Smart Cities Hackathon Tier 2 Challenge 6

Tier 2: Challenge 6

Tier 2 Challenge 6: HTML API Data Challenge

This challenge will see you creating a new webpage, making a javascript call to retrieve a sensor reading, and using the design assets provided to add a header and make a dynamic thermometer reading. You will then use a third-party chart library to plot readings for the last 24 hours.

Step 1

The assets for this challenge can be downloaded from: https://github.com/aql-com/iot_event
Download the Thermometer.svg file (from Hackathon Materials -> Smart Cities -> Design Assets -> Individual Graphics) and store it, keeping the path to the file for later.

Step 2

Log into the aql Core IoT platform (https://core.aql.com/login) with your provided event credentials.

You can use the sensor you created in Challenge 1 or select a temperature sensor from one of the demo sensors included. Note down the SensorID as we will need this later for the API.

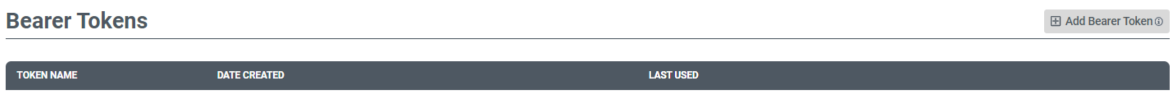
If you created a bearer token in Challenge 5, please jump to Step 5.

Step 3

Select the settings option in the bottom left of the nav bar.

Step 4

Create a new bearer token and allocate a name for this token.



Copy the token as you will need this for the JavaScript later.

Step 5

Create a new folder for your project.

Inside the folder, create an HTML file (e.g., `index.html`) and a JavaScript file (e.g., `script.js`).

Step 6

Open the folder in your chosen editor. Visual studio code is free to download from https://code.visualstudio.com/download

In `index.html`, add HTML structure for your webpage, including elements for displaying the thermometer and temperature details - a sample for this can be seen below.

Ensure to link your JavaScript file to the HTML file using a `<script>` tag.

Add the event logos and your own colours to the html. Be as creative as you would like at this stage - you can ask the Hackathon AI to help you with all these steps.

Sample:
You'll need to replace 'path_to_this_asset' in your thermometer image source with the path to from this file to the Thermometer.svg you downloaded earlier. You can check this path works directly in the VS Code window.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Thermometer Web Page</title>
  <link rel="stylesheet" href="styles.css"> <!-- You can link a CSS
file for styling -->
</head>
<body>

  <div class="container">
    <h1>Temperature Monitoring</h1>
    <div class="thermometer-wrapper">
      <!-- Embed the SVG file using the img tag -->
      <img id="thermometer" src="path_to_this_asset/Thermometer.svg"
alt="Thermometer" width="200" height="400">
      <div id="thermometer-level">
        <!-- Thermometer body -->
        <div id="thermometer-background"></div>
        <!-- Thermometer level -->
        <div id="level"  height="0"></div>
      </div>
    </div>
    <div id="temperature"></div> <!-- Temperature details will be 
displayed here -->
  </div>

  <script src="script.js"></script> <!-- Link your JavaScript file -->
</body>
</html>
```

Step 7

The HTML above requires CSS to position the live reading over the SVG. You can experiment and add further CSS to add your style to the page.

Add the CSS file “styles.css”;

```
.thermometer-wrapper {
    height: 400px;
    width: 200px;
    position: relative;
}

#thermometer {
    position: absolute;
}

#thermometer-level {
    position: absolute;
    height: 270px;
    width: 20px;
    left: 85px;
    top: 24px;
}

#thermometer-background {
    position: absolute;
    top: 0;
    left: 0;
    height: 100%;
    width: 20px;
    background: grey;
}

#level {
    position: absolute;
    bottom: 0px;
    left: 0;
    width: 20px;
    background: red;
}
```

Step 8

Next, we will add the JavaScript to return the latest sensor reading, you will need to replace the Sensor ID and the Bearer token. This script makes use of api.core.com. The method

```
https://api.core.aql.com/v1/sensors/${sensorid}/sensor-data/latest
```

Returns the latest reading for a single sensor. There are methods to send an array of sensors which will allow you to load multiple sensors to the UI. Review the documentation and extend the Javascript to use this method or use the sample below.

```
// When the DOM content is loaded
document.addEventListener("DOMContentLoaded", function() {
    // API endpoint URL
    let sensorid = "REPLACE WITH YOUR SENSOR ID";
    const apiUrl =
`https://api.core.aql.com/v1/sensors/${sensorid}/sensor-data/latest`;

    // Bearer token
    const bearerToken = "REPLACE WITH YOUR BEARER TOKEN";

    // Fetch temperature data from the API
    fetch(apiUrl, {
        method: "GET",
        headers: {
            "Authorization": `Bearer ${bearerToken}`
        }
    })
    .then(response => response.json())
    .then(data => {
        // Get the temperature value from the API response

        const temperature = data.value;

        // Map temperature to thermometer height
        const levelHeight = mapTemperatureToHeight(temperature);

        // Update thermometer level based on temperature
        document.getElementById("level").style.height =
`${levelHeight}px`;


    })
    .catch(error => console.error("Error fetching temperature:",
error));

});


// Function to map temperature to thermometer height
function mapTemperatureToHeight(temperature) {
    const minTemperature = 0;
    const maxTemperature = 100;
    const minHeight = 0;
    const maxHeight = 270;

    const normalizedTemperature = Math.max(Math.min(temperature,
maxTemperature), minTemperature);
    const heightPercentage = normalizedTemperature / maxTemperature;
    const levelHeight = maxHeight * heightPercentage;

    return levelHeight;
}
```

Save the file and open the index.html file from your file directory into a browser window. Your value should appear in the visual thermometer.

If there is an offset in the visual thermometer presentation, ask your Tier 2 AI Assistant for another copy of the styles.css file and check there are no omissions or differences from your current one.

Tier 2: Challenge 6

If you right select on the browser and select inspect any JavaScript errors returned will be displayed there. If you have any problems, you can provide your code and error to your Tier 2 AI Assistant and it will be able to help.

You can use the curl statement from challenge 6 to add readings that change the UI. Upload readings and hit refresh and your UI should change to reflect this latest value.