AMALITECH

Week 2.3

Lab Activity: Advice Generator App

Learning Objectives:

- **API Calls:** Practice making asynchronous requests to fetch data from an external API.
- **Promises:** Understand and implement promises to handle the asynchronous nature of API calls.
- **Async/Await (Optional):** Explore the more modern `async/await` syntax for working with promises.
- **DOM Manipulation & Event Handling:** Update the UI dynamically based on the fetched advice.
- **Error Handling:** Implement error handling mechanisms for failed API requests.

Project Setup:

1. **Get the Project Files**: Download the project files from the link given by the trainer.
2. **Familiarize with the UI:** Review the provided HTML, CSS, and design assets to understand the layout and desired functionality.

Tasks:

1. HTML Structure:
    - Analyze the provided HTML to identify elements for displaying the advice, button, and any loading indicators.
    - Ensure all necessary elements are present and have the appropriate IDs or classes for JavaScript interaction.
2. CSS Styling:
    - Customize the CSS (or use the provided styles) to match the design requirements.
    - Focus on layout, typography, colors, and responsive design.
3. JavaScript Functionality:
    - **API Interaction**:
        - Identify the API endpoint provided in the project instructions.
        - Use `fetch` (or a library like Axios) to make a GET request to the API.

- **Use Promises (or `async/await`) to handle the response**:
    1. Parse the JSON response data.
    2. Extract the advice text and ID.
    3. Update the UI elements with the fetched advice.
- **Event Handling**:
    1. Add an event listener to the button to trigger a new advice request when clicked.
    2. Optionally, display a loading indicator while the request is in progress.
- **Error Handling**:
    1. Catch and handle errors gracefully if the API request fails (e.g., display an error message).
    2. Implement a retry mechanism (e.g., a "Try Again" button) in case of errors.

Evaluation:

- **Functionality**:
    - Advice is fetched and displayed correctly in the UI.
    - Clicking the button triggers a new advice request.
    - Error handling is implemented for failed requests.
- **Code Quality**:
    - Clean, organized, and well-structured JavaScript code.
    - Effective use of promises (or async/await) for asynchronous operations.
    - Proper DOM manipulation and event handling techniques.
- **UI/UX:**
    - The application matches the design provided.
    - The interface is intuitive and easy to use.
    - Loading indicators and error messages enhance the user experience.