LESSON 48

Scripting and DOM Manipulation

40 minutes | Introduction to Web Frontend | Outcome 3.3 | Outcome 1.22 | Outcome 2.7 | Outcome 1.15 | Outcome 2.20 | Outcome 1.23 | Outcome 2.21

WHAT THIS LESSON DOES

In this lesson, you'll learn the basics of scripting with JavaScript and manipulating the Document Object Model (DOM). Follow step-by-step instructions to set up a project, create HTML and CSS files, and use JavaScript to add interactivity to your web page.

CURRICULUM OUTCOMES

Outcome	What the specification asks for
3.3	use appropriate programming languages to develop an interactive website that can display information from a database that meets a set of users' needs
1.22	read, write, test, and modify computer programs
2.7	implement algorithms using a programming language to solve a range of problems
1.15	consider the quality of the user experience when interacting with computers and list the principles of universal design, including the role of a user interface and the factors that contribute to its usability
2.20	identify and fix/debug warnings and errors in computer code and modify as required
1.23	reflect and communicate on the design and development process
2.21	critically reflect on and identify limitations in completed code and suggest possible improvements

WHAT STUDENTS SHOULD BE ABLE TO DO

- Understand the concept of the Document Object Model (DOM) and its role in web interactivity.
- Develop skills in using JavaScript to access and modify HTML elements dynamically.
- Apply event handling techniques to create responsive user interactions.
- Manipulate element styles and content through scripting to enhance web page functionality.
- Build a structured web project with integrated HTML, CSS, and JavaScript files.

BEFORE THE LESSON

- Confirm VS Code opens and lets students create folders and files, since the whole lesson depends on it. No extra extensions are needed; a browser opening a local index.html file is enough to see results.
- Have your own working copy of the finished project so you can show the expected page and diff it against a student whose buttons do nothing.
- Decide how students save: a folder in Documents survives, but a shared or roaming profile may not keep the css and js subfolders reliably.

Scripting and DOM Manipulation

HOW THE LESSON RUNS

- Introduction to Scripting and DOM
- Creating Your Project Folder
- Creating the HTML File
- Adding CSS Folder and Stylesheet
- Adding JS Folder and JavaScript File
- Using Onclick Attribute
- Adding Event Listeners
- Manipulating Text Colour and Size
- Challenge: Make Text Small
- Experiment With Other Event Listeners
- Wrap Up

TEACHING MOVES

- 1. Introduction to Scripting and DOM. Say plainly that the DOM is the live page as JavaScript sees it, not the HTML file on disk. Refreshing the browser rebuilds it from the file; script changes only last until refresh.
- 2. Creating Your Project Folder. Insist the folder name has no spaces and that they Open Folder, not just open the file. If the folder name is not in the Explorer pane, later right-click New File steps will go astray.
- 3. Creating the HTML File. Check everyone named it exactly index.html, not index.html.txt. Open it in the browser now so they have a plain baseline before any styling or script exists.
- 4. Adding CSS Folder and Stylesheet. The common failure is a wrong href path. Confirm the link reads css/styles.css and that styles.css is inside the css folder, not beside index.html. A navy heading confirms it worked.
- 5. Adding JS Folder and JavaScript File. Stress that the script tag goes at the very end of the body, so the elements exist before the script runs. Placing it in the head is why later getElementById will return null.
- 6. Using Onclick Attribute. Show onclick as the quick, tightly-coupled way: markup and behaviour live together. Set it up as the thing addEventListener will improve on, so the contrast in the next step lands.
- 7. Adding Event Listeners. Draw the split explicitly: HTML holds structure, JS holds behaviour, and addEventListener keeps them apart. Point out you can attach several listeners to one element, which an onclick attribute cannot do cleanly.
- 8. Manipulating Text Colour and Size. Show that style properties in JavaScript are camelCase and take string values, so it is element.style.fontSize = '24px', not font-size. This trips up students who know CSS syntax.
- 9. Challenge: Make Text Small. Let them work from the size button pattern independently. Before they type, ask which element, which event and which property they will change. That sentence is the whole solution.
- 10. Experiment With Other Event Listeners. Have finishers try mouseover and mouseout as a pair on the same element. This makes reversible behaviour visible and shows an event does not have to be a click.
- 11. Wrap Up. Ask the class why splitting HTML, CSS and JS into separate files is better than one file. Draw out maintenance and reuse, the reasoning they will need in a written answer.

WHAT TO WATCH FOR

- Students expect script changes to persist and are confused when a refresh resets the page. Refresh the browser in front of them and let the text revert. Reinforce that the file on disk is unchanged and the DOM is rebuilt fresh each load.
- getElementById returns null and the buttons do nothing, usually because the script runs before the elements exist or an id is misspelt. Open the browser console together and read the error. Check the script tag is at the end of the body and that the id in JS matches the id in the HTML exactly, case included.
- Students write CSS property names in JavaScript with hyphens, so element.style.font-size fails silently. Show the camelCase rule: fontSize, backgroundColor. Have them fix one property live and watch it take effect.

DIFFERENTIATION

- Emerging: If files are fighting them, give the working folder to copy and have them focus only on the JavaScript step, not the setup. Pair them with someone whose page already styles correctly so they can compare paths side by side.
- Developing: Ask them to explain aloud which element, event and property each button changes before they trust it works. Have them add a comment above each listener saying what it does in plain English.
- Proficient: Challenge them to make one button toggle a state, larger then back to normal, using an if check on the current size. Ask them to rewrite the remaining onclick attribute as an addEventListener and justify why, the kind of design reasoning useful as evidence in an Applied Learning Task.