LESSON 50

Integrating External Libraries and APIs

40 minutes | Introduction to Web Frontend | Outcome 1.22 | Outcome 1.23 | Outcome 2.7 | Outcome 2.15 | Outcome 2.20 | Outcome 3.3 | Outcome 1.15

WHAT THIS LESSON DOES

In this lesson, you'll explore how to integrate external libraries and APIs into your web projects. Learn to set up a workspace, add jQuery, build a weather app with HTML, and fetch data from the OpenWeatherMap API to display weather information.

CURRICULUM OUTCOMES

Outcome	What the specification asks for
1.22	read, write, test, and modify computer programs
1.23	reflect and communicate on the design and development process
2.7	implement algorithms using a programming language to solve a range of problems
2.15	explain what is meant by the World Wide Web (WWW) and the Internet, including the client server model, hardware components and communication protocols
2.20	identify and fix/debug warnings and errors in computer code and modify as required
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
1.15	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

WHAT STUDENTS SHOULD BE ABLE TO DO

- Understand the purpose and benefits of integrating external libraries and APIs into web projects.
- Master the setup of a project workspace and inclusion of libraries like jQuery via CDN.
- Develop proficiency in using jQuery syntax, selectors, and event handling for interactive web elements.
- Gain skills in fetching and displaying data from external APIs using AJAX requests.
- Explore and customise API data output to enhance application functionality.

BEFORE THE LESSON

- Confirm VS Code and a Live Server or equivalent preview works on the room machines, and that the school filter allows both code.jquery.com and api.openweathermap.org. Test one full request yourself before class.
- Have the OpenWeatherMap API key ready to share, and check it still returns data. Keys can be rate limited or expired, so verify on the day.
- Save a working copy of the finished index.html, styles.css and script.js somewhere the class can reach in case a student's request never succeeds.

Integrating External Libraries and APIs

HOW THE LESSON RUNS

- Introduction to Integrating Libraries and APIs
- Setting Up Your Project Folder
- Creating the HTML Structure
- Adding Basic CSS Styling
- Creating the JavaScript File
- Understanding jQuery Syntax, Selectors, and Events
- Adding a Click Event Listener with jQuery
- Fetching Weather Data from the API
- Display More Information
- Challenge: Display Additional Weather Data
- Wrap Up

TEACHING MOVES

- 2. Setting Up Your Project Folder. Insist everyone opens the folder as the workspace, not a loose file. Walk the room and check the folder name shows in the Explorer pane before anyone types code.
- 3. Creating the HTML Structure. Point out the jQuery script tag in the head loads before script.js at the bottom, so jQuery is available when their code runs. Get them to name the button id and display div ids aloud.
- 5. Creating the JavaScript File. Stress the folder structure: css/styles.css and js/script.js must match the paths in the HTML exactly. A mistyped path is the most common reason nothing works and shows no error.
- 6. Understanding jQuery Syntax, Selectors, and Events. Anchor the $(selector).action() pattern against CSS selectors they already know. Write $('#getWeather') on the board and ask which HTML element it targets before moving on.
- 7. Adding a Click Event Listener with jQuery. Have them test the alert before touching the API. A working alert proves jQuery loaded and the button is wired, so any later failure is the request, not the setup.
- 8. Fetching Weather Data from the API. Use the restaurant menu analogy from the lesson: the app requests specific data, the server replies in JSON. Show the request URL structure and where the API key slots in.
- 9. Display More Information. Open the browser console together and read the whole data object aloud. Show that data.main.humidity is a path into nested JSON, not a flat variable.
- 10. Challenge: Display Additional Weather Data. Send them to the console first, not the documentation. Have them find data.wind.speed in the logged object themselves before adding it to the display.

WHAT TO WATCH FOR

- Students think a page that loads without an error is working, when the AJAX request has silently failed or returned nothing. Make console logging a habit. Show them the Network tab and console so they read the actual response rather than trusting the visible page.
- Confusing the library and the API: assuming jQuery fetches the weather or that OpenWeatherMap is part of jQuery. State it plainly: jQuery is code running in their browser, the API is a service on another server. AJAX is the bridge between the two.
- Reaching for API values as flat names, like humidity, instead of the nested path data.main.humidity. Return to the logged object and trace the dots together. Have them point to each level in the JSON before writing the path.

DIFFERENTIATION

- Emerging: If the folder structure fights them, hand over the saved working files and have them get the alert firing first, then swap in the API code one line at a time.
- Developing: Ask them to explain what happens between the click and the text appearing on screen, in order, before adding the wind speed. If they can narrate it, they can extend it.
- Proficient: Have them handle a failed request: add an error case to the AJAX call and display a friendly message for a bad location. This response handling is strong evidence for an Applied Learning Task.