Assignment 6. Advanced JavaScript and DOM Manipulation

Objective

In this assignment, students will create an interactive web application centered around their project's theme. The project should showcase JavaScript fundamentals, DOM manipulation, event handling, and animations, focusing on engagement and user-friendliness.

Rules for Submission

1. **ALL TEAM MEMBERS must submit the assignment.**
2. Upload your project files (HTML, CSS, and JavaScript) in a zip folder.
3. SUBMIT REPORT(DOCX FILE) AND YOUR PROJECT FOLDER IN ZIP FORMAT. Ensure that the project runs without errors. REPORT MUST CONTAIN:
    - Team name, member names, and group
    - All tasks with screenshots of code and resulting webpages
    - Step-by-step description of achieving the goals of each task
    - URL of deployed website
4. The assignment must be submitted by the deadline, which is specified in the lms.astanait.edu.kz. **To receive a grade, you must submit by the deadline and defend your work at the practice lesson time.**
5. Prepare to describe during the defense how each feature from the requirements was implemented. Reflect on challenges encountered and how they were solved.

Part 1. Advanced JS

1. DOM Manipulation and Styling

Selecting and Manipulating HTML Elements

- Use `document.querySelector` or similar methods (e.g., `getElementById`, `getElementsByClassName`, `getElementsByTagName`) to select HTML elements. Demonstrate the ability to target specific elements effectively, whether they are unique or part of a group. For example, use `document.querySelectorAll` to select rating stars (e.g., icons). Allow users to click on a star to set their rating, changing its color to indicate selection.

- Modify the content of selected elements dynamically using properties such as `textContent, innerHTML, innerText.` This may involve updating text based on user interactions or external data sources. For example, change a message displayed on the page when a button is clicked or when an event occurs.

Dynamic Style Changes

- Implement methods to modify styles directly through JavaScript. This can include changing CSS properties such as background color, font size, visibility, and positioning. Use properties like `style.backgroundColor, style.display, style.transform` to create a dynamic user interface.
  OR
- Create a functional switch to toggle between Day and Night themes. This could involve changing colors, backgrounds, and text styles. Use classes or inline styles to achieve the desired effect and enhance user experience.

Manipulating Attributes (choose one option to make)

- Implement a “Read More” button that, when clicked, toggles the visibility of additional content by changing the `style.display` property.

OR

- Create an image gallery with thumbnails. When a thumbnail is clicked, dynamically change the `src` of a larger display image to show the selected thumbnail.

OR

- Use `document.getElementById` to select a greeting element. Create an input field that allows users to enter their name, which updates the greeting dynamically when they submit.

OR

- Use `document.querySelectorAll` to select rating stars (e.g., icons). Allow users to click on a star to set their rating, changing its color to indicate selection.

OR

- Use `document.querySelector` to select a content area. Implement a button that, when clicked, fetches and displays new content (e.g., a random fact or quote) in that area.

2. Event Handling

- **Event Listeners on Buttons**

Implement at least one button that triggers an action when clicked using addEventListener. Examples (you can add one of these options or develop your own event listener suitable for your team's project):

  - Create a button that, when clicked, displays the current time in a designated area. Use `new Date().toLocaleTimeString()` in the button's click event to get the current time.

OR

  - Make a reset button that clears all inputs in a form when clicked.Use `document.querySelectorAll('input').forEach(input => input.value = '')` to reset form fields.

OR

  - Create a button that loads additional content (like more posts or products) without refreshing the page. Use `fetch` to retrieve content from a server or an API and append it to a specific container.

- **Keyboard Event Handling**

Use a keydown or keypress event to detect key presses and trigger actions.

Implement keyboard navigation for a navigation menu so users can use arrow keys to move between items. Capture `keydown` events and adjust focus to the next/previous menu item based on arrow key presses.

- **Responding to Events with Callbacks**
  Implement a callback function to handle user interactions.
  - Create a contact form that submits data asynchronously and displays a success message without page reload. Use `fetch` to POST the form data and provide feedback through a callback function.

  OR

  - Implement a multi-step form where users can navigate between steps using “Next” and “Back” buttons. Use a callback function to display the correct step based on user actions, maintaining the state of inputs.
- **Switch Statements**
  Use a switch statement to control logic based on user input or interaction (e.g., responding differently to various key presses or button clicks).
  - Build a product filtering system (e.g., e-commerce site) where users can filter products by category or price range.

  OR

    - Create a news feed application that allows users to switch between different categories (e.g., Sports, Technology) using buttons.

OR

    - Create a function that displays a different greeting based on the time of day (morning, afternoon, evening).

OR

    - Implement a language selector that changes the language of a website based on user choice (e.g., English, Russian, Kazakh).

3. JavaScript Advanced Concepts

- **Objects and Methods**
  Use JavaScript objects and their methods to structure data or handle logic in the project. Display or manipulate object properties on the page.
- **Arrays and Loops**
  Use arrays to manage collections of items and loops (for, while) to iterate over elements (e.g., to display a list of items dynamically).
- **Higher-Order Functions**
  Implement at least one higher-order function that takes another function as an argument (e.g., using map, filter, forEach).
- **Play Sounds**
  Use JavaScript to trigger sound effects. Example – set up a button that plays a notification sound when clicked.
- **Animations**
  Add animations to elements (e.g., using style.transform or CSS transitions), triggered by events.

4. Structure and Separation of Concerns

- **Separation of HTML, CSS, and JavaScript**
  Ensure that the structure (HTML), style (CSS), and behavior (JavaScript) are clearly separated. Avoid inline styling or scripts directly in HTML elements.
- **Clean Code and Readability**
  Write well-organized, readable code. Use comments where necessary to explain logic.
- **Proper Use of Functions and Variables**
  Use functions to encapsulate repeated code, and choose meaningful variable names.

5. Creativity and Engagement

- **Theme Integration**

The project should reflect your team's theme and be relevant to it. Make the content engaging and creative in a way that aligns with your theme.

- **User Experience**
  Ensure the project is easy to use and intuitive for the user.
- **Fun and Interactivity**
  Add fun or engaging elements, such as sounds, animations, or interactive content that encourage user engagement.

Part 2. Deploy the Updated Website

- Commit and push all changes to your project's shared repository.
- Use GitHub Pages or Netlify to rebuild and re-deploy your site.
- **<u>Submit the URL link in your report.</u>**

Evaluation Criteria (100 points total)

- ➔ **DOM Manipulation and Styling (10%)**
  - ◆ Elements are correctly selected and modified using JavaScript.
  - ◆ Content, attributes, and inline or class-based styles are dynamically updated in response to user interaction.
- ➔ **Event Handling (10%)**
  - ◆ Event listeners are properly implemented for clicks, keyboard input, and other user actions.
  - ◆ Callbacks and switch statements are used effectively to handle different event responses.
- ➔ **Data Structures and Functions (10%)**
  - ◆ Objects, arrays, and loops are effectively used to manage and process data.
  - ◆ Functions are modular, reusable, and demonstrate logical data handling.
- ➔ **Sound Effects (5%)**
  - ◆ Sound effects play correctly in response to specific user actions (e.g., button clicks or key presses).
  - ◆ Audio does not overlap or cause performance issues.
- ➔ **Animations (5%)**
  - ◆ Animations or transitions are smoothly applied to elements.
  - ◆ Animations are event-triggered and enhance interactivity.

- **Clean Code and Structure (5%)**
  - HTML, CSS, and JavaScript are clearly separated and well-organized.
  - Variable names, indentation, and comments follow good coding practices.
- **UX and Engagement (10%)**
  - The interface aligns with the project’s theme and provides a positive user experience.
  - Interactive and visual elements engage users and encourage them to explore.
- **Defense during Practice Lesson (35%)**
  - The student explains the assignment clearly.
  - Demonstrates understanding of the assignment
  - Answers the instructor’s questions correctly.
  - Completes the given task from the instructor correctly.
- **Report (10%)**
  - The report is well-written, clear, and concise.
  - Includes objective, description of steps taken, screenshots of webpage, and final reflection.
  - Proper formatting (headings, bullet points, or numbered sections).