Development Strategy

For this project, you will be writing most of your code in two files: `server.js` and `website/app.js`. Note that it's very important that you plan your project before you start writing any code! Break your project down into small pieces of work and strategize your approach to each one. With these bite-sized amounts, it'll be easier to debug and fix any issues that appear.

Testing

Testing your code as you go is an excellent development approach. If you would like to write and run tests for parts of your implementation code, you can use the file `tests.js` to see examples of test code you might write along the development path.

Feel free to implement your own design workflow, but if you get stuck -- here is a walkthrough to get you up and running!

1. *Start by setting up your project environment.* Make sure Node is installed from the terminal. Install the packages Express, Body-Parser, and Cors from the terminal and them include them your `server.js` file.
   - Create a server running on the port of your choosing
   - Add a `console.log()` to the server callback function, and test that your server is running using Node in the terminal to run the file `server.js`
2. *Add a GET route that returns the `projectData` object in your server code* Then, add a POST route that adds incoming data to `projectData`.
   - The POST route should anticipate receiving three pieces of data from the request body
     - temperature
     - date
     - user response
   - Make sure your POST route is setup to add each of these values with a key to `projectData`.
3. *Acquire API credentials from OpenWeatherMap website*. Use your credentials and the base url to create global variables at the top of your `app.js` code.

- Write an `async` function in `app.js` that uses `fetch()` to make a GET request to the OpenWeatherMap API.
- Create an event listener for the element with the id: `generate`, with a callback function to execute when it is clicked.
- Inside that callback function call your `async` GET request with the parameters:
    - base url
    - user entered zip code (see input in html with id `zip`)
    - personal API key

4. After your successful retrieval of the weather data, you will need to chain another Promise that makes a POST request to add the API data, as well as data entered by the user, to your app.

    - You will need to write another `async` function to make this POST request.
    - The function should receive a path and a data object.
    - The data object should include
        - temperature
        - date
        - user response

    - Remember, you can access the value of DOM elements by selecting them in your JS code.

5. *Finally, chain another Promise that updates the UI dynamically* Write another `async` function that is called after the completed POST request. This function should retrieve data from our app, select the necessary elements on the DOM (`index.html`), and then update their necessary values to reflect the dynamic values for:

    - Temperature
    - Date
    - User input