AMALITECH

Week 2.1

Lab Activity: Password Generator App

Learning Objectives:

- **Modern JavaScript (ES6+):** Understand and apply key ES6+ features to enhance the functionality and maintainability of your front-end code.
- **Arrow Functions:** Utilize arrow functions for concise function expressions and their lexical `this`behavior.
- **Template Literals:** Leverage template literals for creating dynamic strings and embedding expressions.
- **Destructuring Assignment:** Extract values from arrays and objects for cleaner, more readable code.
- **Classes:** Implement the password generator logic using ES6 classes for better organization and code structure.
- **Let and Const:** Understand the difference between `let` and `const` for variable declaration and scoping.
- **Default Parameters:** Utilize default parameters to provide fallback values for function arguments.
- **Spread Syntax and Rest Parameters:** Employ these features for working with arrays and function arguments more efficiently.

Introduction:

In this lab, you'll build a password generator app based on the UI provided. The app will allow users to customize their passwords based on criteria like:

- Password length
- Inclusion of lowercase letters, uppercase letters, numbers, and symbols.

The app will also feature:

- Strength indicator
- Button to copy the generated password to the clipboard.

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**:
   - Create the necessary HTML elements for the following components:
     - A slider for the password length
     - Checkboxes for character types
     - Button to generate the password
     - Copy button
     - Strength indicator
2. **CSS Styling:**
   - Style the elements according to the design provided.
   - Ensure the layout is responsive and works well on different screen sizes.
   - Consider using a CSS preprocessor (like Sass or Less) for easier style organization.
3. **JavaScript Functionality (with ES6 Emphasis):**
   - Write functions to (use ES6 features where applicable):
     - Generate a random character based on selected criteria.
     - Generate a password based on user input.
     - Update the password display area with the generated password.
     - Calculate the strength of the password.
     - Copy the generated password to the clipboard.
   - Use ES6 classes to structure your JavaScript code.
   - Leverage letand const for variable declarations.
   - Apply default parameters for flexibility in function arguments.
   - Utilize spread syntax and rest parameters to streamline array and function argument handling.

4. **Event Handling:**
    - Add event listeners to handle user interactions with the interface.
5. **Testing and Refinement:**
    - Thoroughly test the functionality to ensure it works as expected.
    - Refine the design and functionality based on your ideas and feedback.

Password Strength Rubric

Strength	**Criteria**
Too Weak	Less than 8 characters
Weak	8 or more characters, but only one character type (lowercase, uppercase, numbers, or symbols)
Medium	8 or more characters, with at least two character types.
Strong	12 or more characters, with at least three character types.

Evaluation

Your password generator app will be evaluated based on the following criteria:

1. **ES6 Proficiency:**
    - Effective use of arrow functions for concise syntax.
    - Utilization of template literals for dynamic string creation.
    - Appropriate application of destructuring assignment with arrays and objects.
    - Clear implementation of classes to structure your JavaScript code.
    - Consistent use of letand constfor variable declarations, demonstrating an understanding of their differences.
    - Leveraging of default parameters where applicable for function flexibility.
    - Effective use of spread syntax and rest parameters to streamline operations.
2. **Functionality:**
    - Does the app generate passwords correctly based on user input?
    - Are all the features (length selection, character type selection, copy button, strength calculator) implemented correctly?
3. **Design:**
    - Does the app have a clean, user-friendly interface that aligns with the design requirements?
    - Is the layout visually appealing and easy to understand?
4. **Responsiveness:**
    - Does the app adapt well to different screen sizes and devices?
    - Is the user experience consistent across various viewports?
5. **Code Quality:**

- Is the code well-structured, readable, and maintainable?
- Are your variable and function names clear and descriptive?