# PHASE 4

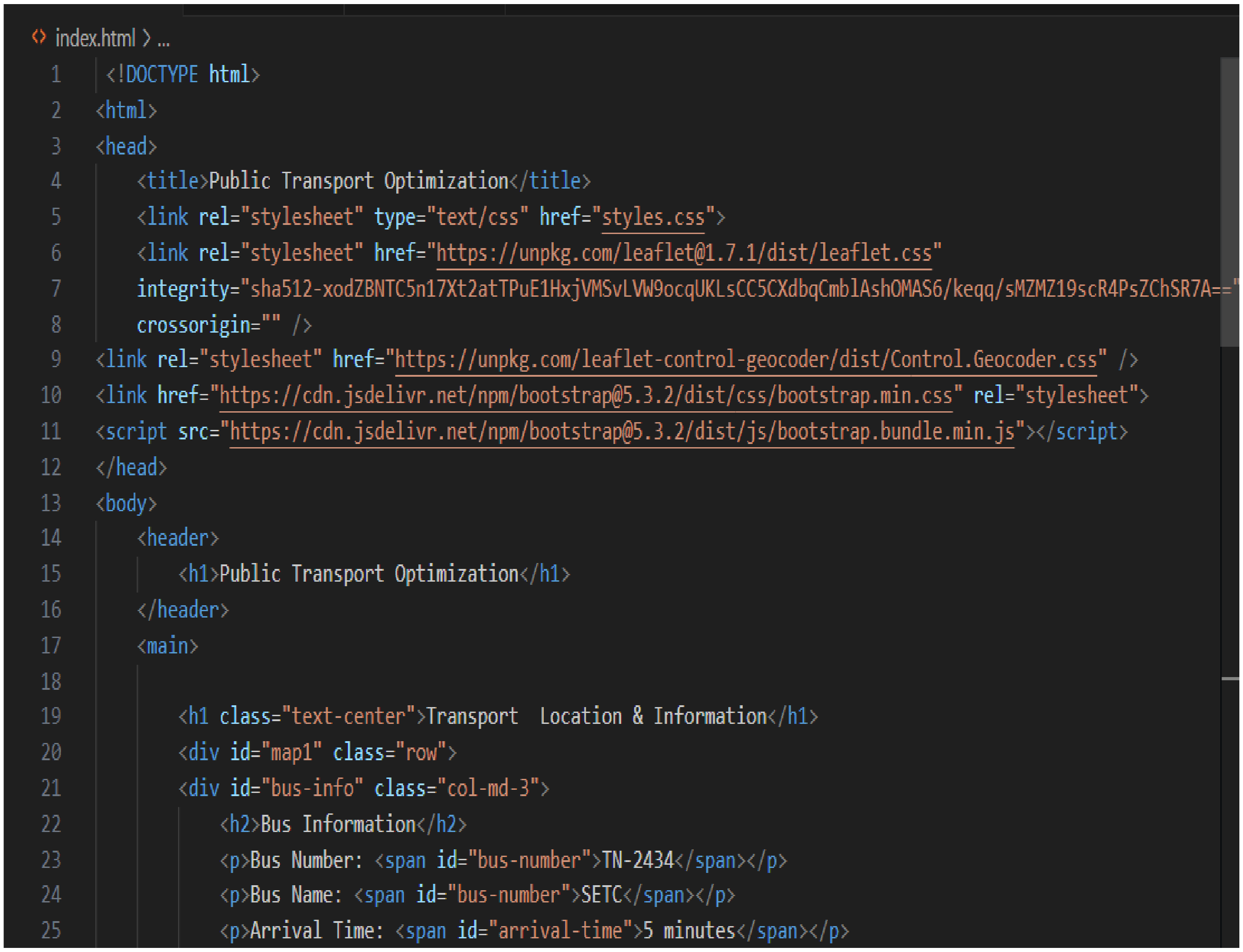
# DEVELOPMENT PART 2

# Public Transport Optimization

In this part you will continue building your project.

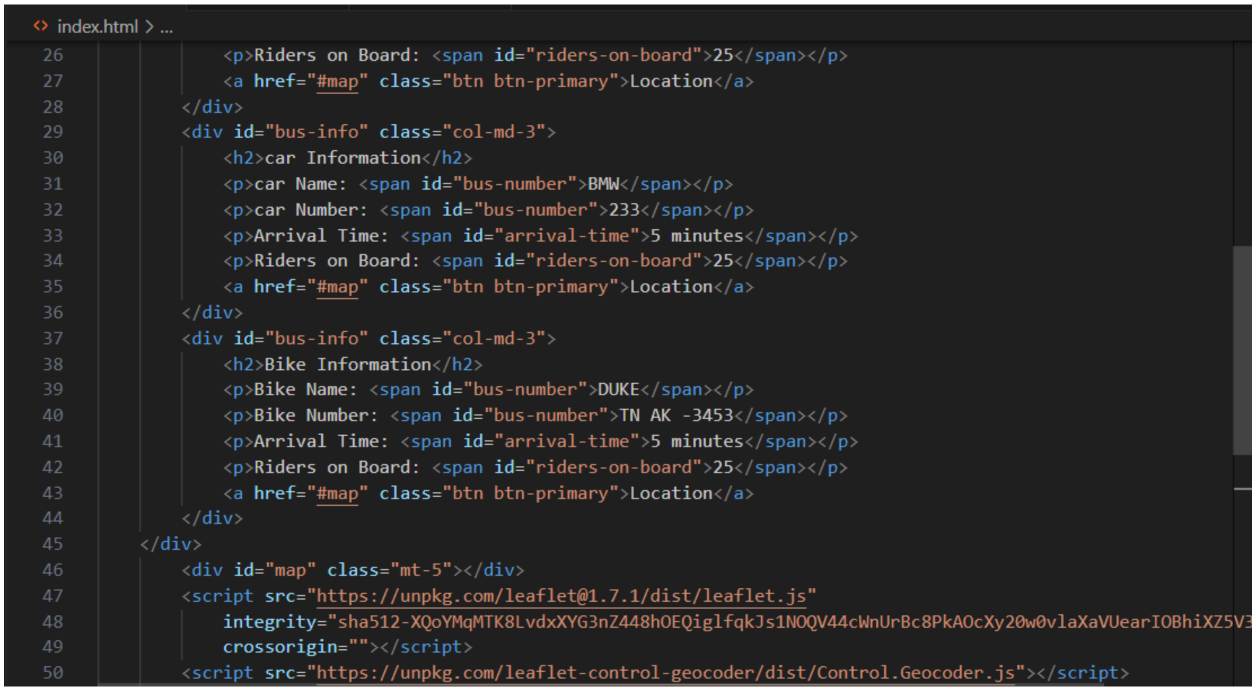
- Continue building the project by developing the real-time transit information platform.
- Use web development technologies (e.g., HTML, CSS, JavaScript) to create a platform that displays real-time transit information.
- Design the platform to receive and display real-time location, ridership, and arrival time data from IoT sensors.

## 1.index.html



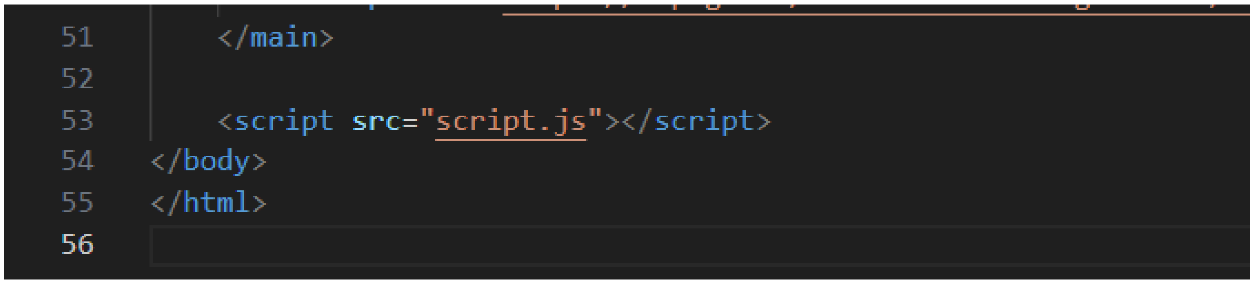
index.html > ...

```html
 <!DOCTYPE html>
<html>
<head>
    <title>Public Transport Optimization</title>
    <link rel="stylesheet" type="text/css" href="styles.css">
    <link rel="stylesheet" href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css"
    integrity="sha512-xodZBNTC5n17Xt2atTPuE1HxjVMSvLVW9ocqUKLsCC5CXdbqCmblAshOMAS6/keqq/sMZMZ19scR4PsZChSR7A=="
    crossorigin="" />
<link rel="stylesheet" href="https://unpkg.com/leaflet-control-geocoder/dist/Control.Geocoder.css" />
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"></script>
</head>
<body>
    <header>
        <h1>Public Transport Optimization</h1>
    </header>
    <main>

        <h1 class="text-center">Transport  Location & Information</h1>
        <div id="map1" class="row">
        <div id="bus-info" class="col-md-3">
            <h2>Bus Information</h2>
            <p>Bus Number: <span id="bus-number">TN-2434</span></p>
            <p>Bus Name: <span id="bus-number">SETC</span></p>
            <p>Arrival Time: <span id="arrival-time">5 minutes</span></p>
```



index.html > ...

```html
                <p>Riders on Board: <span id="riders-on-board">25</span></p>
                <a href="#map" class="btn btn-primary">Location</a>
            </div>
            <div id="bus-info" class="col-md-3">
                <h2>car Information</h2>
                <p>car Name: <span id="bus-number">BMW</span></p>
                <p>car Number: <span id="bus-number">233</span></p>
                <p>Arrival Time: <span id="arrival-time">5 minutes</span></p>
                <p>Riders on Board: <span id="riders-on-board">25</span></p>
                <a href="#map" class="btn btn-primary">Location</a>
            </div>
            <div id="bus-info" class="col-md-3">
                <h2>Bike Information</h2>
                <p>Bike Name: <span id="bus-number">DUKE</span></p>
                <p>Bike Number: <span id="bus-number">TN AK -3453</span></p>
                <p>Arrival Time: <span id="arrival-time">5 minutes</span></p>
                <p>Riders on Board: <span id="riders-on-board">25</span></p>
                <a href="#map" class="btn btn-primary">Location</a>
            </div>
        </div>
            <div id="map" class="mt-5"></div>
            <script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js"
                integrity="sha512-XQoYMqMTK8LvdxXYG3nZ448hOEQiglfqkJs1NOQV44cWnUrBc8PkAOcXy20w0vlaXaVUearIOBhiXZ5V3
                crossorigin=""></script>
            <script src="https://unpkg.com/leaflet-control-geocoder/dist/Control.Geocoder.js"></script>
```

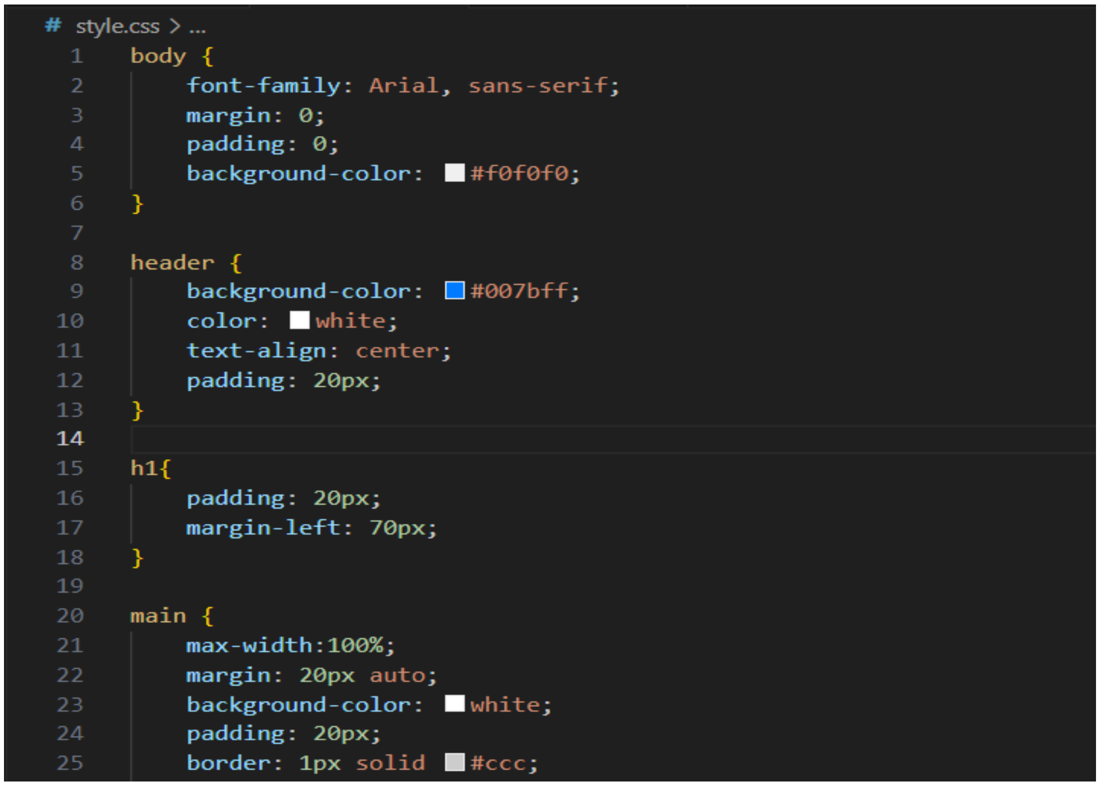


```html
    </main>

    <script src="script.js"></script>
</body>
</html>
```

1. **<!DOCTYPE html>**: This declaration specifies that the document follows HTML5 standards.
2. **<html>**: The root element of an HTML document.
3. **<head>**: This section contains metadata and links to external resources. In this case, it links to an external CSS file and sets the document's character encoding.
4. **<meta charset="UTF-8">**: Specifies the character encoding of the document as UTF-8, which is a widely used encoding for handling text in different languages.
5. **<title>Public Transport Ptimization Platform</title>**: Sets the title of the web page, which appears in the browser's title bar or tab.
6. **<link rel="stylesheet" type="text/css" href="styles.css">**: Links an external CSS file named "styles.css" to style the web page.
7. **<body>**: The main content of the web page is contained within the **<body>** element.
8. **<header>**: The header section typically contains the title or logo of the website. In this case, it displays the title "Environmental Monitoring Platform."

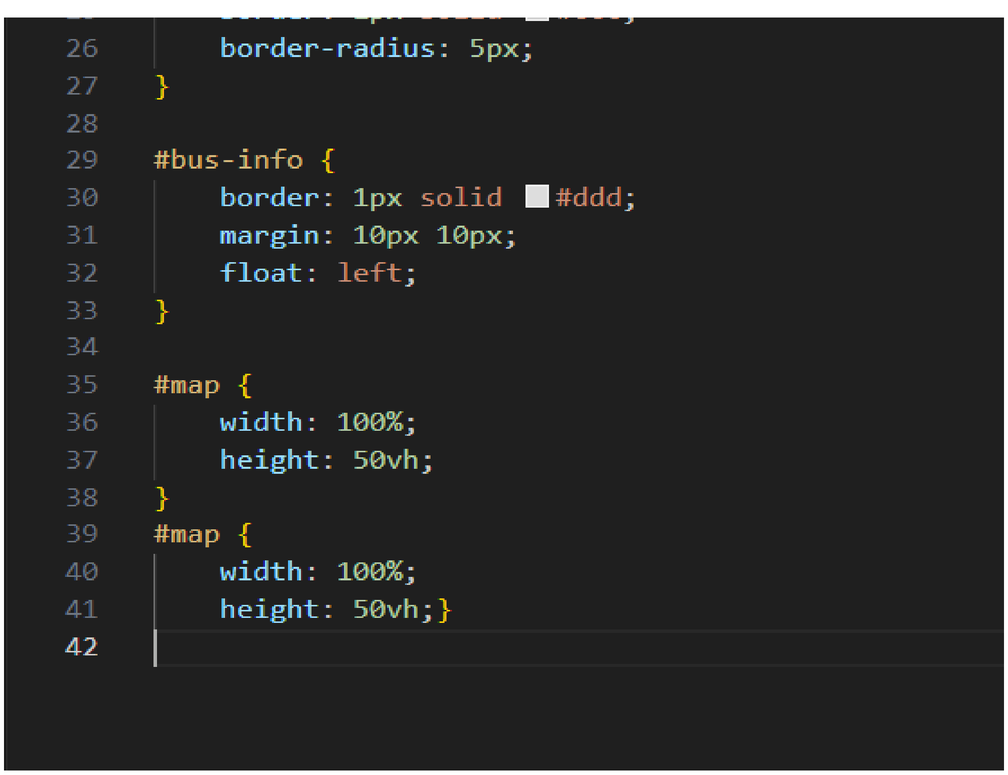
9. **<nav>**: This section contains navigation links. It's structured as an unordered list **<ul>** with list items **<li>**, each of which contains an anchor **<a>** element for navigation.
10. **<p align="center">**: This paragraph element aligns its text to the center. However, the **align** attribute is deprecated in HTML5, and it's recommended to use CSS for text alignment.
11. The **<p>** element provides information about environmental monitoring, its purpose, and what it encompasses. It's a description of the environmental monitoring concept.
12. **<section id="updates">**: This section is labeled "Latest Updates" and contains an unordered list **<ul>** of updates. Each update is presented as a list item **<li>** with a date and a description.
13. **<main>**: The main content of the web page, which contains various sections related to real-time data, data visualization, and information about the platform.
14. **<section class="sensor-data">**: This section is dedicated to displaying real-time sensor data. It contains two data items, "Temperature" and "Humidity," each with a heading **<h3>** and a placeholder paragraph **<p>** for data to be loaded via JavaScript.
15. **<section class="data-visualization">**: This section is for data visualization. It currently includes a canvas element **<canvas>** with the ID "chart" where data visualization elements like charts or graphs can be added. Additionally, it includes a script to include the Chart.js library for creating charts and another canvas element with the ID "lineChart."
16. **<section id="about">**: This section provides information about the platform, its mission, and what it offers.
17. **<section id="contact">**: This section offers contact information, including an email address where users can reach out with questions or feedback.
18. **<section id="disclaimer">**: This section includes a disclaimer regarding the informational nature of the platform and advises users to consult with experts for critical decisions related to the environment.
19. **<footer>**: The footer section displays a copyright notice for the year 2023, indicating ownership of the content.
20. **<script src="node.js"></script>**: This script element links to an external JavaScript file named "node.js." This file is used for real-time updates and functionality related to the platform.

## 2.style.css



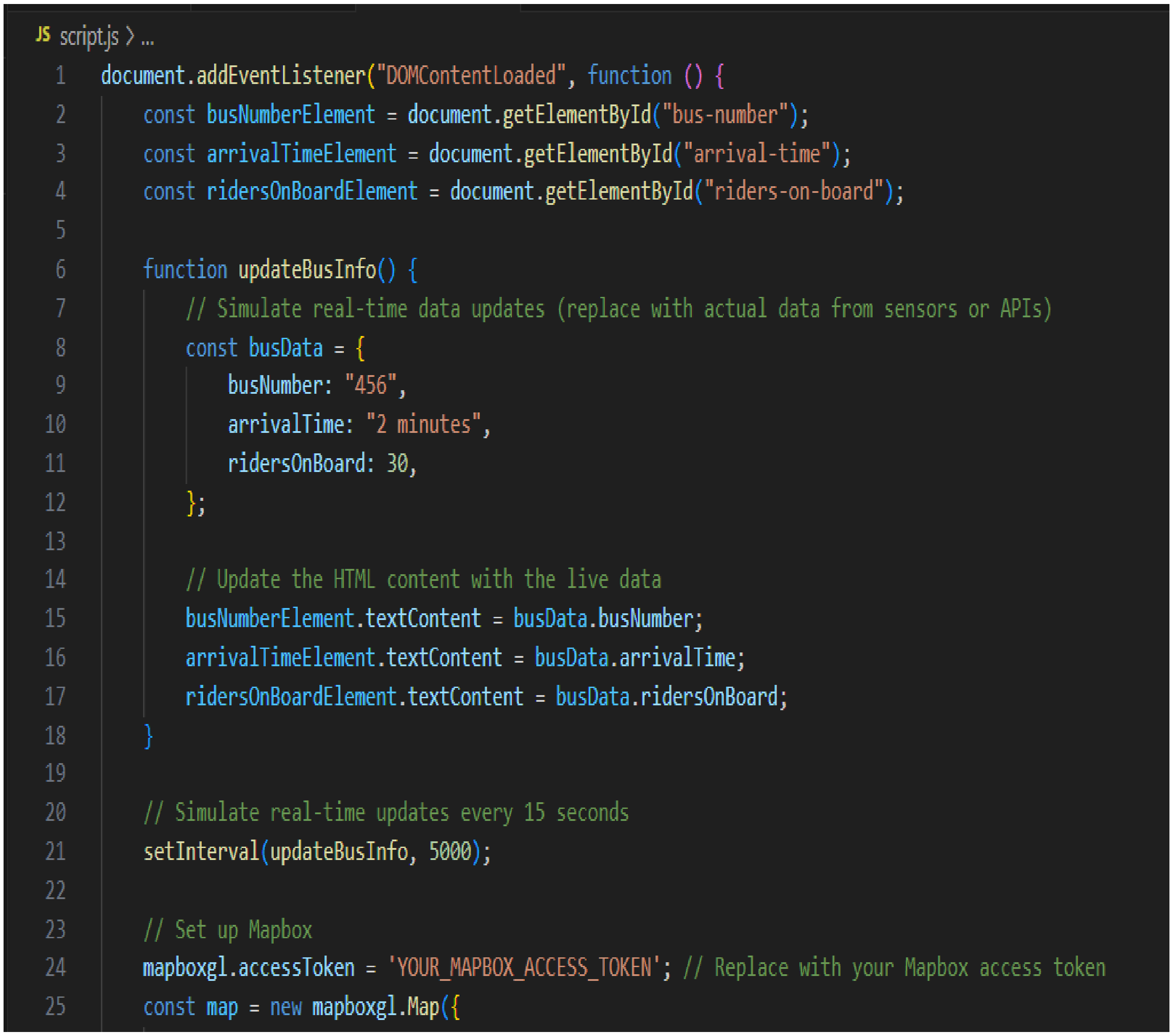
# style.css > ...

```css
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f0f0f0;
}

header {
    background-color: #007bff;
    color: white;
    text-align: center;
    padding: 20px;
}

h1{
    padding: 20px;
    margin-left: 70px;
}

main {
    max-width:100%;
    margin: 20px auto;
    background-color: white;
    padding: 20px;
    border: 1px solid #ccc;
```



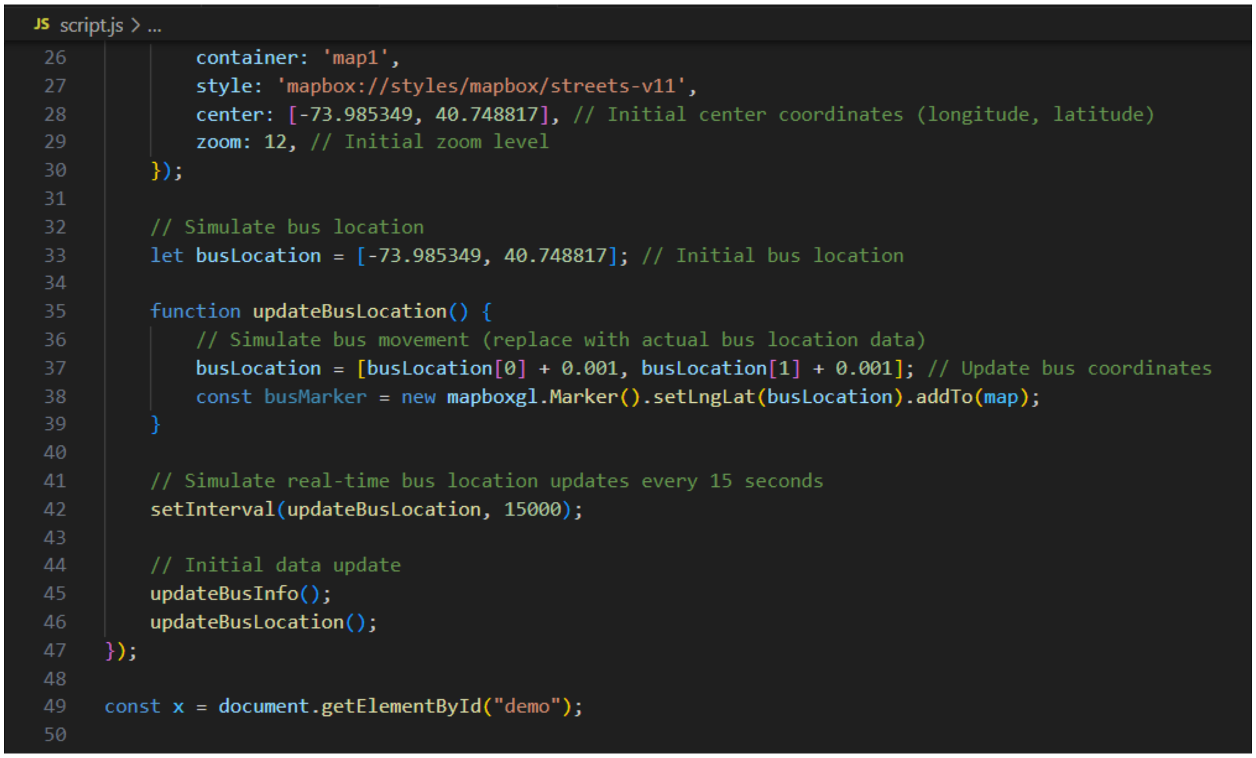
```css
    border-radius: 5px;
}

#bus-info {
    border: 1px solid #ddd;
    margin: 10px 10px;
    float: left;
}

#map {
    width: 100%;
    height: 50vh;
}
#map {
    width: 100%;
    height: 50vh;}

```

## 3.script.js

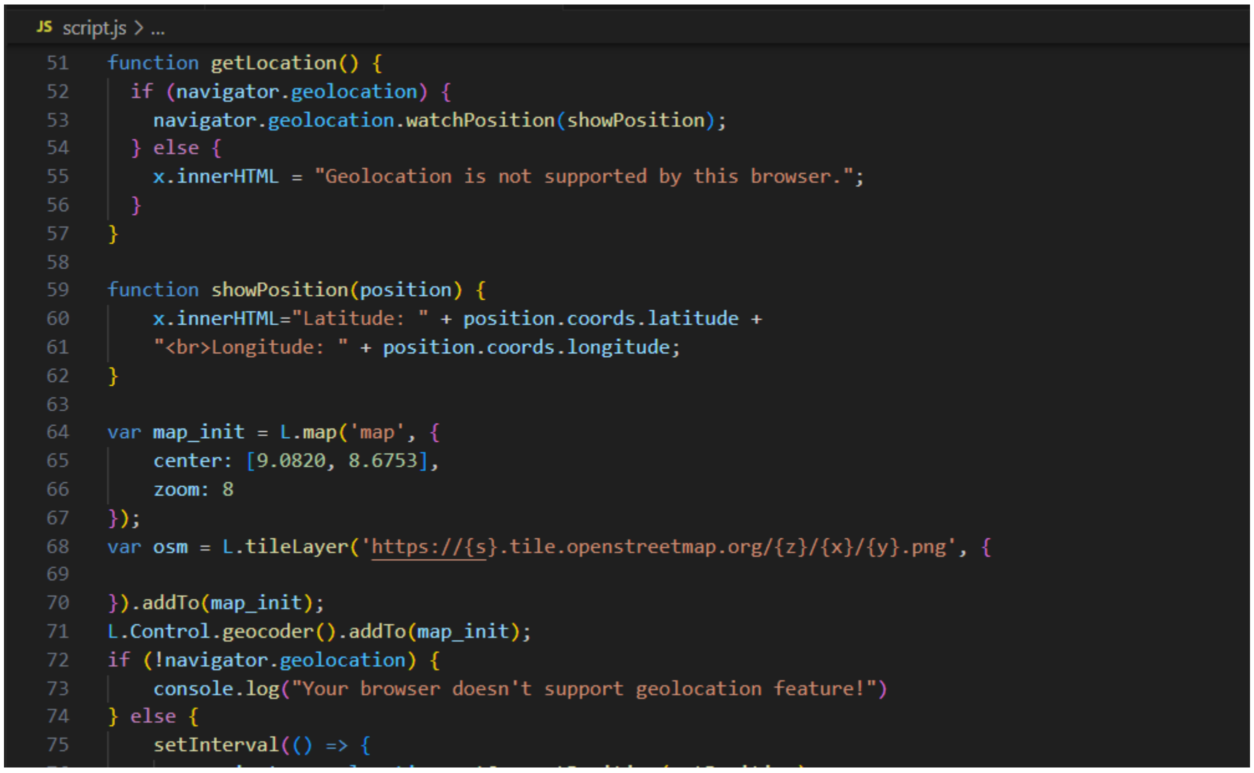


script.js > ...

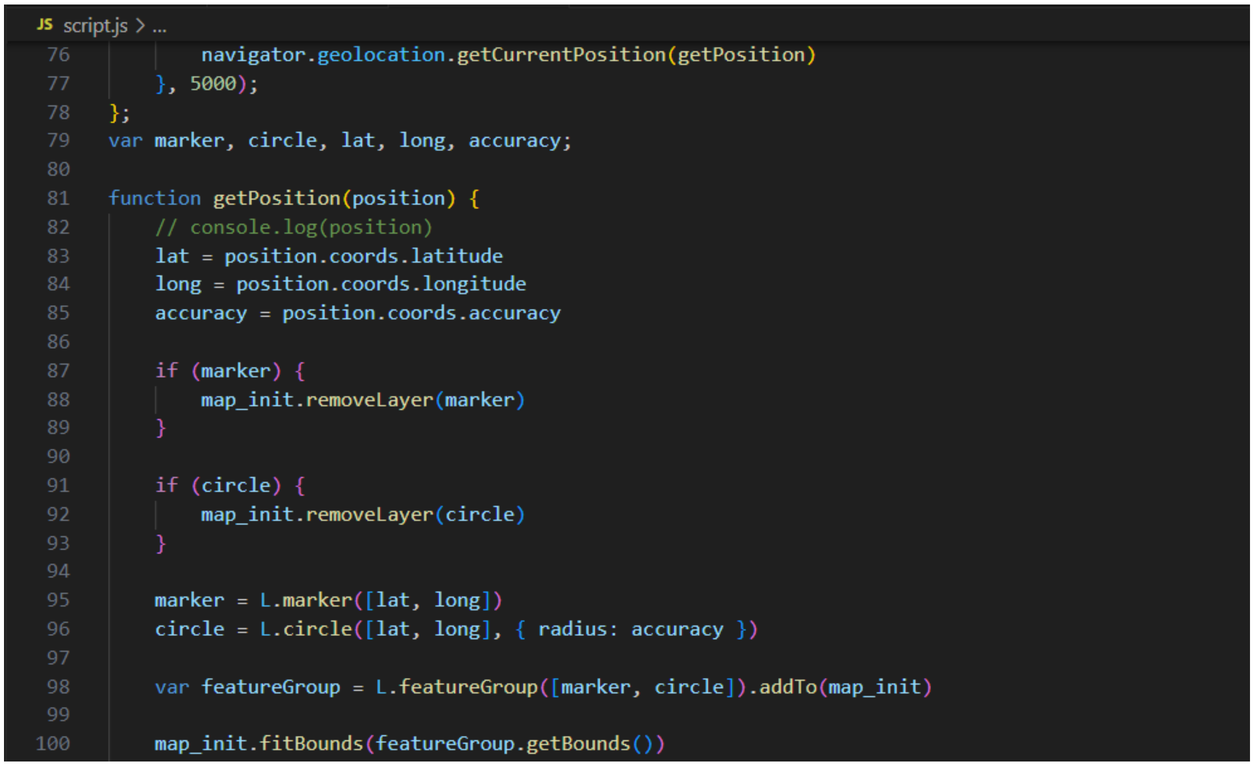
```javascript
document.addEventListener("DOMContentLoaded", function () {
    const busNumberElement = document.getElementById("bus-number");
    const arrivalTimeElement = document.getElementById("arrival-time");
    const ridersOnBoardElement = document.getElementById("riders-on-board");

    function updateBusInfo() {
        // Simulate real-time data updates (replace with actual data from sensors or APIs)
        const busData = {
            busNumber: "456",
            arrivalTime: "2 minutes",
            ridersOnBoard: 30,
        };

        // Update the HTML content with the live data
        busNumberElement.textContent = busData.busNumber;
        arrivalTimeElement.textContent = busData.arrivalTime;
        ridersOnBoardElement.textContent = busData.ridersOnBoard;
    }

    // Simulate real-time updates every 15 seconds
    setInterval(updateBusInfo, 5000);

    // Set up Mapbox
    mapboxgl.accessToken = 'YOUR_MAPBOX_ACCESS_TOKEN'; // Replace with your Mapbox access token
    const map = new mapboxgl.Map({
```


JS script.js > ...

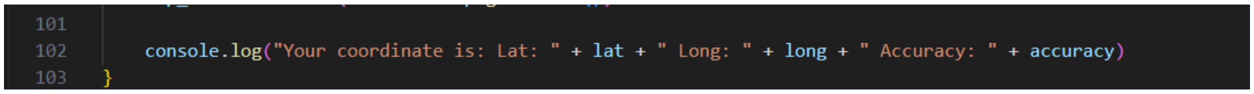
```javascript
            container: 'map1',
            style: 'mapbox://styles/mapbox/streets-v11',
            center: [-73.985349, 40.748817], // Initial center coordinates (longitude, latitude)
            zoom: 12, // Initial zoom level
        });

        // Simulate bus location
        let busLocation = [-73.985349, 40.748817]; // Initial bus location

        function updateBusLocation() {
            // Simulate bus movement (replace with actual bus location data)
            busLocation = [busLocation[0] + 0.001, busLocation[1] + 0.001]; // Update bus coordinates
            const busMarker = new mapboxgl.Marker().setLngLat(busLocation).addTo(map);
        }

        // Simulate real-time bus location updates every 15 seconds
        setInterval(updateBusLocation, 15000);

        // Initial data update
        updateBusInfo();
        updateBusLocation();
    });

    const x = document.getElementById("demo");

```


JS script.js > ...

```javascript
function getLocation() {
  if (navigator.geolocation) {
    navigator.geolocation.watchPosition(showPosition);
  } else {
    x.innerHTML = "Geolocation is not supported by this browser.";
  }
}

function showPosition(position) {
    x.innerHTML="Latitude: " + position.coords.latitude +
    "<br>Longitude: " + position.coords.longitude;
}

var map_init = L.map('map', {
    center: [9.0820, 8.6753],
    zoom: 8
});
var osm = L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {

}).addTo(map_init);
L.Control.geocoder().addTo(map_init);
if (!navigator.geolocation) {
    console.log("Your browser doesn't support geolocation feature!")
} else {
    setInterval(() => {
```



```javascript
        navigator.geolocation.getCurrentPosition(getPosition)
    }, 5000);
};
var marker, circle, lat, long, accuracy;

function getPosition(position) {
    // console.log(position)
    lat = position.coords.latitude
    long = position.coords.longitude
    accuracy = position.coords.accuracy

    if (marker) {
        map_init.removeLayer(marker)
    }

    if (circle) {
        map_init.removeLayer(circle)
    }

    marker = L.marker([lat, long])
    circle = L.circle([lat, long], { radius: accuracy })

    var featureGroup = L.featureGroup([marker, circle]).addTo(map_init)

    map_init.fitBounds(featureGroup.getBounds())
```



```javascript

    console.log("Your coordinate is: Lat: " + lat + " Long: " + long + " Accuracy: " + accuracy)
}
```

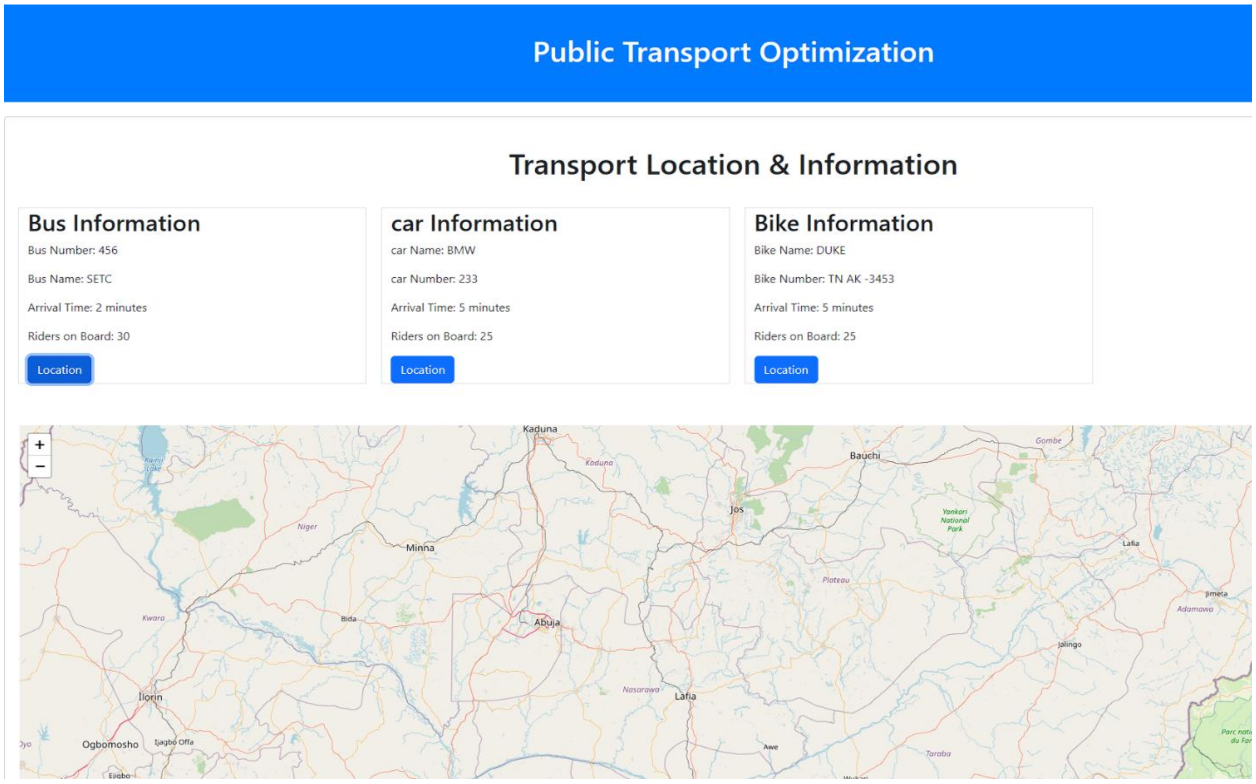
The provided JavaScript code contains functions to update sensor data, create a random chart for data visualization, and create a line chart using the Chart.js library. Let's break down the code and provide an explanation:

1. **updateSensorData Function**:
    - This function simulates the update of temperature and humidity values. In a real application, you would replace the random data with actual sensor data.
    - It generates random values for temperature and humidity within specified ranges.
    - It updates the HTML elements with the new values, specifically the elements with IDs "temperature" and "humidity."
2. **createRandomChart Function**:
    - This function creates a random chart for data visualization. Please note that in a real application, you would use a real charting library and actual data.
    - It utilizes the Chart.js library to create a line chart.

- The chart includes two datasets (Temperature and Humidity) with random data points.
- The chart's configuration includes labels, colors, and other properties.
- The chart is drawn on the canvas element with the ID "chart."

3. **Updating Data and Chart on an Interval**:
    - **setInterval** is used to repeatedly update sensor data and create a new random chart every 5 seconds (5000 milliseconds). You can adjust the interval to suit your needs.
    - The **updateSensorData** function and **createRandomChart** function are called within the interval to provide updated data and charts.
4. **Calling Update Functions on Page Load**:
    - To ensure that the sensor data and initial chart are displayed when the page loads, the **updateSensorData** and **createRandomChart** functions are called immediately after defining them.
5. **createLineChart Function**:
    - This function is responsible for creating a line chart using the Chart.js library. It is distinct from the **createRandomChart** function.
    - It defines the sample data for the line chart (temperature and humidity values for different months). In a real application, you would replace this with your actual data.
    - The function sets various chart properties, including labels, colors, and scale configurations.
6. **Calling createLineChart on Page Load**:
    - To ensure that the line chart is displayed when the document is ready, the **createLineChart** function is called when the "DOMContentLoaded" event is triggered.

## OUTPUT



Public Transport Optimization

Transport Location & Information

Bus Information

Bus Number: 456

Bus Name: SETC

Arrival Time: 2 minutes

Riders on Board: 30

car Information

car Name: BMW

car Number: 233

Arrival Time: 5 minutes

Riders on Board: 25

Bike Information

Bike Name: DUKE

Bike Number: TN AK -3453

Arrival Time: 5 minutes

Riders on Board: 25