LESSON 52

Weather Web App

40 minutes | Introduction to Web Frontend | Outcome 1.22 | Outcome 1.23 | Outcome 1.15 | Outcome 2.15 | Outcome 2.20 | Outcome 3.1 | Outcome 3.3

WHAT THIS LESSON DOES

In this lesson, you'll build a Weather Web App using HTML, CSS, JavaScript, and jQuery. Learn to fetch real-time weather data from an API, style your app, and add a toggle between Celsius and Fahrenheit.

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
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
2.15	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
3.1	understand and list user needs/requirements before defining a solution
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

WHAT STUDENTS SHOULD BE ABLE TO DO

- Understand the fundamentals of building a web application using HTML, CSS, and JavaScript.
- Develop skills in integrating external libraries like jQuery and Fontawesome for enhanced functionality and design.
- Master the process of fetching and displaying real-time data from an API.
- Acquire the ability to implement interactive features such as toggling between temperature units.
- Explore creative ways to enhance web app functionality and user interface design.

BEFORE THE LESSON

- Run through the whole build yourself first: create the WeatherApp folder, the css and js files, and confirm the app displays live data before class. This is the single best insurance against a lesson full of stalled browsers.
- Confirm the school filter allows both the jQuery and Fontawesome CDNs and calls to api.openweathermap.org. If any are blocked, the app fails silently with no weather shown.
- The lesson embeds a shared OpenWeatherMap API key. Test that it still returns data, and know how to register a free key as a fallback if it has hit its rate limit.

Weather Web App

HOW THE LESSON RUNS

- Introduction to Weather Web App
- Setting Up Your Project Folder
- Creating the HTML Structure
- Adding CSS Styling
- Initialising JavaScript
- Fetching and Displaying Weather Data
- Challenge: Enhance Your App
- Wrap Up

TEACHING MOVES

- 2. Setting Up Your Project Folder. Insist everyone opens the folder as their VS Code workspace, not just individual files. Later relative paths to css/styles.css and js/script.js only resolve if the folder is the workspace root.
- 3. Creating the HTML Structure. Point out that jQuery and Fontawesome arrive by CDN link, not a local install. Have students save index.html and open it in the browser now, before styling, so a blank page is caught early.
- 4. Adding CSS Styling. Stress that the file must live in a css subfolder and the HTML link must match that path exactly. A mistyped path means the page loads unstyled with no error message.
- 5. Initialising JavaScript. Walk through the click handler as a placeholder: it shows a fetching message before any API exists. Have them test the empty-location alert so they see the input validation working first.
- 6. Fetching and Displaying Weather Data. Open the browser developer console together and show the network request going out. When a city returns real data, name the path from data.main.temp to the screen so the API response feels concrete, not magic.
- 7. Challenge: Enhance Your App. Set one clear goal per student rather than a menu. Geolocation and a five-day forecast are very different in difficulty, so steer emerging students to humidity or wind speed first.

WHAT TO WATCH FOR

- Students think a page that looks correct is therefore working, and cannot tell why no weather appears. Teach the developer console early. Show that a red error there, a blocked CDN or a failed API call is invisible on the page itself but obvious in the console.
- Confusing what runs where: they expect the API key or the fetch to work like server code, or worry the key is secret. Be explicit that all of this runs in the browser. The key is visible in the JavaScript, which is exactly why production apps hide keys behind a server. Name this as a real limitation.
- Broken relative paths to the CSS or JS file, blamed on the code rather than the folder structure. Have them compare the folder tree in the Explorer pane against the paths written in the HTML. The link href and script src must mirror the folders exactly.

DIFFERENTIATION

- Emerging: Provide the completed index.html and styles.css so they can focus on the JavaScript and API steps rather than fighting layout. Pair them with a partner and have them read the click handler aloud in plain English before running it.
- Developing: Ask them to trace, without running it, what data.weather[0].description holds and where it lands on screen, so they reason about the response, not just copy it.
- Proficient: Push toward the Higher Level treatment: refactor the two API calls to avoid duplication, and handle the case where a city is not found gracefully. Have them add humidity and wind speed as evidence of API integration they could reuse in a Coursework Project write-up.