Discipline

«Fundamentals of WEB technologies»

ENDTERM

Performed digital engineering

2nd year student of the specialty:Musa Nursulu

Almaty

2024

Project: "Weather Forecast" Application

Introduction

I created the "Weather Forecast" application to demonstrate my skills in web development, API integration, and user interface design. This project allows users to check real-time weather conditions for any city, including temperature, weather descriptions, humidity, and wind speed. By combining HTML, CSS, JavaScript, and the OpenWeatherMap API, I aimed to create a functional and visually appealing application.

What I Wanted to Achieve

My main goal was to develop an interactive app where users could:

1. Enter a city name to see its current weather conditions.
2. Receive instant feedback for both valid and invalid city names.
3. Enjoy smooth animations and transitions to enhance user experience.

Technologies I Used

1. **HTML** – To structure the application interface.
2. **CSS** – To design and style the app with a modern and clean look.
3. **JavaScript** – To fetch data from the weather API, process it, and dynamically update the interface.
4. **OpenWeatherMap API** – To fetch real-time weather data.

How It Works

1. The Search Functionality

I added a search box where users can type the name of a city and click the search button. When they do:

- A JavaScript function sends a request to the OpenWeatherMap API.
- If the city is found, the weather information is displayed dynamically.
- If the city is not found, an error message appears.

2. Displaying Weather Information

Once the weather data is retrieved, the app displays:

- **Temperature** in Celsius.
- **Weather condition** (e.g., clear, cloudy, rainy).
- **Humidity** percentage.
- **Wind speed** in kilometers per hour.

3. Handling Errors

If a user enters an invalid city name, the app:

- Displays an animated error message with a "City not found!" image.
- Ensures the interface updates only when the city changes.

What I Learned

1. **Using APIs:** This project taught me how to connect to an external API, send requests, and handle responses. For example, I constructed dynamic API calls like this:

```
search.addEventListener('click', () => {
    const APIKey = '80a58a6a4d95c1ab620b1c31cca12079';
    const city = document.querySelector('.search-box input').value;

    if (city === '') return;

    fetch(`https://api.openweathermap.org/data/2.5/weather?q=${city}&units=metric&appid=${APIKey}`)
        .then(response => response.json())
        .then(json => {
            console.log(json);
```

2. **DOM Manipulation:** I dynamically updated elements on the page based on the retrieved data:

```
temperatureElement.innerHTML = `${tempValue}<span>°C</span>`;
description.textContent = json.weather[0].description;
humidity.textContent = `${json.main.humidity}%`;
wind.textContent = `${parseInt(json.wind.speed)}Km/h`;
```

Animations: Using CSS transitions, I added smooth animations to make the interface more engaging.

How I Structured the Project

1. HTML

The interface is designed using HTML. I divided it into sections like the search box, weather details, and an error message block.

Example Code:

```
<div class="search-box">
    <input type="text" placeholder="Enter city name">
    <button class="bx bx-search"></button>
</div>


<div class="weather-box">
    <p class="temperature">0<span>°C</span></p>
    <p class="description">Cloudy</p>
</div>
```

2. CSS

I styled the app with a clean, modern design, including transparent backgrounds and animations.

Example Code:

```
body {
  background: url('images/back.jpg');
  background-size: cover;
  font-family: "Poppins", sans-serif;
}


.container {
  width: 400px;
  padding: 20px;
  background: rgba(255, 255, 255, 0.1);
  border-radius: 16px;
  color: white;
}
```

3. JavaScript

I wrote the logic for fetching data, updating the DOM, and handling errors.

Key Functions:

- **Fetch Weather Data:**

```
fetch(`https://api.openweathermap.org/data/2.5/weather?q=${city}&units=metric&appid=${APIKey}`)
  .then(response => response.json())
  .then(data => {
    if (data.cod === '404') {
      showError('City not found!');
      return;
    }
    updateWeather(data);
  });
```

- **Update Weather Information:**

```
function updateWeather(data) {
  const temperature = document.querySelector('.temperature');
  temperature.innerHTML = `${Math.round(data.main.temp)}<span>°C</span>`;
}
```

Challenges I Faced

1. **Error Handling:**
   Initially, the app would crash when I entered an invalid city name. I solved this by checking the API response code and showing an appropriate error message.
2. **Animations:**
   Making the transitions smooth and user-friendly required extra time, especially to synchronize them with data updates.
3. **API Key Security:**
   I learned the importance of securing API keys, especially when hosting projects online.

Conclusion

This project helped me understand the importance of combining technical skills with design principles to create a user-friendly application. It also reinforced my knowledge of APIs, JavaScript, and CSS animations. Overall, the "Weather Forecast" app serves as a strong example of my web development abilities.