CS6.201 Introduction to Software Systems

Lab Activity 8
JavaScript Fundamentals and DOM Events

March 19, 2025

Maximum Marks: 20

Time: 2 hours

Instructions

1. Please Mention if any assumptions are made
2. You are not allowed to use any external libraries or frameworks.

---

1 Question 1: Dynamic Form Validation (6 Marks)

Task

Create a dynamic form validation system for a registration form. The form should include the following fields:
Full Name, Email Address, Password, Confirm Password

Requirements

1. **Validate all fields on submission**:
   - **Full Name**: Must not be empty.
   - **Email Address**: Must be in a valid format (use regex to validate).
   - **Password**: Must be at least 8 characters long and contain at least one uppercase letter, one lowercase letter, and one number.
   - **Confirm Password**: Must match the Password field.

2. **Error Messages**:
    - Display appropriate error messages below each field if validation fails.
    - Clear the error message when the user corrects the input.
3. **Success Handling**:
    - If all fields are valid, display a success message in an alert box and clear the form.

Provided Resources

- A template HTML file (`Q1.html`) is provided, which includes the structure of the registration form.
- You need to fill the JavaScript file (`./js/script1.js`) to implement the validation logic.

2 Question 2: Interactive To-Do List Application (7 Marks)

Task

Build an interactive to-do list application using JavaScript and DOM manipulation. The application should allow users to:

- Add tasks to the list by typing in an input field and clicking an "Add" button.
- Mark tasks as completed by clicking on them (apply a strikethrough style).
- Remove tasks by clicking a "Delete" button next to each task.
- Show a count of remaining tasks (tasks not marked as completed).

Requirements

1. **Functional Features**:
    - Users can add tasks by typing into an input field and clicking the "Add" button.
    - Tasks can be marked as completed by clicking on them, which applies a strikethrough style.
    - Tasks can be removed by clicking a "Delete" button next to each task.

- A counter displays the number of remaining tasks (tasks not marked as completed).

2. **Data Management**:
   - Use an array to store the tasks dynamically.
   - Update the array when tasks are added, marked as completed, or deleted.
3. **UI Updates**:
   - Ensure the user interface updates automatically whenever tasks are added, marked as completed, or deleted.

Provided Resources

- A basic HTML template (`Q2.html`) is provided, which includes the structure for the input field, "Add" button, and a container for the task list.
- You need to fill the JavaScript file (`./js/script2.js`) to implement the functionality, just add the asked functionalities.

3 Question 3: Complete the Code (7 Marks)

Task

You are tasked with completing the implementation of a **Memory Match Challenge** game. The game is a card-matching puzzle where players flip two cards at a time to find matching pairs. As the TAs are **generous**, they have already provided HTML and partial JavaScript code which include the structure and some functionality, but several key features remain incomplete. Your task is to complete the missing parts of the JavaScript code to make the game fully functional.

—

Game Features

The game should include the following features:

- **Dynamic Grid**:
  - The grid size adjusts based on the difficulty slider (4x4, 6x6, etc.).
- **Card Matching**:
  - Players flip two cards at a time to find matching pairs.

- **Win Condition**:
  - The game ends when all pairs are matched.
- **Timer**:
  - Tracks how long it takes to complete the game.
- **Moves Counter**:
  - Tracks the number of moves taken to complete the game.
- **Reset Button**:
  - Restarts the game with a new shuffled grid.
- **Start Button**:
  - Initializes the game when clicked.

Provided Resources

- A partially completed HTML file (`Q3.html`) is provided, which includes:
  - A grid container (`#game-container`) for displaying the cards.
  - A difficulty slider (`#difficulty`) to adjust the grid size.
  - A reset button (`#reset-button`) to restart the game.
  - A start button (`#start-button`) to initialize the game.
  - A timer (`#timer`) and moves counter (`#moves`) to track game progress.
- A CSS file is included to style the game.
- A partial JavaScript file (./js/script3.js) is provided with some functions already implemented.

—

Requirements

Complete the following tasks in the JavaScript code:

1. **Shuffle Cards**:
   - Implement the `shuffleArray` function to shuffle the card symbols array.

2. **Check Matches**:
    - Complete the `checkMatch` function to:
        - Check if the two flipped cards match.
        - If they match, leave them flipped and increment the `matchedPairs` counter.
        - If they don't match, flip them back after a short delay (e.g., 1 second).
        - End the game and display an alert when all pairs are matched.
3. **Start Timer**:
    - Complete the `startTimer` function to:
        - Start a timer that increments every second.
        - Display the elapsed time in the `#timer` element.
4. **Initialize the Game**:
    - Ensure the `initGame` function properly resets the game state, generates cards, renders the board, and starts the timer.
5. **Event Listeners**:
    - Add event listeners for the reset button to restart the game.
    - Ensure the difficulty slider updates the grid size dynamically when changed.

—

Submission Guidelines

- Make your submissions in both github and moodle. Submit as a single `(RollNo).zip` file in moodle containing:
    - All HTML, CSS, and JavaScript files.
    - A README file explaining how to run your code and if any assumptions you made.
- Ensure your code is well-commented and easy to understand.
- Edit only in the files provided and asked to edit.

Conclusion

Any plagarism will be penalized. Happy Coding :)