Weather App

This is a simple Weather App that fetches and displays weather information for a specified city. It uses the OpenWeatherMap API to get the current weather details. This project is a part of my learning journey in web development, with a focus on integrating APIs using JavaScript.

Introduction

This project aims to provide weather updates for any city entered by the user. It displays the temperature, humidity, wind speed, and an appropriate weather icon based on the weather conditions.

Features

- Fetches weather data from OpenWeatherMap API.
- Displays city name, temperature, humidity, and wind speed.
- Shows weather icons based on current weather conditions.
- Handles invalid city names gracefully.
-

Technologies Used

- HTML
- CSS
- JavaScript
- OpenWeatherMap API

How It Works

HTML

The HTML file sets up the structure of the web page. It includes the input field for entering the city name, a button to trigger the weather fetch, and placeholders for displaying weather information.

JavaScript

The JavaScript file handles the logic of fetching and displaying the weather data. Here is a detailed explanation:

API Key and URL: The API key for OpenWeatherMap and the base URL for fetching weather data are defined.

DOM Elements: References to necessary DOM elements (input field, button, weather icon, etc.) are obtained.

```
const searchBox = document.querySelector(".search input");
const searchBtn = document.querySelector(".search button");
 const weatherIcon= document.querySelector(".weather-icon");
```

Check Weather Function: This asynchronous function fetches weather data for the specified city. It handles the API response, checks for errors, and updates the DOM with weather information.

```
async function checkWeather(city) {
            const apiurl =
`https://api.openweathermap.org/data/2.5/weather?units=metric&q=${city}&appid=${apikey}`;
            const response = await fetch(apiurl);

            if(response.status ==404){
                document.querySelector(".error").style.display = "block";
                document.querySelector(".weather").style.display = "none";
            }else{

            var data = await response.json();

            console.log(data);
            document.querySelector(".city").innerHTML = data.name;
            document.querySelector(".temp").innerHTML = Math.round(data.main.temp) + '°C';
            document.querySelector(".humidity").innerHTML = data.main.humidity + '%';
            document.querySelector(".wind").innerHTML = data.wind.speed + ' km/hr';
```

Event Listener: An event listener is added to the search button to call the checkWeather function with the city name entered by the user.

```
 searchBtn.addEventListener("click", () => {
            checkWeather(searchBox.value);
        });
```

Weather Icons: Based on the fetched weather data, appropriate weather icons are displayed. For example:

```
if (data.weather[0].main =="Clouds"){
                weatherIcon.src="images/clouds.png";

            }
            else if (data.weather[0].main =="Rain"){
                weatherIcon.src="images/rain.png";
```

Topics Learned

Throughout the development of this Weather App project, several key web development concepts and skills were learned and applied. These include:

1. **HTML Structure and Semantics**:
    - Understanding the basic structure of an HTML document.
    - Creating input fields, buttons, and containers for displaying content.
    - Using semantic HTML elements to create a well-structured webpage.
2. **CSS for Styling**:
    - Applying styles to HTML elements to enhance the visual appearance.
    - Using classes and IDs to target specific elements for styling.
    - Understanding layout concepts like flexbox or grid for positioning elements on the page.
3. **JavaScript Basics**:
    - Writing and linking JavaScript files to an HTML document.
    - Using variables, functions, and control structures (like if statements).
4. **DOM Manipulation**:
    - Selecting and manipulating DOM elements using JavaScript (querySelector, innerHTML, etc.).
    - Adding event listeners to handle user interactions (like clicking a button).
5. **Asynchronous JavaScript and Fetch API**:
    - Using the fetch function to make HTTP requests.
    - Understanding and handling promises with async and await.
    - Parsing JSON responses and using the data in the application.
6. **API Integration**:
    - Learning how to use the OpenWeatherMap API to fetch weather data.
    - Constructing API URLs with query parameters and API keys.
    - Handling API responses and errors.
7. **Error Handling**:
    - Implementing basic error handling for network requests.
    - Displaying error messages to the user when an invalid city name is entered.