Assignment 1: Interactive JavaScript Terminal

Duration: ~ 3.5 hours to 4 hours

Assignment Overview

In this assignment, you will create an interactive JavaScript-based terminal emulator. The goal is to reinforce your skills in DOM manipulation, event handling, asynchronous JavaScript, and dynamic content rendering.

Task Breakdown

Task 1: Initial Setup (20 min.)

- Clone initial HTML, CSS, and JavaScript files.
- Set up the basic HTML structure with necessary elements such as `terminalContent`, `commandInput`, and cursor elements (`commandBeforeCursor` and `commandAfterCursor`).

Task 2: Global Variables and DOM Elements (20 min.)

- Make sure you understand global variables (`cursorPos` , `content` , `isFirstTimeRender`), and all DOM elements.

Task 3: Rendering Terminal Content (20 min.)

- Implement the `renderContent()` function:
  - Clear existing content.
  - Dynamically create and append DOM elements based on the `content` array.
  - Implement scrolling behavior to automatically scroll to new content.

Task 4: Cursor Management and Command Input (20 min.)

- Implement cursor-related helper functions:
  - `focusCommandInput()` to automatically focus on the input field.
  - `updateCursor()` to correctly reflect cursor position and update displayed text around the cursor.
  - Attach event listeners (`input` , `keyup` , and `keypress`) to handle user interactions.

Task 5: Utility Functions for Asynchronous Effects (20 min.)

- Implement asynchronous utility functions:
  - `delay(ms)` to introduce delays using Promises.
  - `displayLoadingDots()` for simulating a loading animation effect in the terminal.
  - `typeLine(line, inputCommand, key)` to simulate typing animation for dynamic content rendering.
  - `typeContent(inputCommand)` to display content sequentially based on commands.

Task 6: Command Data Structure (20 min.)

- Populate the provided `aboutCommand` object:
  - Include personal details, skills, and contact information relevant to you.

Task 7: Main Animation (20 min.)

- Implement the `animateContent(commandName)` function:
  - Clear previous terminal content.
  - Display loading animations.
  - Dynamically render the initial introductory content.
  - Provide initial instructions or welcome messages to guide the user.

Task 8: Command Processing (20 min.)

- Complete the `processCommand()` function:
  - Handle different commands (`intro` , `connect` , `skill` , `help` , `clear` , `ls`).
  - Validate and respond to user input, providing relevant terminal feedback.
  - Implement default error handling for unrecognized commands.

Task 9: Error Handling (20 min.)

- Implement the `pushErrorToTerminal(inputCommand)` function:
  - Clearly inform users when they enter unrecognized commands.
  - Suggest the available commands through an informative system message.

Task 10: Entry Point (20 min.)

- Set up the initial event listener (`DOMContentLoaded`) to trigger your main animation and properly focus the command input.

Submission Guidelines

- Submit your completed HTML, CSS, and JavaScript files.
- Include a README.md file explaining your implementation approach and any additional features.

Grading Criteria

- Correct functionality and adherence to specified behaviors.
- Quality and readability of code (comments, structure).

Good luck!