---

1. API URL and Key
```javascript
apiUrl = "https://api.openweathermap.org/data/2.5/weather?units=metric&q=London";
apiKey = "5e7c075bba3fd5b9e5ef9565459626f1";
```
- **`apiUrl`**: This is the base URL for the OpenWeatherMap API. It includes:
  - `units=metric`: Ensures the temperature is returned in Celsius.
  - `q=London`: Specifies the default city (London) for which weather data is fetched.
- **`apiKey`**: This is your unique API key required to authenticate requests to the OpenWeatherMap API.

---

2. DOM Element Selection
```javascript
const btn = document.querySelector(".search-section button");
const icon = document.querySelector(".weather-icon");
```
- **`btn`**: Selects the button in the `.search-section` that triggers the weather data fetch.
- **`icon`**: Selects the `<img>` element where the weather icon will be displayed.

---

3. Fetching Weather Data
```javascript
async function getWeatherData() {
    try {
        const response = await fetch(apiUrl + `&appid=${apiKey}`);

        if (!response.ok) {
            throw new Error(`HTTP error! Status: ${response.status}`);
        }

        const data = await response.json();
        console.log(data);
```
- **`getWeatherData`**: This is an asynchronous function that fetches weather data from the API.
- **`fetch`**: Sends a request to the API URL with the `apiKey` appended for authentication.
- **Error Handling**: If the response is not successful (e.g., city not found or server error), an error is thrown.
- **`data`**: The response from the API is parsed as JSON and stored in the `data` variable.

---

4. Updating the DOM with Weather Data
```javascript
        document.querySelector(".temp").innerHTML = Math.round(data.main.temp) + "°C";
        document.querySelector(".city").innerHTML = data.name;
        document.querySelector(".wind").innerHTML = data.wind.speed + " km/h";
        document.querySelector(".humidity").innerHTML = data.main.humidity + " %";
```
- **Temperature**: The temperature is rounded to the nearest integer and displayed in the `.temp` element.
- **City Name**: The city name is displayed in the `.city` element.
- **Wind Speed**: The wind speed is displayed in the `.wind` element.
- **Humidity**: The humidity percentage is displayed in the `.humidity` element.

---

5. Updating the Weather Icon
```javascript
        if (data.weather[0].main == "Clouds") {
            icon.src = "clouds.png";
        } else if (data.weather[0].main == "Rain") {
            icon.src = "rain.png";
        } else if (data.weather[0].main == "Clear") {
            icon.src = "clear.png";
        } else if (data.weather[0].main == "Drizzle") {
            icon.src = "drizzle.png";
        } else if (data.weather[0].main == "Mist") {
            icon.src = "mist.png";
        } else if (data.weather[0].main == "Snow") {
            icon.src = "snow.png";
        }
```
- The `data.weather[0].main` property contains the main weather condition (e.g., "Clouds", "Rain").
- Based on the condition, the `src` attribute of the `icon` element is updated to display the corresponding weather icon (e.g., `clouds.png`, `rain.png`).

---

6. Error Handling
```javascript
    } catch (error) {
        console.error("Error fetching weather data:", error);
        alert("Failed to fetch weather data. Please check the city name and try again.");
    }
}
```
- If an error occurs during the fetch operation (e.g., network issue, invalid city name), it is caught in the `catch` block.
- The error is logged to the console, and an alert is shown to the user.

---

7. Handling the Search Button Click
```javascript
btn.addEventListener("click", () => {
    const searchValue = document.querySelector(".search-section input").value;
    apiUrl = `https://api.openweathermap.org/data/2.5/weather?units=metric&q=${searchValue}`;
    getWeatherData();
});
```
- When the user clicks the search button:
  - The value entered in the search input field is retrieved.
  - The `apiUrl` is updated with the new city name.
  - The `getWeatherData` function is called to fetch and display the weather data for the new city.

How It Works
1. The user enters a city name in the search input field.
2. When the search button is clicked, the `apiUrl` is updated with the new city name.
3. The `getWeatherData` function fetches weather data from the OpenWeatherMap API.
4. The fetched data is used to update the DOM with the temperature, city name, wind speed, humidity, and weather icon.
5. If the city is not found or an error occurs, an alert is shown to the user.

Example API Response
The API response for a city like "London" might look like this:
```json
{
  "coord": { "lon": -0.1257, "lat": 51.5085 },
  "weather": [{ "id": 800, "main": "Clear", "description": "clear sky", "icon": "01d" }],
  "main": { "temp": 15.5, "feels_like": 14.8, "temp_min": 14.0, "temp_max": 17.0, "pressure": 1015,
"humidity": 72 },
  "wind": { "speed": 3.6, "deg": 200 },
  "name": "London"
}
```
- **`main.temp`**: Temperature in Celsius.
- **`name`**: City name.
- **`wind.speed`**: Wind speed in meters per second.
- **`main.humidity`**: Humidity percentage.
- **`weather[0].main`**: Main weather condition (e.g., "Clear").

Improvements
1. **Input Validation**: Ensure the user enters a valid city name.
2. **Loading State**: Show a loading spinner while fetching data.
3. **Default City**: Fetch weather data for the user's current location by default.
4. **Error Messages**: Provide more specific error messages (e.g., "City not found").
5. **Responsive Design**: Make the UI responsive for different screen sizes.