Lesson 7: Using JavaScript and CSS Classes for Animations (A3 Project)

Lesson Duration: ~ 2 hours

Lesson Goals:

- Understand the relationship between JavaScript and CSS transitions.
- Learn how to dynamically add/remove CSS classes using JavaScript to control animations.
- Apply these concepts in an interactive mini-game project (A3).

Part 1: Concept Introduction - JavaScript & CSS Transitions (30 min.)

- **Explanation:**
  - How CSS transitions and animations work.
  - Controlling animations through JavaScript by toggling CSS classes.
  - Benefits of separating JavaScript logic from visual animations.
- **Example:**
  - For example if the element has a CSS attribute:

```
.element{
    opacity: 0;
    transition: opacity 0.2s ease-in-out;
}
```

And a CSS class like this:

```
.animated-class{
    opacity: 1;
}
```

Then at javascript if we add the `animate-class` to the element, we will get a nice **ease-in-out** animation:

```
element.classList.add('animated-class');
element.classList.remove('animated-class');
```

Part 2: Interactive Project (A3) - Animated Target Game (1 hours)

- **Objective:**
  - Implement a game where targets randomly appear/disappear using CSS transitions controlled by JavaScript.
- **Tasks:**
  - Dynamically generate game elements using JavaScript.
  - Control visibility and animations by toggling CSS classes.
  - Track and display score and remaining game time.
- **Key JavaScript Functions to Implement:**
  - `startGame()` to initialize and start the timers and game logic.
  - `endGame()` to conclude the game, stop animations, and display final score.
  - Event handlers for user interactions (clicks on targets).
- **CSS Class Manipulation:**

```
aim.addEventListener("click", () => {
    // Update score, deactivate target, control animation via CSS classes
});
```

Part 3: Solution Walkthrough & Discussion (30 min.)

- **Take up solution:**
  - Instructor reviews a detailed implementation of the animated target game.
  - Highlight best practices for animation optimization and event handling.
- **Discussion:**
  - Analyze common issues encountered.
  - Discuss performance optimization, browser compatibility, and UX considerations (maybe).