ENGO 551 Lab 3
Example ChatGPT Prompts
Mitchell Aitken

Query

On my html page in the console I am getting these errors, what do they mean?

```
map-init.js:14 Uncaught ReferenceError: OverlappingMarkerSpiderfier is not defined
    at map-init.js:14:11
map-init.js:27 Fetched data:
{type: 'FeatureCollection', features: Array(1000), crs: {…}}
map-init.js:65 Adding marker:
i {options: {…}, _latlng: D, _initHooksCalled: true, _popup: i, _events: {…}, …}
map-init.js:27 Fetched data:
{type: 'FeatureCollection', features: Array(1000), crs: {…}}
map-init.js:65 Adding marker:
i {options: {…}, _latlng: D, _initHooksCalled: true, _popup: i, _events: {…}, …}
map-init.js:27 Fetched data:
{type: 'FeatureCollection', features: Array(1000), crs: {…}}
map-init.js:65 Adding marker:
i {options: {…}, _latlng: D, _initHooksCalled: true, _popup: i, _events: {…}, …}
map-init.js:27 Fetched data:
{type: 'FeatureCollection', features: Array(1000), crs: {…}}
map-init.js:65 Adding marker:
i {options: {…}, _latlng: D, _initHooksCalled: true, _popup: i, _events: {…}, …}
map-init.js:27 Fetched data:
{type: 'FeatureCollection', features: Array(1000), crs: {…}}
map-init.js:65 Adding marker:
i {options: {…}, _latlng: D, _initHooksCalled: true, _popup: i, _events: {…}, …}
map-init.js:27 Fetched data:
{type: 'FeatureCollection', features: Array(1000), crs: {…}}
map-init.js:65 Adding marker:
i {options: {…}, _latlng: D, _initHooksCalled: true, _popup: i, _events: {…}, …}
map-init.js:27 Fetched data:
{type: 'FeatureCollection', features: Array(1000), crs: {…}}
map-init.js:65 Adding marker:
i {options: {…}, _latlng: D, _initHooksCalled: true, _popup: i, _events: {…}, …}
map-init.js:27 Fetched data:
{type: 'FeatureCollection', features: Array(1000), crs: {…}}
map-init.js:65 Adding marker:
i {options: {…}, _latlng: D, _initHooksCalled: true, _popup: i, _events: {…}, …}
8
map-init.js:68 Uncaught (in promise) TypeError: Cannot read properties of undefined (reading 'addMarker')
    at Object.onEachFeature (map-init.js:68:21)
    at i.addData (GeoJSON.js:127:12)
    at i.addData (GeoJSON.js:107:11)
    at i.initialize (GeoJSON.js:92:9)
    at new i (Class.js:22:20)
    at Object.Ke (GeoJSON.js:439:9)
    at addBuildingPermitsToMap (map-init.js:47:7)
    at map-init.js:42:13
```

Response:
The errors you're encountering seem to revolve around two main issues:

OverlappingMarkerSpiderfier is not defined: This error indicates that the OverlappingMarkerSpiderfier library is either not included in your HTML file or is not properly loaded by the time your map-init.js script attempts to use it. To resolve this, ensure that you include the OverlappingMarkerSpiderfier library in your HTML file before your map-init.js script. If you haven't already done so, find a CDN link or download the library and link it in your HTML.

TypeError: Cannot read properties of undefined (reading 'addMarker'): This error suggests that at the point in your script where addMarker is called, the object or variable expected to have this method is undefined. This could be related to the OverlappingMarkerSpiderfier issue or could be an issue with how you're initializing or using your map or marker cluster group.

To address these issues, you can follow these steps:

For OverlappingMarkerSpiderfier:
Ensure you include the OverlappingMarkerSpiderfier library in your HTML, similar to how you've included other scripts. If you don't have the URL for the library, you can search for "OverlappingMarkerSpiderfier CDN" to find one.
html
Copy code

```
<script src="path_to_OverlappingMarkerSpiderfier.js"></script>
```

Replace "path_to_OverlappingMarkerSpiderfier.js" with the actual path or URL to the OverlappingMarkerSpiderfier script.

For the addMarker issue:
Verify that the object you're calling addMarker on is correctly initialized and is not undefined. This might be your OverlappingMarkerSpiderfier instance or another map-related object.
Ensure that the code that initializes this object is executed before you try to use addMarker. This might involve reordering your script or ensuring that certain functions are only called after the map and relevant plugins are fully loaded.
General Debugging Tips:
Check the order of your script tags to ensure dependencies are loaded in the correct order.
Use console.log statements to debug and check the state of your variables or objects before you call methods on them.
Ensure that any asynchronous operations (like fetching data) are properly handled before you use the results in your map.
If these suggestions don't resolve the issue, it might be helpful to review the specific sections of your map-init.js script where these errors are occurring to ensure that everything is set up correctly

Query
What does this error mean?

```
Uncaught Error: Map container not found.
```

```
    at i._initContainer (Map.js:1103:10)
    at i.initialize (Map.js:136:8)
    at new i (Class.js:22:20)
    at t.map (Map.js:1728:9)
    at map-init.js:3:13
map-init.js:12 Uncaught TypeError: L.markerClusterGroup is not a function
    at map-init.js:12:28
map-init.js:34 Uncaught TypeError: Cannot read properties of undefined (reading 'clearLayers')
    at updateMap (map-init.js:34:24)
    at HTMLButtonElement.onclick (index.html:69:39)
updateMap @ map-init.js:34
onclick @ index.html:69
```

Response
Uncaught Error: Map container not found: This error occurs when the Leaflet library tries to initialize a map before the HTML element intended to contain the map (usually a <div> with a specific id like mapid) is loaded into the DOM. This can happen if your JavaScript code that initializes the map (var map = L.map('mapid')...) runs before the browser has fully parsed the HTML of the page.

Uncaught TypeError: L.markerClusterGroup is not a function: This error suggests that the Leaflet MarkerCluster plugin hasn't been correctly loaded before you try to use L.markerClusterGroup(). This could be due to the script not being loaded properly, being loaded after your map-init.js script, or due to a version incompatibility.

Query
instead of using this search thing, can you use a custom widget that looks a little nicer?

```
<!-- Date Range Input -->
    <div style="margin-top: 20px;">
        <label for="startDate">Start Date:</label>
        <input type="date" id="startDate">
        <label for="endDate">End Date:</label>
        <input type="date" id="endDate">
        <button onclick="updateMap()">Search</button>
    </div>
```

Response
To use a custom date range picker widget that looks nicer and provides a more user-friendly interface, you can integrate a third-party JavaScript library such as daterangepicker.js. This library offers a wide range of options to customize the date picker's appearance and functionality.

First, you'll need to include the daterangepicker.js library and its CSS in your HTML file. You can include these from a CDN for simplicity. Here's how you can update your HTML to include the necessary resources and set up a basic date range picker:

Query
With the code below, I can see a pop up after clicking on the marker, but either I cant see what it actually says or theres no details showing for it, can you suggest me some things to check? (pasted javascript code)

Response
If the pop-ups are appearing but not displaying the expected details, there could be a few reasons for this. Let's troubleshoot and ensure the pop-up content is correctly set up:

Check GeoJSON Property Names: Make sure that the property names used in the popupContent string match exactly with those provided in the GeoJSON data from the Open Calgary API. The property names are case-sensitive and must correspond precisely to those in the GeoJSON structure. For example, if the GeoJSON properties are issued_date instead of issueddate, you'll need to adjust your code accordingly.

Inspect the GeoJSON Data: Log the GeoJSON data to the console to inspect its structure and confirm that the properties you're trying to display actually exist and contain data. You can add a console.log(feature) inside the onEachFeature function to see the structure of each feature.

javascript
Copy code
```
function addBuildingPermitsToMap(data) {
    L.geoJson(data, {
        onEachFeature: function (feature, layer) {
            console.log(feature); // Log the feature to inspect its properties
            // ...rest of the code
        }
    });
}
```
Ensure Proper HTML Structure in the Pop-up: Make sure the HTML structure you're building for the popupContent is correct. Incorrect HTML can lead to display issues. For simplicity, start with plain text to ensure the data is displayed, then gradually add HTML formatting.

javascript
Copy code
```
var popupContent = 'Issued Date: ' + feature.properties.issueddate; // Start simple
```
Check for Undefined Properties: If any of the properties are undefined, it could cause the content not to display correctly. You can add checks or fallbacks for undefined properties.

javascript
Copy code
```
var popupContent = `
    <strong>Issued Date:</strong> ${feature.properties.issueddate || 'N/A'}<br>
    <strong>Work Class Group:</strong> ${feature.properties.workclassgroup || 'N/A'}<br>
    // ...rest of the properties
`;
```
Error Handling: Add error handling to your fetch request to catch any issues that might occur when retrieving the data.

javascript
Copy code

```
function fetchBuildingPermits(startDate, endDate) {
    const url = `https://data.calgary.ca/resource/c2es-76ed.geojson?$where=issueddate >= '${startDate}' AND issueddate <= '${endDate}'`;

    return fetch(url)
        .then(response => {
            if (!response.ok) {
                throw new Error('Network response was not ok');
            }
            return response.json();
        })
        .catch(error => console.error('Error fetching data: ', error));
}
```

If you've gone through these steps and the issue persists, double-check the data being returned by the API call and ensure that the JavaScript console doesn't show any errors. If there are errors or warnings in the console, addressing them might solve the issue with the pop-ups.