Smart Cities Hackathon Tier 2 Challenge 7

Tier 2: Challenge 7

Tier 2 Challenge 7: 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.

The assets for this challenge can be downloaded from: https://github.com/aql-com/iot_event

Step 1

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 Tier 2 Challenge 6 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.

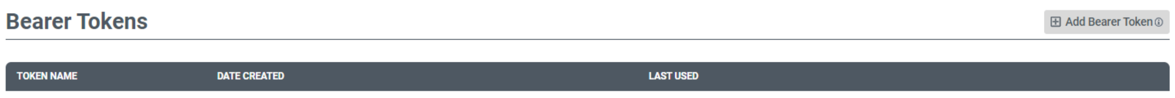
If you created a bearer token in Challenge 6, please jump to Step 4.

Step 2

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

Step 3

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


Bearer Tokens

TOKEN NAME	DATE CREATED	LAST USED

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

Step 4

Choose a location on your computer for the project files for example c:\code

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 5

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

Open `index.html` in a text editor.

Add HTML structure for your webpage, including elements for displaying the thermometer and temperature details.

Link your JavaScript file (`script.js`) to the HTML file using `<script>` tag.
Add the event logos and your own colours to the html. A sample can be seen below. Be as creative as you would like at this stage. You can ask the Hackathon AI to help you with all these steps.

```
<!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="../Design Assets/Individual Sensor
Graphics/Thermometer/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 6

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

```
.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 7

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 in a browser window. Your value should appear in the visual thermometer. 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.

Ways to extend this step

Extend the HTML to show the value as a reading below the thermometer, or extend to include other readings such as humidity or any other air quality sensor.

Tier 2: Challenge 7

Step 8

Once you have the latest value showing, you are going to add a charting element. First, you will need to update the HTML to add the chart JS and element for the chart.

```html
<!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="../Design Assets/Individual Sensor
Graphics/Thermometer/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>

  <canvas id="chart-wrap"></canvas> <!-- Chart will be displayed here
-->

  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
  <script src="script.js"></script> <!-- Link your JavaScript file -->
</body>
</html>
```

Step 9

Next, you need to extend the JavaScript to populate the chart with the readings for today. Review the code below and add the changes to your existing Javascript code

```
// 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 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));

    // Fetch chart data
    getChartData(bearerToken, sensorid);
});

// 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;
}

// Function to fetch chart data
const getChartData = (bearerToken, sensorid) => {
    const apiUrl =
`https://api.core.aql.com/v1/sensors/${sensorid}/sensor-data/aggregate/
ave`;
    const today = new Date().toISOString().slice(0, 10);
    const body = JSON.stringify({
        "startDate": `${today} 00:00:00`,
        "endDate": `${today} 23:59:59`,
        "granularity": "quarter_hour"
    });

    // Fetch chart data from the API
    fetch(apiUrl, {
        method: "POST",
        headers: {
            "Accept": "application/json",
            "Authorization":`Bearer ${bearerToken}`,
            "Content-Type": "application/json"
        },
        body: body
    })
    .then(response => {
        if (response.ok) {
            return response.json();
        } else {
            throw new Error("Failed to fetch chart data");
        }
    })
    .then(data => {
        processGraphData(data);
    })
    .catch(error => console.error("Error fetching chart data:",
error));
}


// Function to process graph data
const processGraphData = apidata => {
    console.log(apidata);

    if (Object.keys(apidata).length > 0) {
        let graphLabels = [];
        let graphValues = [];

        // Extract graph labels and values from the API response
        for (let i = 0; i < apidata.length; i++) {
            graphLabels.push(apidata[i].sensorReadingDate.substring(11,
16));
            graphValues.push(apidata[i].Average);
        }

        // Create a line chart using Chart.js
        new Chart(
            document.getElementById('chart-wrap'),
            {
                type: 'line',
                data: {
                    labels: graphLabels,
                    datasets: [
                        {
                            label: 'Temperature over time',
                            data: graphValues,
                            borderColor: '#fab400',
                            backgroundColor: '#fab400',
                        }
                    ]
                }
                }
            );
        }
}
```

Step 10

Save all the files and refresh your browser and you should see a sensor reading and a line graph of reading for today's values. If you right click the browser and select inspect and select console any JavaScript errors will be displayed here. Any problems can be debugged using the Hackathon AI.

Ways to extend this step

Extend the chart for several days, include other sensor readings, adjust the colour, Use other node charts to plot a scatter or bar chart. With HTML, Javascript and the https://api.core.aql.com/doc/ you can create rich visualisations. Get creative, we're all excited to see what you can do!