Open Weather (Approach & Challenges)

Contents

Challenges

- **API Limitation**: The OpenWeather API has a limitation of 60 calls per minute. This means that we can only make 60 calls to the API in a minute. This limitation can be a challenge when we are trying to fetch data for multiple cities at the same time.
- **Data Parsing**: The data returned by the OpenWeather API is in JSON format. Parsing this data and extracting the required information can be a challenge, especially when dealing with nested JSON objects.
- **Initial Location**: The OpenWeather API requires the city name to fetch the weather data. This means that we need to know the city name in advance to fetch the weather data. If the user does not provide the city name, we need to have a default city to fetch the weather data.
- **Error Handling**: Handling errors that occur during the API call is important to ensure that the application does not crash. We need to handle errors such as network errors, server errors, and invalid responses from the API.
- **Internationalization**: The OpenWeather API returns data in multiple languages. We need to handle the internationalization of the data to display it in the user's preferred language.

Approach & Methodology

- **API Calls**: To fetch weather data from the OpenWeather API, I have used the `fetch` API in JavaScript. I have made asynchronous API calls to fetch the weather data for the specified city.
- **Debounce API Calls for frequent searches**: To avoid hitting the API rate limit, I use a debounce function to limit the number of API calls made in a short period of time. This ensures that we do not exceed the rate limit set by the OpenWeather API. **`(more requests = more cost)`**

- **Data Parsing**: I have used JavaScript to parse the JSON data returned by the OpenWeather API. I have extracted the required information such as temperature, humidity, wind speed, etc., from the JSON response.

- **Error Handling**: I have implemented error handling to catch any errors that occur during the API call. I display an error message when an error occurs, or the city name is not found in the API response.

= **Initial Location**: I have used an external API to fetch the user's location based on their IP address, then used the city name to fetch the weather data from the OpenWeather API. This improves the user experience by providing weather data for the user's location without requiring them to enter the city name.

- **Internationalization**: I have implemented internationalization to display the weather data in the user's preferred language.

Conclusion

In conclusion, fetching weather data from the OpenWeather API involves challenges such as API limitations, data parsing, error handling, and internationalization. By following the approach and methodology mentioned above, we can overcome these challenges and provide a seamless weather data fetching experience to the users.

Additional

Why I have used `Debounce` not `Throttle`?

Debounce and Throttle can be used to limit the number of API calls made in a short period of time. The difference between the two is that debounce will wait for a specified time period before making the API call, while throttle will limit the number of API calls made in a specified time period.

In my case I do not have search suggestions or live search, so there is no need to make API calls while typing. I only need to make an API call when the user has finished typing the city name. Therefore, I have used debounce to wait for a specified time period before making the API call.