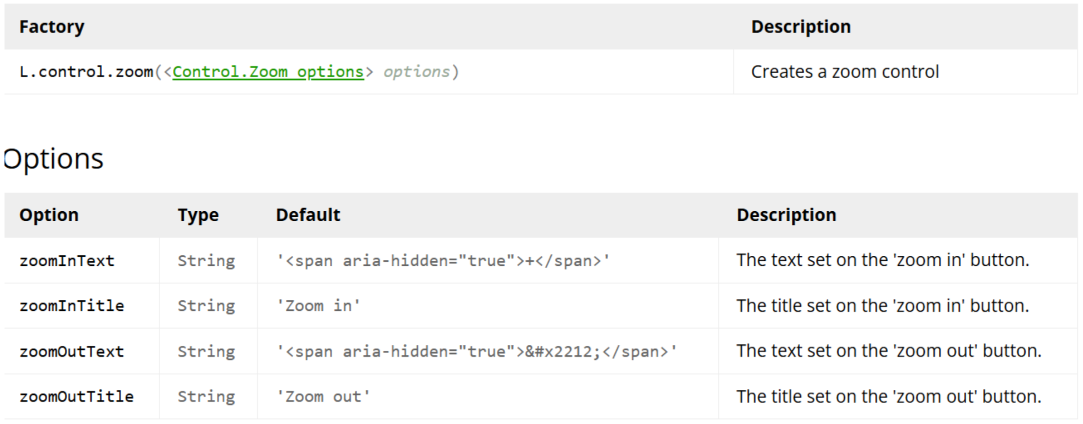


```
var osm = L.tileLayer('https://tile.openstreetmap.org/{z}/{x}/{y}.png', {
    maxZoom: 19,
    attribution: '&copy; <a href="http://www.openstreetmap.org/copyright">OpenStreetMap</a>'        You,
}).addTo(map);
```

This will add the tilelayer to map

Section2: Adding map controls

4. Zoom: The Zoom buttons are added by default, except when you set its zoom Control option to false. You can also change the texts for Zoom In and Zoom Out text.

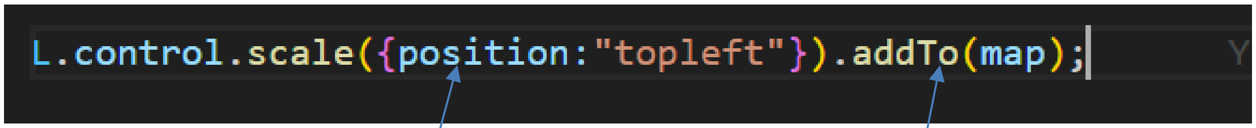


Factory	Description
`L.control.zoom(<Control.Zoom options> options)`	Creates a zoom control

Options

Option	Type	Default	Description
zoomInText	String	`'<span aria-hidden="true">+</span>'`	The text set on the 'zoom in' button.
zoomInTitle	String	`'Zoom in'`	The title set on the 'zoom in' button.
zoomOutText	String	`'<span aria-hidden="true">&#x2212;</span>'`	The text set on the 'zoom out' button.
zoomOutTitle	String	`'Zoom out'`	The title set on the 'zoom out' button.

5. Now Scale is added to maps as:



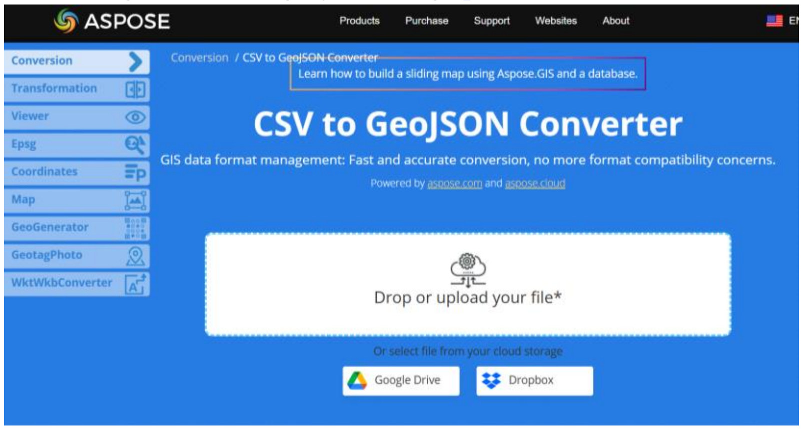
```
L.control.scale({position:"topleft"}).addTo(map);
```

This is then added to map

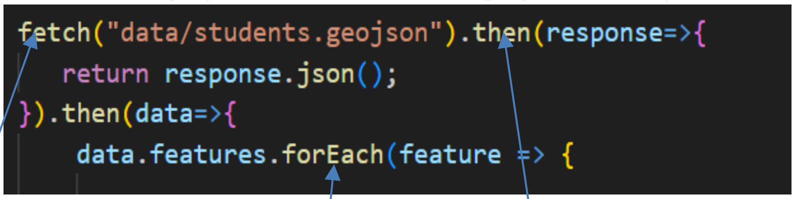
Position is set to Top Left

Section3: Adding student GeoJSON

6. Now we were provided with the student CSV file. The csv file contains name, their address, Municipality, District. Now our objective is to display the location of individual students in the map by using marker. And display the infos of the students as popup , when clicked on that marker.
   i. Converting the CSV to geojson using [apose]


ASPOSE
Products
Purchase
Support
Websites
About
Conversion
Transformation
Viewer
Epsg
Coordinates
Map
GeoGenerator
GeotagPhoto
WktWkbConverter
Conversion / CSV to GeoJSON Converter
Learn how to build a sliding map using Aspose.GIS and a database.
CSV to GeoJSON Converter
GIS data format management: Fast and accurate conversion, no more format compatibility concerns.
Powered by aspose.com and aspose.cloud
Drop or upload your file*
Or select file from your cloud storage
Google Drive
Dropbox

   ii. The obtained geojson is the added to the project directory and then fetched as

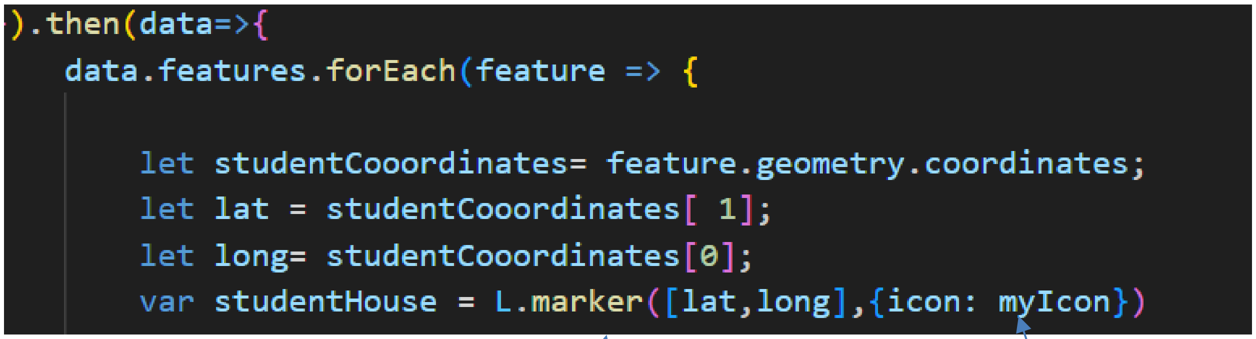


```
fetch("data/students.geojson").then(response=>{
   return response.json();
}).then(data=>{
    data.features.forEach(feature => {
```

2.The geojson is then converted to JSON

1. This will fetch the geojson file 3. forEach loop is used to access the details of each person

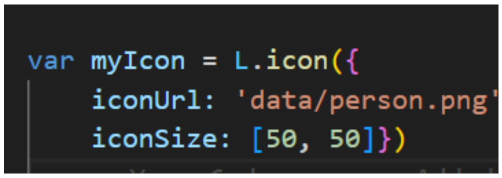
   iii. The location information of the student are the used to add marker in their location as:



```
).then(data=>{
  data.features.forEach(feature => {

     let studentCooordinates= feature.geometry.coordinates;
     let lat = studentCooordinates[ 1];
     let long= studentCooordinates[0];
     var studentHouse = L.marker([lat,long],{icon: myIcon})
```

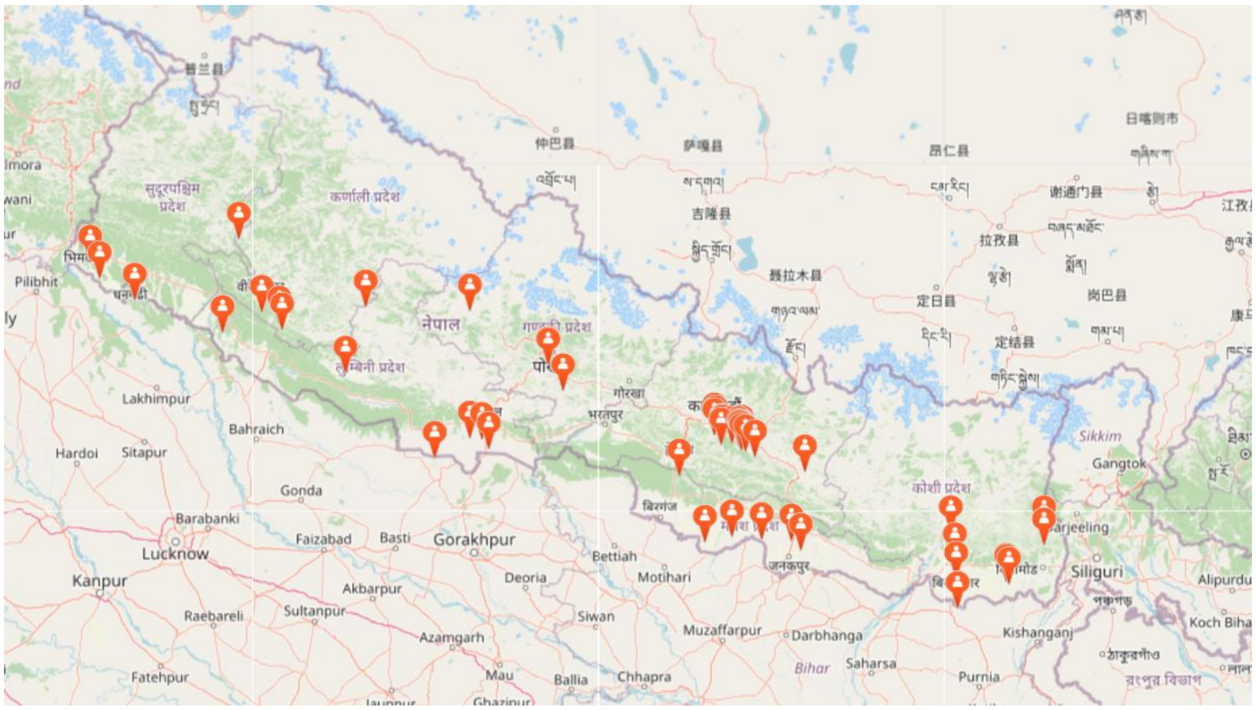
This is the function that is used
To add marker in location , (lat , long)

marker can be
customized as

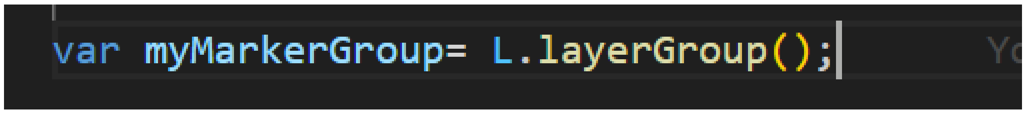


```
var myIcon = L.icon({
    iconUrl: 'data/person.png'
    iconSize: [50, 50]})
```

The result is:

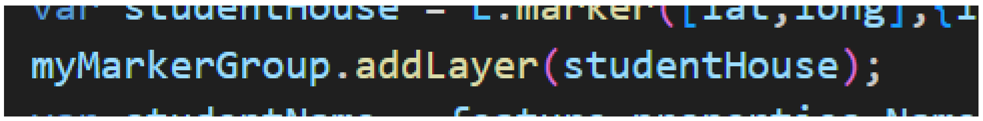


The marker is then added to myMarkerGroup as:



```
var myMarkerGroup= L.layerGroup();
```

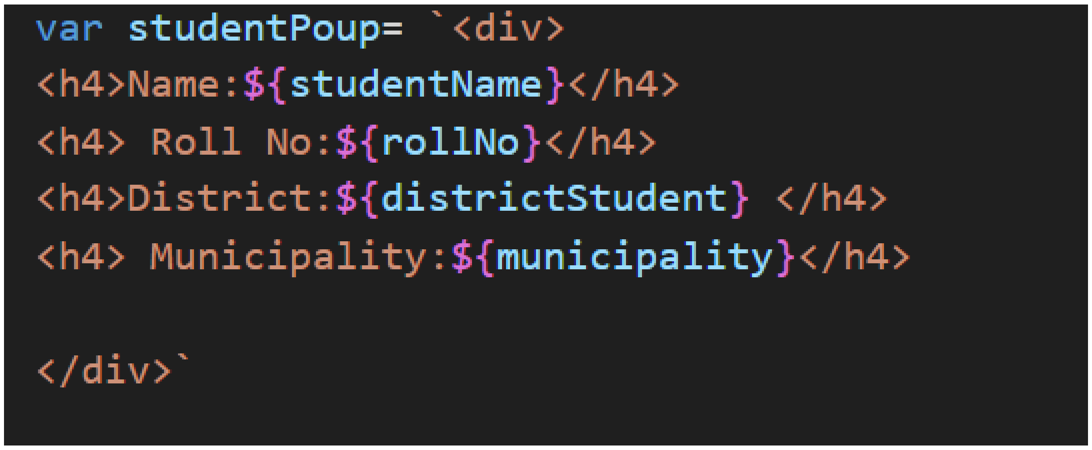
This will create a layer group name myMarkerGroup.



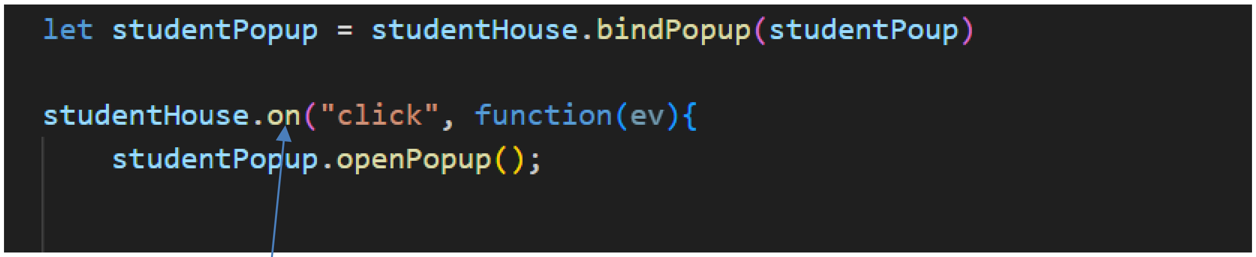
```
var studentHouse = L.marker([lat,long],{i
myMarkerGroup.addLayer(studentHouse);
var studentName = feature.properties.Name
```

This will add the markers to myMarkerGroup. Note: This should be inside for each loop.

iv. Now the for popup. The popup content is: They are made using div.

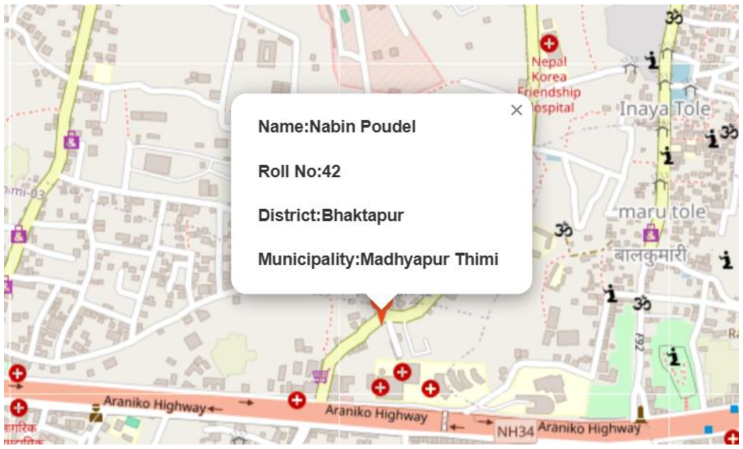


```
var studentPoup= `<div>
<h4>Name:${studentName}</h4>
<h4> Roll No:${rollNo}</h4>
<h4>District:${districtStudent} </h4>
<h4> Municipality:${municipality}</h4>

</div>`
```



```
let studentPopup = studentHouse.bindPopup(studentPoup)

studentHouse.on("click", function(ev){
    studentPopup.openPopup();
```

This is the event function. This syntax means. When marker studentHouse is clicked. It will display popup content as set previously.



Name:Nabin Poudel

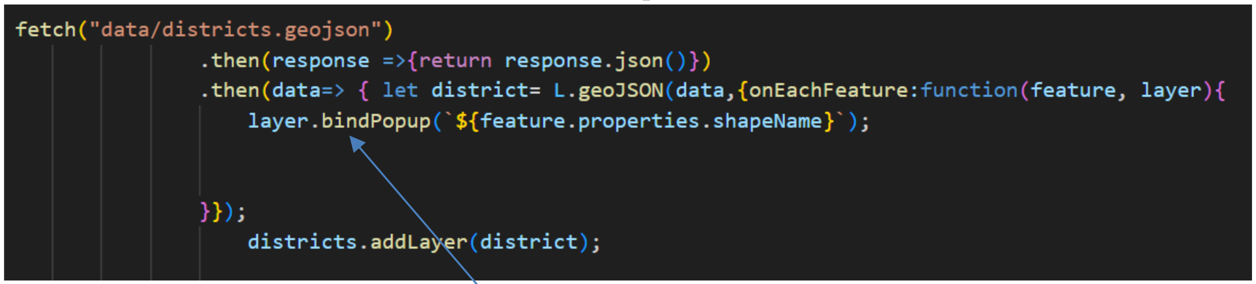
Roll No:42

District:Bhaktapur

Municipality:Madhyapur Thimi

For Districts, Municipalities and Provinces

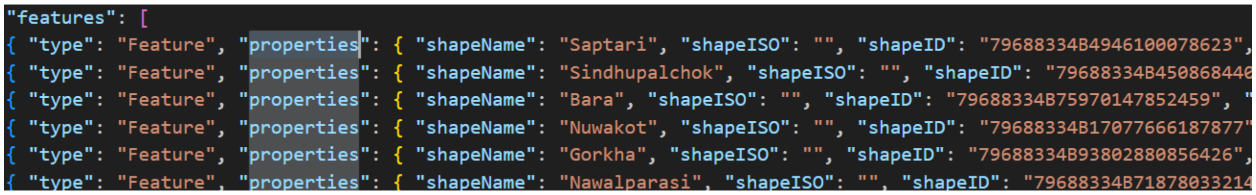
It follows similar process as students. But option method **onEachFeature** is used. A Function that will be called once for each created Feature. This function loops over each feature.



```
fetch("data/districts.geojson")
                .then(response =>{return response.json()})
                .then(data=> { let district= L.geoJSON(data,{onEachFeature:function(feature, layer){
                    layer.bindPopup(`${feature.properties.shapeName}`);


                }});
                    districts.addLayer(district);
```

This will bind popup to each district, and show district name.

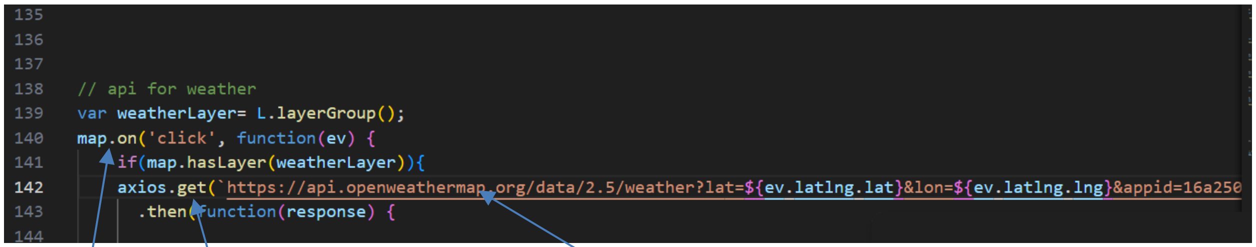
The structure of district geojson is as:



```
"features": [
{ "type": "Feature", "properties": { "shapeName": "Saptari", "shapeISO": "", "shapeID": "79688334B4946100078623",
{ "type": "Feature", "properties": { "shapeName": "Sindhupalchok", "shapeISO": "", "shapeID": "79688334B450868440
{ "type": "Feature", "properties": { "shapeName": "Bara", "shapeISO": "", "shapeID": "79688334B75970147852459", "
{ "type": "Feature", "properties": { "shapeName": "Nuwakot", "shapeISO": "", "shapeID": "79688334B17077666187877"
{ "type": "Feature", "properties": { "shapeName": "Gorkha", "shapeISO": "", "shapeID": "79688334B93802880856426",
{ "type": "Feature", "properties": { "shapeName": "Nawalparasi", "shapeISO": "", "shapeID": "79688334B71878033214
```

Municipalities and provinces have the same steps to load geojson and display popups.

Section4: API integration

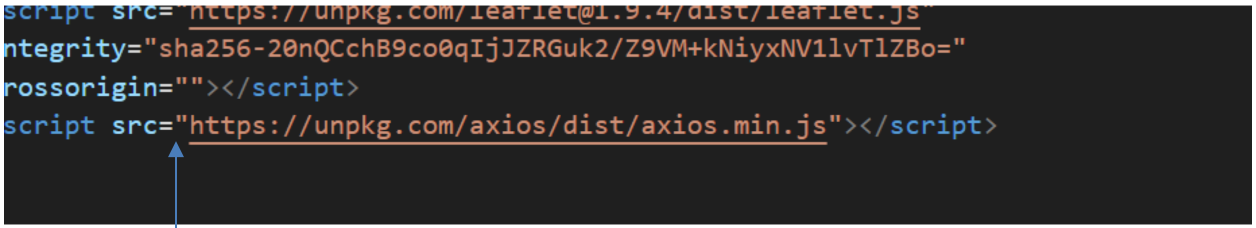


```


// api for weather
var weatherLayer= L.layerGroup();
map.on('click', function(ev) {
    if(map.hasLayer(weatherLayer)){
    axios.get(`https://api.openweathermap.org/data/2.5/weather?lat=${ev.latlng.lat}&lon=${ev.latlng.lng}&appid=16a250
      .then(function(response) {

```

Event functiom

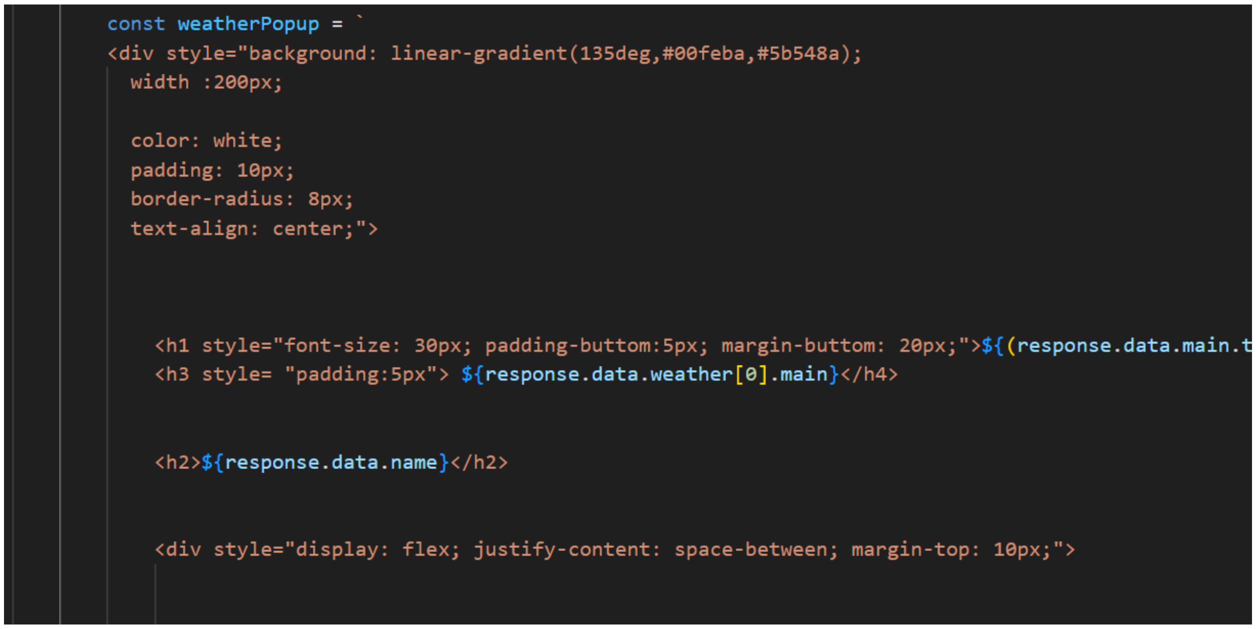
This is the axios method
To send get request

Api for weather



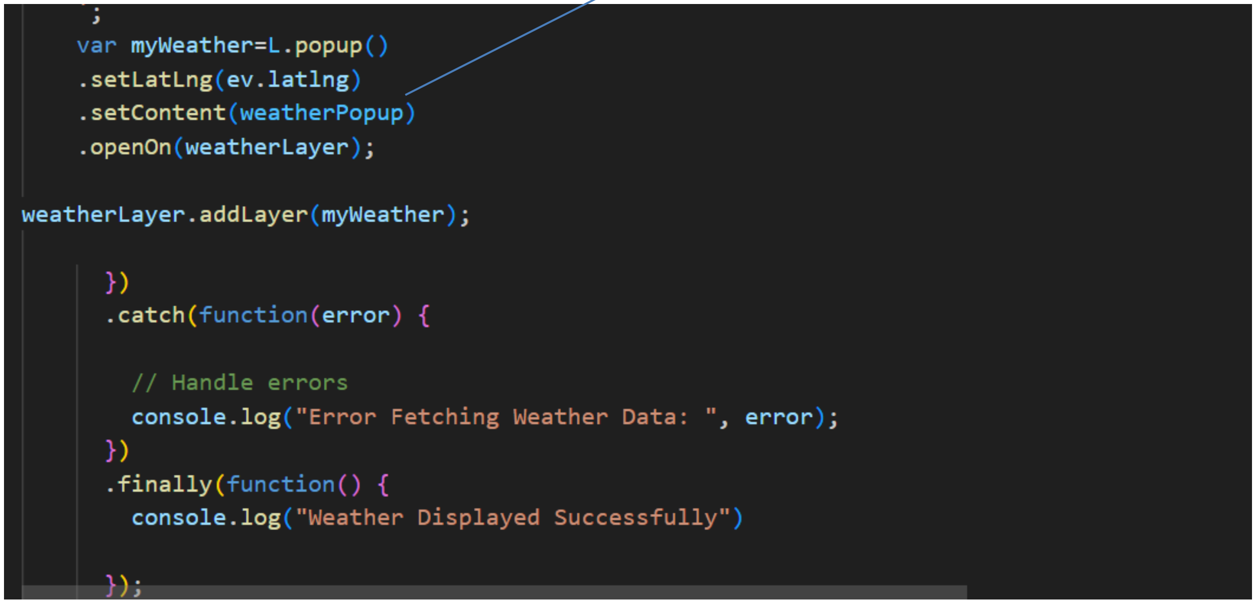
```
script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js"
ntegrity="sha256-20nQCchB9co0qIjJZRGuk2/Z9VM+kNiyxNV1lvTlZBo="
rossorigin=""></script>
script src="https://unpkg.com/axios/dist/axios.min.js"></script>
```

For the axios get to work we must include this script in head section of html.

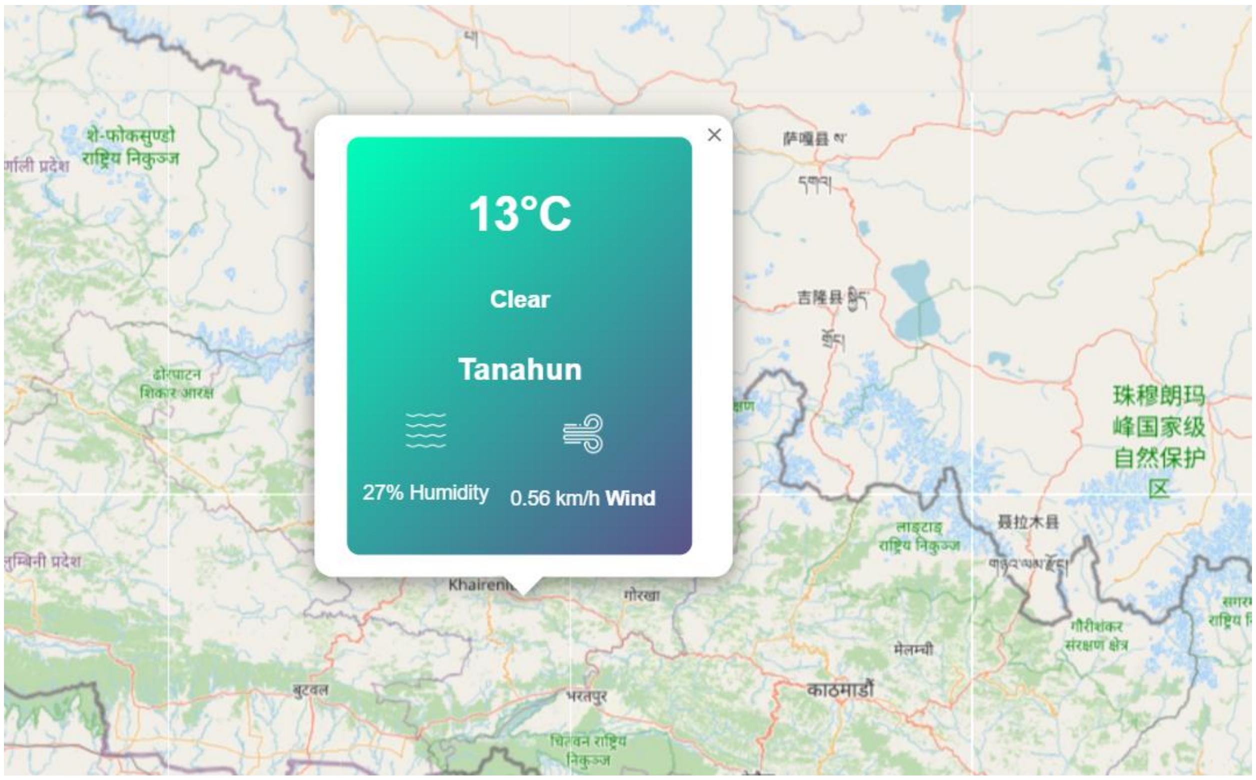

```
    const weatherPopup = `
    <div style="background: linear-gradient(135deg,#00feba,#5b548a);
      width :200px;

      color: white;
      padding: 10px;
      border-radius: 8px;
      text-align: center;">



        <h1 style="font-size: 30px; padding-buttom:5px; margin-buttom: 20px;">${(response.data.main.t
        <h3 style= "padding:5px"> ${response.data.weather[0].main}</h4>


        <h2>${response.data.name}</h2>


        <div style="display: flex; justify-content: space-between; margin-top: 10px;">
```

This will set the styled popup using css.

This will display the popup in the location
Where it is clicked.


```
      ;
    var myWeather=L.popup()
    .setLatLng(ev.latlng)
    .setContent(weatherPopup)
    .openOn(weatherLayer);

weatherLayer.addLayer(myWeather);

      })
      .catch(function(error) {

        // Handle errors
        console.log("Error Fetching Weather Data: ", error);
      })
      .finally(function() {
        console.log("Weather Displayed Successfully")

      });
```

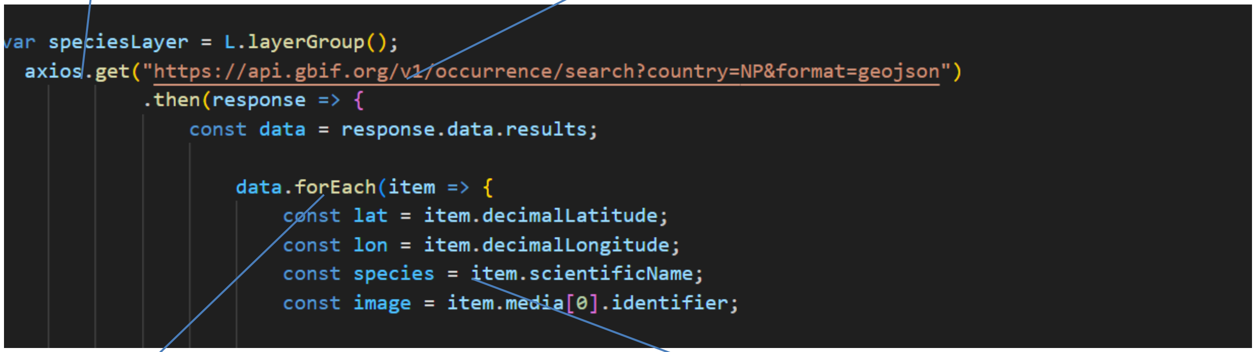
The result:



For Global Biodiversity Data:

This will make layer Group
For species marker

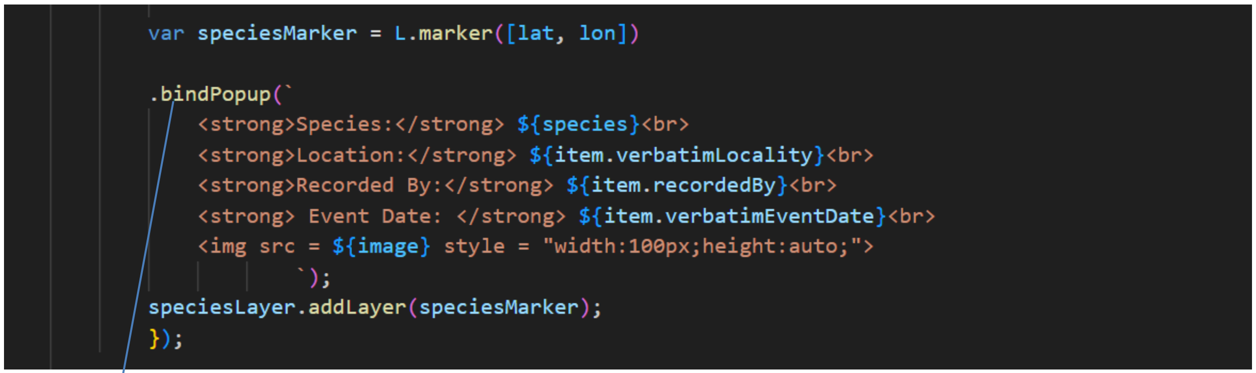
This is the axios get request.
This will return the json file, which contains data for animals spotting location,and their infos.



```
var speciesLayer = L.layerGroup();
  axios.get("https://api.gbif.org/v1/occurrence/search?country=NP&format=geojson")
        .then(response => {
            const data = response.data.results;

                data.forEach(item => {
                    const lat = item.decimalLatitude;
                    const lon = item.decimalLongitude;
                    const species = item.scientificName;
                    const image = item.media[0].identifier;
```

For each loop to extract data for each spotting

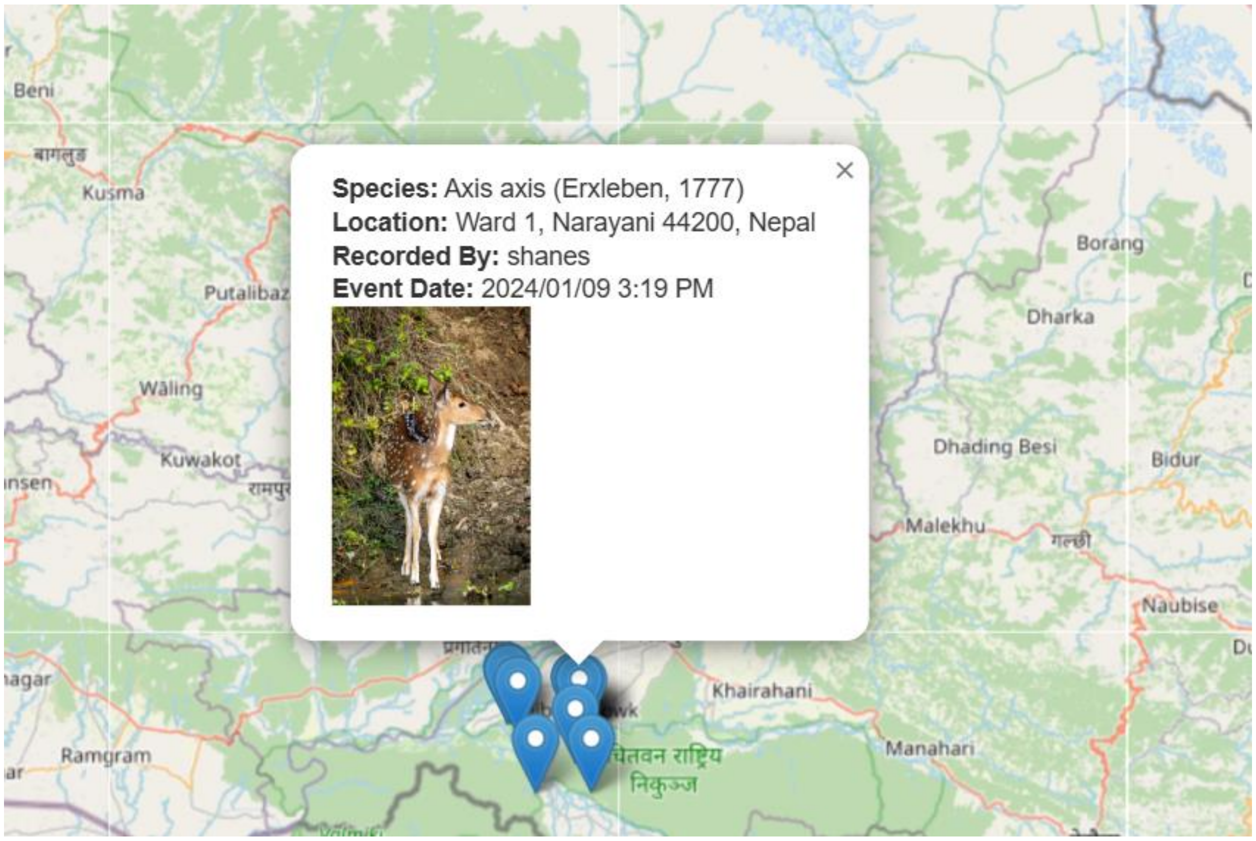
data is extracted for json for each spotting



```
        var speciesMarker = L.marker([lat, lon])

        .bindPopup(`
            <strong>Species:</strong> ${species}<br>
            <strong>Location:</strong> ${item.verbatimLocality}<br>
            <strong>Recorded By:</strong> ${item.recordedBy}<br>
            <strong> Event Date: </strong> ${item.verbatimEventDate}<br>
            <img src = ${image} style = "width:100px;height:auto;">
                    `);
        speciesLayer.addLayer(speciesMarker);
        });
```

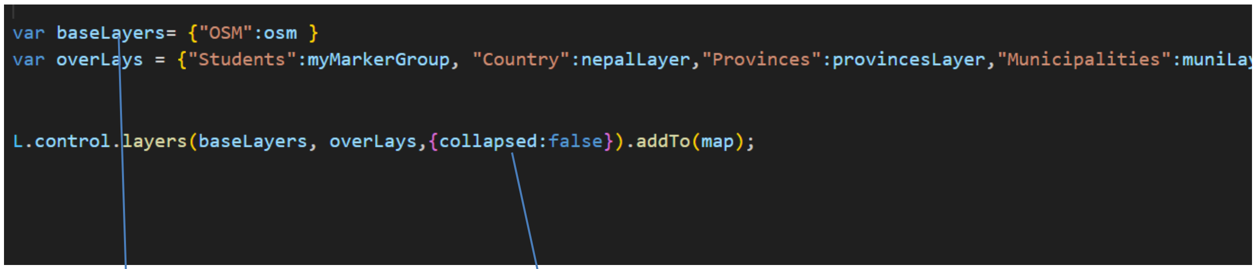
Popup for each spotting.

Result



Species: Axis axis (Erxleben, 1777)
Location: Ward 1, Narayani 44200, Nepal
Recorded By: shanes
Event Date: 2024/01/09 3:19 PM

Section5: Adding base map and all other overlays as custom overlays



```
var baseLayers= {"OSM":osm }
var overLays = {"Students":myMarkerGroup, "Country":nepalLayer,"Provinces":provincesLayer,"Municipalities":muniLay


L.control.layers(baseLayers, overLays,{collapsed:false}).addTo(map);
```

The base layers and overlays are made

This will make the controls not collapsible and always Expanded