**Project Weather Information Page**

* ***Presentation Title: Weather Information Page “HTML Structure”***

#### **Slide1: Introduction**

* **Title:** Overview of the Weather Information Page
* **Content:**
  + A simple, user-friendly webpage to display weather information for any city.
  + Built using HTML for structure, CSS for styling, and JavaScript for dynamic functionality.
  + Integrated with a weather API to fetch real-time data.

#### **Slide 2: HTML Structure Overview**

* **Title:** HTML - The Backbone of the Page
* **Content:**
  + The code defines the structure of the webpage.
  + Contains semantic tags for better readability and SEO.
  + Divided into three primary sections:
    - **Header**
    - **Main Content**
    - **Footer**

#### **Slide 3: The <head> Section**

* **Title:** Setting the Foundation
* **Content:**
  + Defines meta information and links external stylesheets.
  + Includes:
    - **Meta Charset:** UTF-8 for character encoding.
    - **Viewport Meta Tag:** Ensures responsiveness on all devices.
    - **Title Tag:** Displays the webpage title as "Weather Information."
    - **CSS Link:** Connects the styles.css file for styling.

#### **Slide 4: The Header Section**

* **Title:** Welcoming Users
* **Code Snippet**

**<header class="header">**

**<h1>Weather Information</h1>**

**<p>Your go-to source for current weather updates</p>**

**</header>**

* **Content:**
  + Displays the page title and a tagline.
  + Styled using CSS to ensure it is visually appealing.
  + Positioned at the top of the page.

#### **Slide 5: The Main Content Section**

* **Title:** Core Functionality of the Page
* **Content:**
  + **Search Section**
    - Allows users to input a city name.
    - Contains:
      * A text input field.
      * A search button.
  + **Weather Display Section**
    - Displays weather information fetched via the API.
    - Structured inside a weather-card for clarity and design.
* **Code Snippet (Search Section):**

**<section class="search-section">**

**<input type="text" id="cityInput" class="search-input" placeholder="Enter city name">**

**<button id="searchButton" class="search-button">Search</button>**

**</section>**

#### **Slide 6: Weather Card Structure**

* **Title:** Displaying Weather Data
* **Code Snippet:**

**<section class="weather-display">**

**<div class="weather-card">**

**<h2 id="cityName">City Name</h2>**

**<p id="temperature">Temperature: -- &#8451;</p>**

**<p id="description">Condition: --</p>**

**<p id="humidity">Humidity: --%</p>**

**<p id="windSpeed">Wind Speed: -- km/h</p>**

**</div>**

**</section>**

* **Content:**
  + A neatly styled card to display:
    - City Name
    - Temperature
    - Weather Condition
    - Humidity
    - Wind Speed

#### **Slide 7: The Footer Section**

* **Title:** Wrapping Up the Page
* **Code Snippet:**

**<footer class="footer">**

**<p>&copy; 2025 Weather Information. All rights reserved.</p>**

**</footer>**

* **Content:**
  + Provides copyright information.
  + Placed at the bottom for a professional layout.

#### **Slide 8: Integration with JavaScript**

* **Title:** Making It Interactive
* **Content:**
  + HTML is the structure, but JavaScript brings the page to life.
  + A <script> tag connects the script.js file, enabling user interaction and API calls.
  + Example functionality:
    - Fetching weather data when the "Search" button is clicked.
    - Dynamically updating the weather card.

#### **Slide 9: Conclusion**

* **Title:** Final Thoughts
* **Content:**
  + The HTML structure serves as a solid foundation for the Weather Information Page.
  + It is clean, semantic, and optimized for responsiveness.
  + Collaborates with CSS and JavaScript for styling and interactivity.
* ***Presentation Title: Weather Information Page “JavaScript Structure”***

#### **Step 1: Adding Event Listener**

* The searchButton is targeted using document.getElementById.
* An event listener is attached to trigger on a click event.
* Inside the event listener:
  + The user input (cityInput) is retrieved and validated to ensure it's not empty.
  + If the input is empty, an alert prompts the user to enter a city name.

#### **Step 2: Fetching Weather Data**

* The API key (apiKey) and the URL for OpenWeatherMap are defined dynamically using the city name entered by the user.
* A fetch request is made to the API endpoint:
  + If the response is successful, the data is converted into JSON.
  + If the city name is invalid or the API fails, an error is thrown.

#### **Step 3: Handling the API Response**

* If the API returns data successfully, the displayWeather function is called, passing the received weather data.
* If an error occurs (e.g., invalid city name), the error is logged in the console, and an alert notifies the user.

#### **Step 4: Displaying Weather Data**

* The displayWeather function updates the HTML elements with the received weather details:
  + cityName: Displays the city name from the API response.
  + temperature: Shows the temperature in Celsius.
  + description: Describes the current weather condition (e.g., "clear sky").
  + humidity: Indicates the percentage of humidity.
  + windSpeed: Shows the wind speed in km/h.

#### **Key Features**

* **Error Handling:** Alerts the user if the city is not found or if there's an issue with fetching data.
* **Dynamic Updates:** Weather information is updated dynamically on the webpage without reloading.
* ***Presentation Title: Weather Information Page “CSS Structure”***

#### **Step 1: Global Reset and Font Styling**

* Resets margin and padding for body, h1, h2, and p elements to ensure consistency.
* Uses box-sizing: border-box for better layout control.
* Applies the font 'Lato', sans-serif for a sleek, modern appearance.
* Sets the background color to a dark shade #1e1e2e for a professional, minimalistic look.
* The text color is set to #ffffff for readability against the dark background.

#### **Step 2: Header Styling**

* The .header class:
  + Background color: #3a3a55 (dark grayish-blue for a clean, elegant tone).
  + Text color: #ffffff to contrast with the dark background.
  + Padding: 20px for spaciousness.
  + Text alignment: center for a balanced layout.

#### **Step 3: Main Content Area**

* The .main-content class:
  + Adds padding of 20px for spacing around the content.

#### **Step 4: Search Section**

* The .search-section class:
  + Aligns the search bar and button using flex for a clean, centered layout.
  + Adds a margin-bottom of 20px for separation from the rest of the content.

#### **Step 5: Input Field Styling**

* The .search-input class:
  + Padding: 10px for comfortable input space.
  + Font size: 16px for readability.
  + Width: 60% for a responsive design.
  + Borders: Rounded with 5px for a modern touch.

#### **Step 6: Button Styling**

* The .search-button class:
  + Padding: 10px 20px for a clickable and visually appealing size.
  + Background color: #4a90e2 (bright blue for a noticeable CTA).
  + Text color: #fff to contrast with the button background.
  + Rounded corners with 5px for a soft look.
  + Transition effect for hover (background 0.3s ease) to enhance interactivity.
  + Hover state: Changes background to a darker blue #357abd for feedback.

#### **Step 7: Weather Display Styling**

* The .weather-display class:
  + Uses flex to center the weather card horizontally and vertically.

#### **Step 8: Weather Card Design**

* The .weather-card class:
  + Background color: #29293d (dark gray for subtle contrast against the page background).
  + Padding: 20px for comfortable spacing.
  + Rounded corners (10px) for a polished look.
  + Shadow effect (box-shadow) for depth and focus.
  + Width: 300px for a compact yet readable size.

#### **Step 9: Text Styling in Weather Card**

* The h2 element:
  + Font size: 24px for prominence.
  + Text color: #4a90e2 to match the button color scheme.
* The p element:
  + Font size: 18px for readability.
  + Minimal margins (5px 0) for a clean appearance.

#### **Step 10: Footer Design**

* The .footer class:
  + Background color: #333 (dark gray for cohesion with the theme).
  + Text color: #fff for readability.
  + Padding: 10px for adequate spacing.
  + Center-aligned text for simplicity.

### **Summary**

This CSS layout creates a professional, responsive, and modern design for the weather information page. The dark theme, clean fonts, and subtle hover effects combine to deliver an aesthetically pleasing user experience.

GitHub:

https:// github.com/syedabdullahofficial17/Weather-Information-Page-Project/upload/main

***OutPut:***

