Instructions

Overview

In this project, students are expected to build a website using the Express/Node.js platform, with the Axios HTTP client, that integrates a chosen public API from the given list: **Public API Lists**. The website should interact with the chosen API, retrieve data, and present it in a user-friendly manner.

Objectives

- Develop an understanding of how to integrate public APIs into web projects.
- Gain practical experience using Express/Node.js for server-side programming.
- Enhance understanding of client-server communication using Axios.
- Demonstrate ability to manipulate, present, and work with data retrieved from APIs.

Example Ideas

- Use the JokeAPI to Create a website that gives the user a joke based on their name.
- Use the OpenWeatherMap API to build a website that tells a user if it will rain tomorrow in their location of choice.
- Use the Blockchain API to check the price of a cryptocurrency for the user.
- Use the CocktailDB API to make a website that gives the user a random cocktail recipe with images of the cocktail.
- Use the Open UV API to make a website based on your home location that tells you if you need to apply sunscreen today.
- Bonus points for choosing your own API and creating something of your own. Sometimes, you'll decide on an API and then realise that it doesn't work, that's all a part of the process, just move on to the next one and try again!

Requirements

1. API Choice

- Browse through the provided list and choose an API of interest. This choice should be guided by the potential to retrieve, manipulate, and present data in a meaningful and interactive way. I recommend choosing an API that does not require authentication and is CORS enabled. (What is CORS?)

2. Project Planning

- Think through your project, researching the chosen API, its features, what data it will provide, and how it will be used in your web application.

3. Project Setup

- Set up a new Node.js project using Express.js.
- Include Axios for making HTTP requests.
- Include EJS for templating.
- Ensure that the project has a structured directory and file organization.

4. API Integration

- Implement at least a GET endpoint to interact with your chosen API.
- Use Axios to send HTTP requests to the API and handle responses.

5. Data Presentation

- Design the application to present the retrieved data in a user-friendly way. Use appropriate HTML, CSS, and a templating engine like EJS.

6. Error Handling

- Ensure that error handling is in place for both your application and any API requests. You can console log any errors, but you can also give users any user-relevant errors.

7. Documentation

- Include comments throughout your code to explain your logic.

8. Code Sharing

- Use what you have learnt about GitHub to commit and push your project to GitHub so that you can share it with other students in the Q&A area, I'd love to see what you've build too! You can tweet at me @yu_angela
- Include a Readme.md file that explains how to start your server, what commands are needed to run your code. e.g. npm i and then nodemon index.js

Recommended Resources

- Express.js: **Getting Started Guide**
- Node.js: **Documentation**
- Axios: **Documentation**
- Public APIs: **API List**
- More free APIs: https://rapidapi.com/collection/list-of-free-apis