**Mapbox GL JS API Used into React Mapbox GL Components**

**1. Showing a Popup on click**

**Step 1.1**

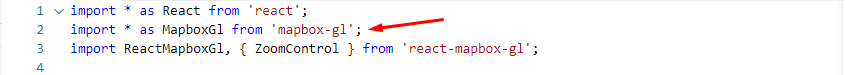
Open our existing PCF component along with the react mapbox gl component.

**Step 1.2**

Open *TSMapComponent/src/components/Map1/index.tsx* the file

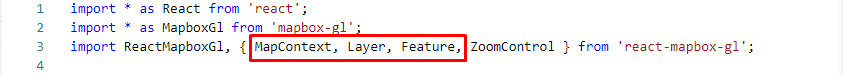
**Step 1.2.1** Add below code to use mapbox-gl API.

import \* as MapboxGl from 'mapbox-gl';



**Step 1.2.2** import below components from **react-mapbox-gl**

* MapContext
* Layer
* Feature



**Step 1.2.3** Add below methods into **Map1FC** component

* showPopup
  + mapboxgl JS API used in this method only Ex. **MapboxGl.Popup()**
* showCursorPointer
* hideCursorPointer

function showPopup(map: MapboxGl.Map, e: React.SyntheticEvent<any>) {

        let event: MapboxGl.MapLayerMouseEvent = (e as any);

        if (event.features === undefined) {

            return;

        }

        let feature = event.features[0];

        let geometry: GeoJSON.Point = feature.geometry as GeoJSON.Point;

        let coordinates: GeoJSON.Position = geometry.coordinates.slice();

        // Ensure that if the map is zoomed out such that multiple

        // copies of the feature are visible, the popup appears

        // over the copy being pointed to.

        while (Math.abs(event.lngLat.lng - coordinates[0]) > 180) {

            coordinates[0] += event.lngLat.lng > coordinates[0] ? 360 : -360;

        }

        var description = '';

        var properties = feature.properties;

        if (properties !== null) {

            description = properties.description;

        }

        new MapboxGl.Popup()

            .setLngLat(coordinates as [number, number])

            .setHTML(description)

            .addTo(map);

    }

    function showCursorPointer(map: MapboxGl.Map) {

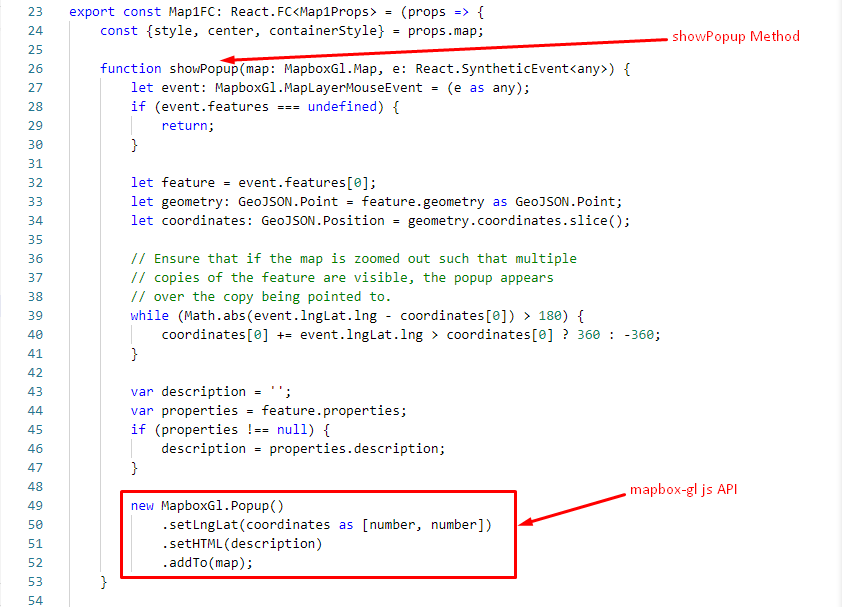
        map.getCanvas().style.cursor = 'pointer';

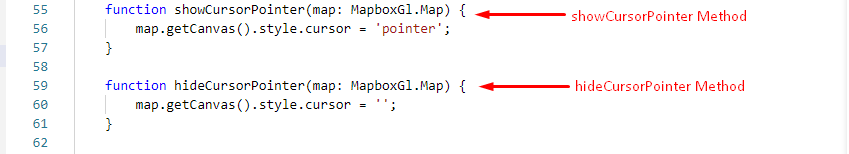
    }

    function hideCursorPointer(map: MapboxGl.Map) {

        map.getCanvas().style.cursor = '';

    }





**Step 1.2.4** Add below code inside Mapbox React Component as children.

<MapContext.Consumer>{(map) => {

    if (map === undefined) {

        return undefined;

    }

    return (

        <Layer

            id={LayerId.SYMBOL\_LAYER}

            type="symbol"

            layout={{

                'icon-image': 'harbor-15',

            }}

            onClick={(e: React.SyntheticEvent<any>) => showPopup(map, e)}

            onMouseEnter={() => showCursorPointer(map)}

            onMouseLeave={() => hideCursorPointer(map)}

        >

            <Feature coordinates={center} properties={{

                description: `<span><strong>Icon Name: </strong><span></span>harbor-15</span>`

            }}/>

        </Layer>

    );

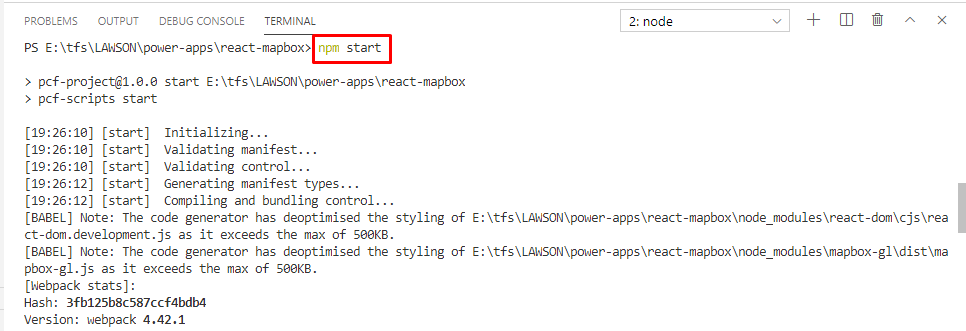
}}</MapContext.Consumer>

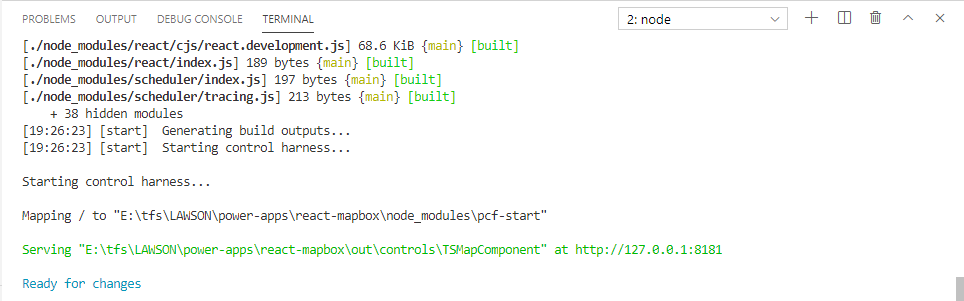


**Step 1.3**

Then run the below command and task get executed on your system as shown below.

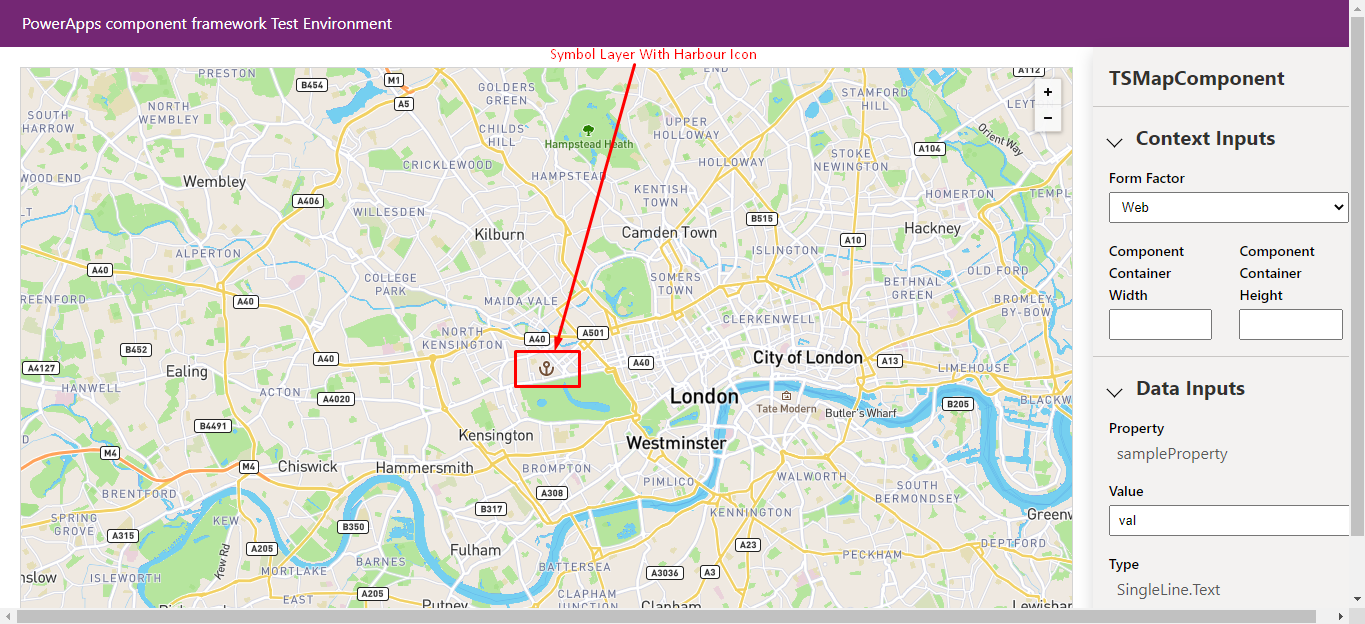
* npm start



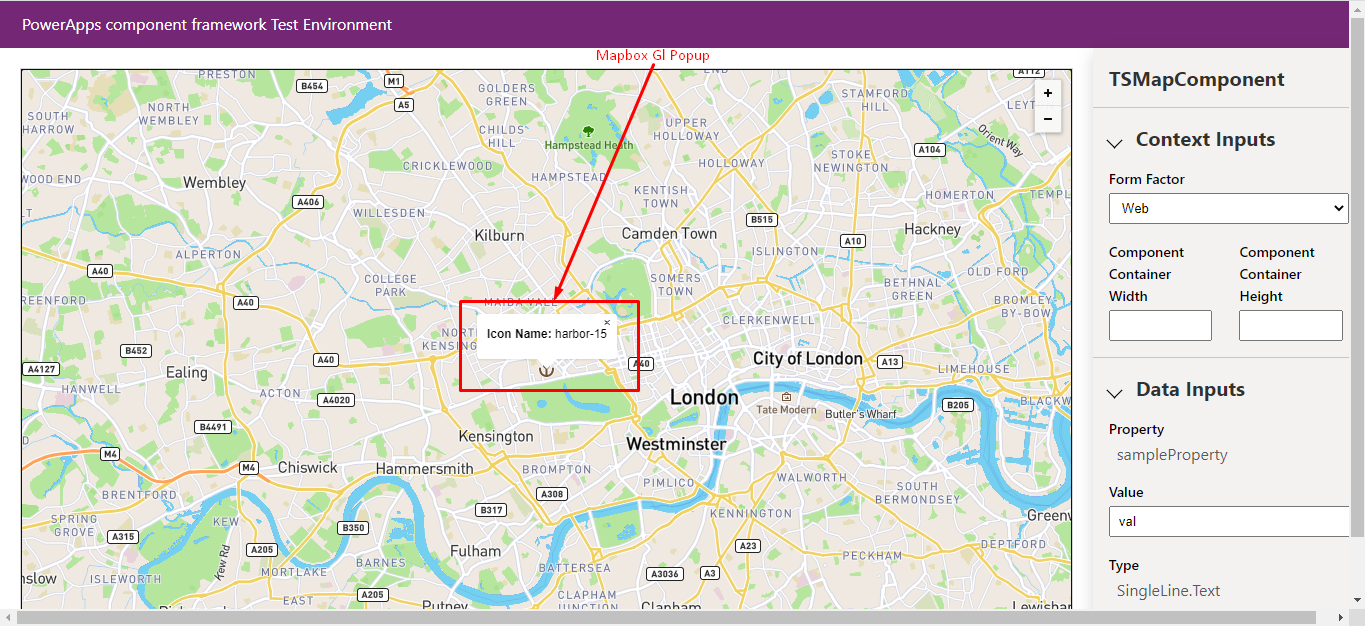


This opens a web browser to a local IP address, in my case [**http://127.0.0.1:8181/**](http://127.0.0.1:8181/)

Layer Screenshot:



Popup Screenshot:



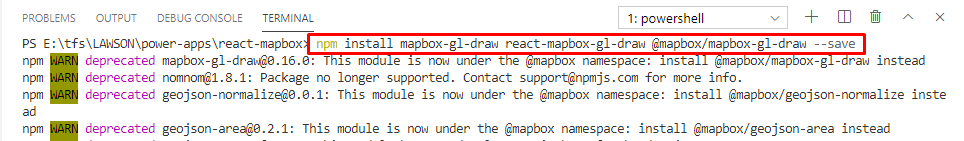
**2. Showing a Drawing Tool Control**

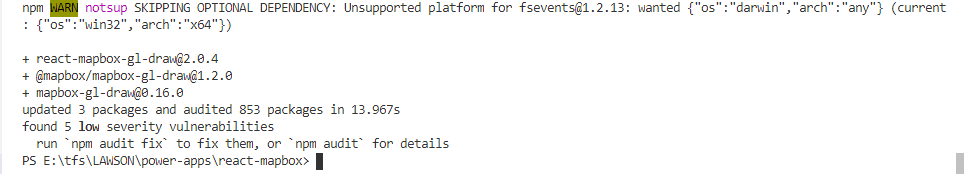
**Step 2.1**

Install the below dependencies

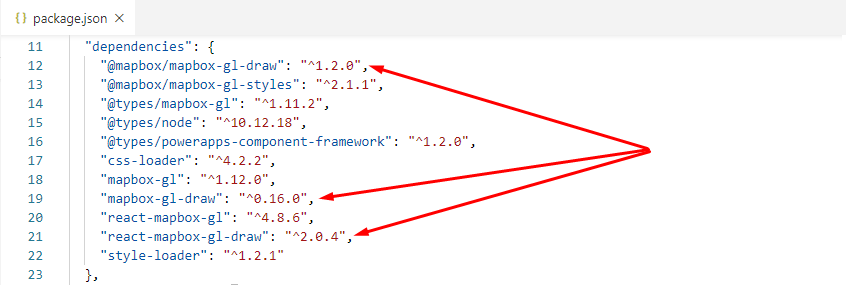
* mapbox-gl-draw
* react-mapbox-gl-draw
* @mapbox/mapbox-gl-draw

npm install mapbox-gl-draw react-mapbox-gl-draw @mapbox/mapbox-gl-draw --save





Installed Screeshot:

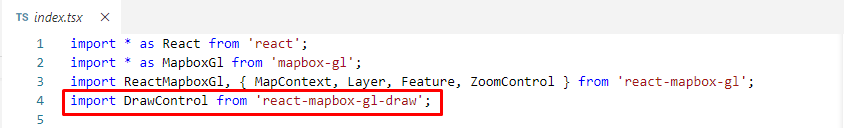


**Step 2.2**

Open *TSMapComponent/src/components/Map1/index.tsx* the file

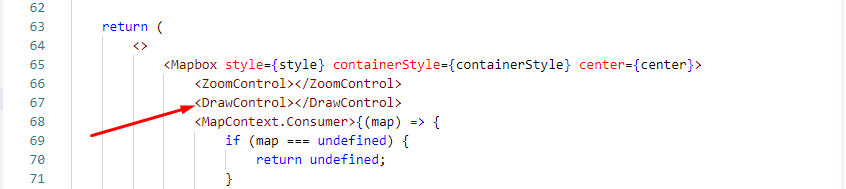
**Step 2.2.1** import **DrawControl** component from **react-mapbox-gl-draw**

import DrawControl from 'react-mapbox-gl-draw';



**Step 2.2.2** Add below code inside Mapbox React Component as children.

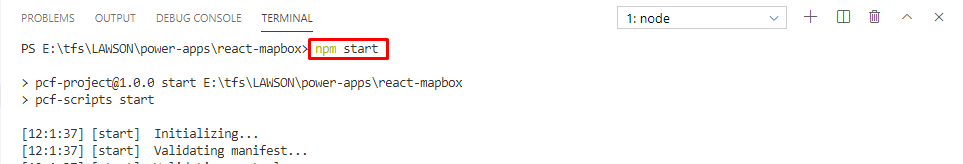
<DrawControl></DrawControl>



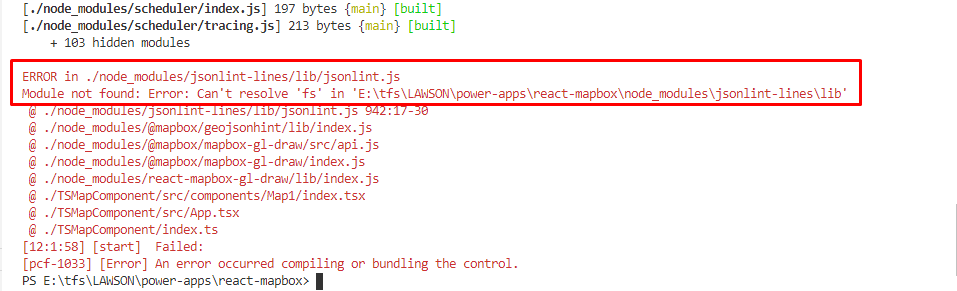
**Step 2.3**

Run the application using below command

* npm start



Get the below **error** after run this command.



**Step 2.4**

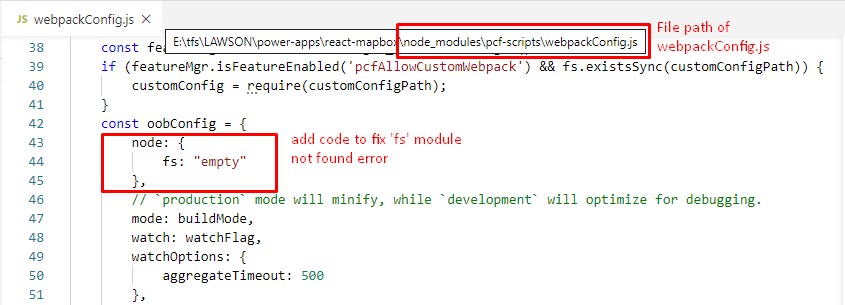
Resolve the **fs** module not found error issue into webpack config of pcf-scripts node package.

Add below lines into *node\_modules/pcf-scripts/webpackConfig.js* file

node: {

  fs: "empty"

},



**Step 2.5**

Now run the application using below command and started successfully.

* npm start

