Objects & DOM Manipulation (15 hours)

(Tasks 1–5) → 4 hours

Task 1: User Profile Object Manipulation

Requirements

1. Create an object named userProfile with at least the following properties:
    a. name (string)
    b. age (number)
    c. email (string)
    d. isActive (boolean)
2. Update **at least two properties** using **dot notation**.
3. Update **at least two properties** using **bracket notation**.
4. Add **one new property dynamically** using bracket notation.
5. Display the **entire updated object** in the console.
6. Log each property value individually with clear labels.

Task 2: Function Modifying an Object

Requirements

1. Create an object representing a user or product with at least **three properties**.
2. Write a function that:
    a. Accepts the object as a parameter.
    b. Modifies **two or more properties** of the object.
3. The function must return a **formatted string** summarizing the updated object data.
4. Call the function and log the returned summary to the console.
5. Verify that the original object has been updated.

Task 3: Nested Shopping Cart Object

Requirements

1. Create a cart object containing:
    a. A nested object or array for items.
    b. Each item must have name, price, and quantity.
2. Access nested properties correctly to calculate the **total cost**.
3. Calculate the total price using **object values only** (no hardcoded numbers).
4. Log the final total price to the console with a clear message.
5. Ensure the solution works if item values change.

Task 4: Looping Through Student Marks

Requirements

1. Create an object where:
   a. Keys are student names.
   b. Values are marks (numbers).
2. Loop through the object using a suitable method.
3. For each student:
   a. Check if the mark is **greater than or equal to the pass mark**.
   b. Determine pass or fail.
4. Log each student's name, mark, and pass/fail status.
5. Do not hardcode student names inside the loop.

Task 5: Object Methods Using this

Requirements

1. Create an object with:
   a. At least two properties.
   b. At least one method.
2. The method must:
   a. Use the this keyword to access object properties.
   b. Update at least one property of the object.
3. Log the updated property values from inside the method.
4. Call the method and confirm that the object's data has changed.
5. Log the final object state to the console.

(Tasks 6–10) → 5 hours

Task 6: Selecting Elements & Updating Content

Requirements

1. Create HTML elements that include:
   a. At least one element with an id
   b. At least two elements sharing the same class
2. Select:
   a. One element using getElementById()
   b. Multiple elements using getElementsByClassName()
   c. One element using querySelector()
3. Update the textContent of each selected element.
4. Change at least **two style properties** of the selected elements dynamically.
5. Ensure all updates happen using JavaScript only.

Task 7: Toggle Class on Multiple Elements

Requirements

1. Create a button that the user can click.
2. Select multiple elements using a class selector.
3. Attach a click event listener to the button.
4. Loop through the selected elements using a loop.
5. Use classList.toggle() to add or remove a CSS class on each element.
6. The toggled class must visibly change the element's appearance.

Task 8: Dynamically Create a List from Objects

Requirements

1. Create an array containing **objects** (each object should have at least two properties).
2. Select a container element from the DOM.
3. Dynamically create a list element (ul or ol) using createElement().
4. Loop through the array and:
    a. Create a list item for each object.
    b. Insert object data into the list item using textContent.
5. Append each list item to the list using appendChild().
6. Append the final list to the DOM.

Task 9: DOM Traversal

Requirements

1. Select a specific element from the DOM.
2. Access and modify:
    a. Its parent element
    b. At least one child element
    c. One sibling element
3. Update either the textContent or styles of the traversed elements.
4. Use DOM traversal properties only (no re-selecting elements).
5. Ensure the changes are visible in the browser.

Task 10: Attribute Manipulation Based on User Interaction

Requirements

1. Create at least one interactive element (button or checkbox).
2. Select a target element whose attribute will be modified.
3. On user interaction:
    a. Check if a specific attribute exists using hasAttribute().
    b. Add the attribute using setAttribute() if it does not exist.
    c. Remove the attribute using removeAttribute() if it exists.
4. The attribute change must affect the element's behavior or appearance.
5. Log the attribute state to the console after each interaction.

.

(Tasks 11–14) → 4 hours

Task 11: Using the Event Object with Multiple Buttons

Requirements

1. Create **at least three buttons** on the page.
2. Select all buttons using a single selector.
3. Attach the same event listener to each button.
4. Use the **event object** to:
    a. Identify which button was clicked.
    b. Identify the event type.
5. Display the clicked button's text and event type inside the DOM (not only in the console).
6. Ensure the output updates every time a button is clicked.

Task 12: Form Validation with Prevent Default

Requirements

1. Create a form with **at least two input fields** and a submit button.
2. Attach a submit event listener to the form.
3. Prevent the default form submission behavior.
4. Validate that **all required fields are filled**.
5. Display an error message inside the DOM for each empty field.
6. Allow form submission logic to continue only when all fields are valid.

Task 13: Real-Time Input Validation

Requirements

1. Create a form with multiple input fields.
2. Validate input values **while the user types** using an appropriate event.
3. Highlight invalid inputs by:
    a. Adding a CSS class when invalid.
    b. Removing the class when valid.
4. Display real-time validation messages next to each input.
5. Ensure validation feedback updates immediately as input changes.

Task 14: Keyboard Events Interaction

Requirements

1. Select an element where content will be displayed dynamically.
2. Listen for both keydown and keyup events.
3. Detect **specific keys** pressed by the user.
4. Update the content or style of the selected element based on the key pressed.
5. Use the event object to identify the key.
6. Ensure the page responds instantly to keyboard input.

Task 15: Interactive To-Do List with Validation

Task 15 -> 2 hours

Requirements

1. Create an object named todoData to store tasks added by the user.
2. Dynamically create an input field, an add button, and a task list using JavaScript.
3. Allow users to mark tasks as completed and remove tasks using click events.
4. Update both the DOM and the todoData object when tasks are added or removed.
5. Reuse DOM references and ensure all interactions work on a single page without reloads.