**Code Inspection for Sensor Data Visualization**

**Description of Landing Page:** The landing page visualizes real-time and historical sensor data (temperature in °C) for a selected sensor and line. The page fetches live and historical data from an API and displays them in three charts: a line chart for the sensor's temperature over time, a scatter chart, and an area chart. Additionally, the page shows various statistics like the latest temperature, the mean, min, and max temperature.

**Layout Description:**

* **Fonts:**
  + The font style is not explicitly defined in the provided code. Assuming defaults are used in the HTML/CSS structure for the page.
* **Colours:**
  + Colors are not directly referenced in the JavaScript code. These might be defined in the accompanying CSS file.
* **Text and Text Styling:**
  + sensorTitle dynamically updates the title on the page to display the selected sensor and line.
  + Text elements such as temperatureDisplay, meanTemperature, minTemperature, and maxTemperature display temperature statistics.

**Images, Logos, and Other Visual Elements:**

* **No images or logos** are directly referenced in the JavaScript code, but dynamic data from the sensor could be visualized in the charts.

**Links:**

* **Links to Other HTML/CSS/JS Pages:**
  + The code does not contain links to other pages but makes API calls to fetch data.
* **Bootstrap Links:**
  + No direct links to external libraries like Bootstrap are shown in the provided JavaScript. These may be linked elsewhere in the HTML or CSS.

**Extra Details – Interactive Elements:**

* **Charts:**
  + The page uses interactive charts powered by the **ApexCharts** library.
    - **Line chart** (sensorChart): Displays temperature data over time.
    - **Scatter chart** (sensorScatterChart): Displays temperature data points as a scatter plot.
    - **Area chart** (sensorAreaChart): Shows temperature data with a gradient-filled area chart.
  + **Zooming and panning**: All three charts have zoom enabled along the x-axis (time), allowing users to explore the data interactively.
  + **Live data fetch**: The charts update every 8 seconds with new data from the API.

**JavaScript Explanation and Application:**

* **Base Styles:**
  + The JavaScript does not define base styles. Presumably, styling is handled elsewhere, likely in the associated CSS file.
* **Event Listeners:**
  + DOMContentLoaded event listener ensures the DOM is fully loaded before running the script, including rendering charts and fetching data.
* **Variables:**
  + selectedLine and selectedSensor: Track which sensor and line are currently selected.
  + historicalData and liveData: Arrays to store historical and real-time data, respectively.
  + isFirstLiveDataFetch: Flag to track if the live data is being fetched for the first time.
  + sensorChart, sensorScatterChart, sensorAreaChart: Instances of ApexCharts used for rendering the data charts.
  + initialTimeRange: Defines the initial time range for chart data visualization.
* **Chart Initialization:**
  + **Line Chart (sensorChart)**:
    - A line chart is set up to plot the temperature over time.
    - The chart uses a datetime x-axis and a Temperature (°C) y-axis.
    - Zoom is enabled along the x-axis, with a range of 3,500,000 milliseconds (roughly 58 minutes).
  + **Scatter Chart (sensorScatterChart)**:
    - A scatter chart is initialized with a datetime x-axis and Temperature (°C) y-axis.
    - Markers are red (#FF4560) and are sized 5px.
  + **Area Chart (sensorAreaChart)**:
    - The area chart has a gradient fill with opacity variations.
    - It also uses a datetime x-axis and a Temperature (°C) y-axis.
* **Fetching Data:**
  + **Live Data Fetch (fetchLiveData)**: Fetches real-time data every 8 seconds from the API (localhost:5000/api/live-data/{selectedLine}). The data is processed and appended to the live data array, updating the main line chart and secondary charts.
  + **Historical Data Fetch (fetchHistoricalData)**: Fetches historical data for the selected sensor and line from the API (localhost:5000/api/historical/{line}/{sensor}) and initializes the charts with this data.
* **Combining Historical and Live Data:**
  + On the first live data fetch, historical and live data are combined, sorted, and displayed on the main chart.
  + For subsequent updates, new data points are appended to the chart without reloading the entire dataset.
* **Data Caching:**
  + Data is cached in localStorage to preserve the data across page reloads. This helps improve the performance by reusing previously fetched data.
* **Updating Charts:**
  + **updateSecondaryCharts(data)** updates both the scatter and area charts with the combined data.
  + **updateStatistics(data)** calculates and displays the latest temperature, mean, min, and max temperatures.

**Summary of Key Features:**

* Interactive line, scatter, and area charts for visualizing temperature data.
* Real-time updates every 8 seconds via live data fetch.
* Historical data from an API to display past temperature readings.
* Caching live data for better performance and offline use.
* Dynamic update of statistics (latest, mean, min, max temperature).