RESKILLING LAB
DT-06 TRAINING & SELF-LEARNING PACK

WEB TECHNOLOGIES

From Web User to Web Creator

Beginner → Intermediate | Recommended learning time: 50–80 hours
Expanded Learner Handbook | Version 1.0 | September 2026

How to Use This Handbook

DT-06 develops foundational and intermediate competences for creating, managing and improving websites. It is designed for VET learners, employees, SME staff, jobseekers and career changers who need a practical understanding of how the web works and how accessible, responsive, secure and maintainable websites are produced.

The pathway combines standards-based HTML, CSS and JavaScript with content management, hosting, accessibility, performance, search readiness and basic web security. Learners should practise only on their own sites, local development environments, training systems or websites for which they have explicit authorisation.

1. Pathway Overview

Item	DT-06 Specification
Pathway	Web Technologies
Target level	Beginner → Intermediate
Recommended workload	50–80 hours
Target groups	VET learners; SME staff; jobseekers; career changers; content and digital staff; junior IT personnel
Entry requirements	Basic digital literacy; ability to use a browser, files/folders and common productivity tools
Possible target roles	Junior Web Developer; Website Administrator; Digital Content Specialist; Junior Front-End Assistant
Core technologies	HTML; CSS; JavaScript; browser developer tools; CMS; hosting; Git concepts
Main evidence	Multi-page website; responsive layout; accessible forms; JavaScript interaction; CMS plan; performance/security checklist; capstone website

1.1 Learning Outcomes

- Explain how browsers, servers, URLs, DNS, HTTP and HTTPS work together.
- Create semantically structured web pages using modern HTML.
- Use CSS for typography, spacing, layout, responsive design and reusable styling.
- Apply Flexbox and Grid to common page layouts.
- Use JavaScript fundamentals to add safe client-side interaction.
- Work with the DOM, events and basic form validation.
- Design mobile-first and responsive interfaces.
- Apply core web accessibility principles and test common accessibility issues.
- Create usable forms and navigation structures.
- Explain CMS concepts and manage website content systematically.
- Understand domains, hosting, deployment and basic version-control workflows.
- Apply basic SEO and web-performance practices.
- Recognise common web-security risks and apply defensive development practices.
- Use browser developer tools to inspect, test and troubleshoot pages.
- Plan and produce a small professional website for an SME or project.

2. Initial Skills Check

#	Statement	0 No	1 Partly	2 Yes
1	I can explain the difference between a website, webpage, browser and web server.	☐	☐	☐
2	I understand what a URL, domain name and DNS do.	☐	☐	☐
3	I can create a basic HTML document.	☐	☐	☐
4	I can use semantic HTML elements such as header, nav, main and footer.	☐	☐	☐
5	I can apply CSS selectors and style text, spacing and borders.	☐	☐	☐
6	I can build a basic responsive layout.	☐	☐	☐
7	I understand Flexbox or CSS Grid at a basic level.	☐	☐	☐
8	I can explain variables, conditions and functions in JavaScript.	☐	☐	☐
9	I can use JavaScript to respond to a user event.	☐	☐	☐
10	I can create an accessible form with labels.	☐	☐	☐
11	I can identify basic accessibility problems.	☐	☐	☐
12	I understand the purpose of a CMS.	☐	☐	☐
13	I understand the difference between HTTP and HTTPS.	☐	☐	☐
14	I can identify basic web-security risks such as unsafe input or exposed credentials.	☐	☐	☐

Indicative formative interpretation: 0–10 Foundation needed; 11–18 Developing; 19–23 Functional; 24–28 Ready for intermediate practice.

3. How the Web Works

The World Wide Web is an information system that uses internet infrastructure to deliver linked resources. A browser is a client application that requests resources such as HTML, CSS, JavaScript, images and data from servers and renders them into an interactive interface.

A URL identifies a resource and commonly contains a scheme, host, path, query and fragment. Domain names provide human-readable names, while the Domain Name System resolves names to network addresses used to reach servers.

HTTP defines request-response communication between clients and servers. A browser can request a resource using methods such as GET or submit data using methods such as POST. Servers return status codes, headers and content. HTTPS adds transport encryption and server authentication through TLS.

Web development is therefore not only visual design. A functioning site depends on content structure, presentation, behaviour, network communication, hosting, security, accessibility and maintainability.

Practical Application

- Apply the concept to a fictional SME website or a local training project.
- Identify one usability, accessibility, performance or security consideration.
- Produce a small piece of evidence: code, diagram, checklist, test result or improvement note.

4. HTML: Structure and Meaning

HTML provides the structure and meaning of web content. Elements describe headings, paragraphs, lists, links, images, tables, forms and sections. Browsers use this structure to create the document tree rendered on screen.

Semantic HTML uses elements according to meaning rather than appearance. Elements such as header, nav, main, article, section, aside and footer help browsers, assistive technologies, search systems and developers understand the page.

Heading levels should represent a logical hierarchy. A heading should not be chosen merely because its default size looks attractive. Visual appearance belongs primarily to CSS.

Valid, well-structured HTML reduces ambiguity and improves maintainability. Developers should prefer native HTML functionality before recreating standard controls with custom scripts.

Practical Application

- Apply the concept to a fictional SME website or a local training project.
- Identify one usability, accessibility, performance or security consideration.
- Produce a small piece of evidence: code, diagram, checklist, test result or improvement note.

5. Links, Images, Lists and Tables

Links are fundamental to the web. Link text should describe the destination or action rather than relying on vague phrases such as 'click here'. Relative URLs are useful for resources within a site, while absolute URLs identify complete external locations.

Images should have meaningful alternative text when they communicate information. Decorative images can use empty alternative text so assistive technology can ignore them. Image dimensions and formats also affect performance.

Lists communicate groups or sequences. Ordered lists are appropriate when sequence matters; unordered lists are appropriate when it does not.

Tables are intended for tabular data, not general page layout. Header cells and meaningful structure help users understand relationships between rows and columns.

Practical Application

- Apply the concept to a fictional SME website or a local training project.
- Identify one usability, accessibility, performance or security consideration.
- Produce a small piece of evidence: code, diagram, checklist, test result or improvement note.

6. Forms and User Input

Forms allow users to send information. Common controls include text inputs, email fields, checkboxes, radio buttons, select elements, text areas and buttons.

Every form control needs an accessible name, usually provided by a visible label. Placeholder text is not a substitute for a label because it can disappear and may provide insufficient context.

Client-side validation can improve usability by identifying errors before submission, but server-side validation remains necessary because client-side checks can be bypassed.

Forms should request only necessary information, provide clear instructions, preserve user input when errors occur where possible, and communicate errors in ways that are both visible and programmatically understandable.

Practical Application

- Apply the concept to a fictional SME website or a local training project.
- Identify one usability, accessibility, performance or security consideration.
- Produce a small piece of evidence: code, diagram, checklist, test result or improvement note.

7. CSS Foundations

CSS controls presentation. Rules consist of selectors and declarations. The cascade determines which declarations apply based on origin, importance, specificity and source order.

Inheritance allows some properties, such as text-related properties, to pass from parent to child elements. Understanding inheritance and the cascade prevents excessive use of highly specific selectors or !important.

The box model describes content, padding, border and margin. Layout problems often result from misunderstanding how these dimensions interact.

CSS should be organised for reuse. Repeated values can be represented through custom properties, and classes can express reusable visual patterns.

Practical Application

- Apply the concept to a fictional SME website or a local training project.
- Identify one usability, accessibility, performance or security consideration.
- Produce a small piece of evidence: code, diagram, checklist, test result or improvement note.

8. Typography, Colour and Visual Hierarchy

Typography affects readability, hierarchy and brand consistency. Websites should use a limited, purposeful type scale, adequate line spacing and comfortable line lengths.

Colour communicates emphasis and identity but must not be the only means of conveying information. Text and interactive elements need sufficient contrast for readability.

Visual hierarchy helps users understand what is most important. Size, weight, spacing, alignment and position can guide attention without creating unnecessary decoration.

Design consistency reduces cognitive load. Buttons, links, cards and form controls should behave and appear consistently across pages.

Practical Application

- Apply the concept to a fictional SME website or a local training project.
- Identify one usability, accessibility, performance or security consideration.

- Produce a small piece of evidence: code, diagram, checklist, test result or improvement note.

9. CSS Layout: Flexbox and Grid

Flexbox is a one-dimensional layout system useful for arranging items in a row or column, distributing space and aligning elements. It is well suited to navigation bars, button groups, cards and component-level layouts.

CSS Grid is a two-dimensional layout system for rows and columns. It is useful for page regions, galleries and layouts where alignment across both dimensions matters.

Flexbox and Grid reduce reliance on older layout techniques. They should be selected according to the layout problem rather than treated as competing technologies.

Modern layouts should adapt to content. Fixed dimensions can create overflow and accessibility problems when users zoom text or use smaller screens.

Practical Application

- Apply the concept to a fictional SME website or a local training project.
- Identify one usability, accessibility, performance or security consideration.
- Produce a small piece of evidence: code, diagram, checklist, test result or improvement note.

10. Responsive and Mobile-First Design

Responsive web design allows interfaces to adapt to different viewport sizes and device characteristics. It combines flexible layouts, responsive media and conditional styling.

A mobile-first approach begins with the essential experience for smaller screens and progressively enhances the layout as space becomes available. This often produces simpler content prioritisation.

Media queries apply styles when conditions such as viewport width are met. Breakpoints should respond to content needs rather than specific device models.

Responsive design also concerns interaction. Touch targets, navigation, forms and tables must remain usable on smaller screens.

Practical Application

- Apply the concept to a fictional SME website or a local training project.
- Identify one usability, accessibility, performance or security consideration.
- Produce a small piece of evidence: code, diagram, checklist, test result or improvement note.

11. JavaScript Foundations

JavaScript adds behaviour and logic to web pages. Core concepts include values, variables, operators, conditions, loops, arrays, objects and functions.

Variables store values that may be used or changed during execution. Conditions allow different behaviour depending on data, while loops repeat operations.

Functions group reusable logic and can accept inputs and return outputs. Clear function names and small responsibilities improve maintainability.

JavaScript should enhance a usable HTML foundation rather than unnecessarily replace it. Progressive enhancement allows core content and actions to remain robust when scripting fails or is unavailable.

Practical Application

- Apply the concept to a fictional SME website or a local training project.
- Identify one usability, accessibility, performance or security consideration.
- Produce a small piece of evidence: code, diagram, checklist, test result or improvement note.

12. The DOM and Browser Events

The Document Object Model represents an HTML document as objects that JavaScript can inspect and modify. Scripts can select elements, change text or attributes, create elements and respond to interaction.

Events represent occurrences such as clicks, input, form submission or keyboard activity. Event listeners connect these occurrences to functions.

DOM manipulation should be deliberate. Excessive updates can reduce performance and create interfaces that are difficult to reason about.

Dynamic content must remain accessible. If an interface changes after an action, focus management, labels and status communication may need to be considered.

Practical Application

- Apply the concept to a fictional SME website or a local training project.
- Identify one usability, accessibility, performance or security consideration.
- Produce a small piece of evidence: code, diagram, checklist, test result or improvement note.

13. JavaScript Forms and Validation

JavaScript can support validation by checking input before a form is submitted and giving immediate feedback. Common checks include required values, expected formats and logical relationships between fields.

Validation messages should explain how to correct the problem. Merely changing a border colour is insufficient because users may not perceive the visual cue.

Client-side validation improves user experience but cannot be trusted as a security boundary. The server must independently validate and sanitise submitted data.

Sensitive values, passwords and secret keys should never be embedded in public client-side JavaScript.

Practical Application

- Apply the concept to a fictional SME website or a local training project.
- Identify one usability, accessibility, performance or security consideration.
- Produce a small piece of evidence: code, diagram, checklist, test result or improvement note.

14. Data, JSON and APIs

Modern websites often retrieve or submit data through APIs. JSON is a widely used text format for representing structured data through objects, arrays, strings, numbers, booleans and null values.

An API defines how software components communicate. Web APIs often expose endpoints that clients call through HTTP requests.

JavaScript can use fetch-style interfaces to request data asynchronously. Applications should handle loading states, errors, timeouts and unexpected data rather than assuming every request succeeds.

API keys and credentials require secure handling. Secrets that grant privileged access must not be placed in browser-delivered code or public repositories.

Practical Application

- Apply the concept to a fictional SME website or a local training project.
- Identify one usability, accessibility, performance or security consideration.
- Produce a small piece of evidence: code, diagram, checklist, test result or improvement note.

15. Browser Developer Tools and Debugging

Modern browsers provide developer tools for inspecting HTML, CSS, network requests, JavaScript execution, storage, performance and accessibility-related information.

The Elements/Inspector view helps identify which CSS rules apply to an element. The Console displays messages and errors. The Network panel reveals requested resources, response status and timing.

Debugging is a structured process: reproduce the problem, reduce it to a smaller case, inspect evidence, form a hypothesis, test a change and verify the result.

Developers should avoid random changes without understanding. A temporary fix can create new problems elsewhere if the root cause is not identified.

Practical Application

- Apply the concept to a fictional SME website or a local training project.
- Identify one usability, accessibility, performance or security consideration.
- Produce a small piece of evidence: code, diagram, checklist, test result or improvement note.

16. Web Accessibility Foundations

Web accessibility means that websites and digital services can be perceived, operated and understood by people with diverse abilities. It also improves usability in many temporary and situational conditions.

WCAG organises accessibility around four principles: perceivable, operable, understandable and robust. Current projects should consult the current WCAG 2.2 recommendation and applicable legal requirements.

Semantic HTML provides much accessibility automatically. Native buttons, links, form controls and headings should be preferred over custom components when they meet the requirement.

Keyboard access, visible focus, meaningful labels, text alternatives, sufficient contrast and logical heading structure are foundational practices. Accessibility should be tested throughout development, not only at the end.

Practical Application

- Apply the concept to a fictional SME website or a local training project.
- Identify one usability, accessibility, performance or security consideration.
- Produce a small piece of evidence: code, diagram, checklist, test result or improvement note.

17. Accessible Navigation and Interaction

Navigation should be predictable and operable without a mouse. Keyboard users need to reach interactive elements in a logical order and see where focus is located.

Skip links can allow users to bypass repeated navigation and move directly to main content. Landmarks and headings support structural navigation with assistive technology.

Custom menus, dialogs and widgets require careful keyboard and focus behaviour. ARIA can add semantics when native HTML cannot express a required pattern, but incorrect ARIA can make an interface less accessible.

Error messages, dynamic updates and modal dialogs need appropriate focus and status management so that important changes are communicated.

Practical Application

- Apply the concept to a fictional SME website or a local training project.
- Identify one usability, accessibility, performance or security consideration.
- Produce a small piece of evidence: code, diagram, checklist, test result or improvement note.

18. Content Management Systems

A content management system separates much of the content-management process from direct code editing. Common CMS capabilities include page editing, media libraries, menus, themes, plugins, user roles and publishing workflows.

CMS use does not remove the need for web knowledge. Poor themes, excessive plugins, insecure configuration and inaccessible content can create significant problems.

Roles should follow least privilege. Editors may need content permissions without administrative access to plugins, server settings or user management.

Updates, backups and plugin/theme lifecycle management are essential. Unsupported components can become security and compatibility risks.

Practical Application

- Apply the concept to a fictional SME website or a local training project.
- Identify one usability, accessibility, performance or security consideration.
- Produce a small piece of evidence: code, diagram, checklist, test result or improvement note.

19. Domains, DNS, Hosting and Deployment

A domain name is registered through a registrar and mapped through DNS records to services such as web hosting or email. Hosting provides the infrastructure that stores and serves website resources.

Deployment is the process of moving tested website changes into an environment accessible to users. Professional workflows often separate development, testing/staging and production.

DNS changes can take time to propagate because records are cached. Incorrect DNS configuration can make a site or email service unavailable.

HTTPS should be enabled for modern websites. Certificates establish encrypted connections and help users verify the server identity associated with a domain.

Practical Application

- Apply the concept to a fictional SME website or a local training project.
- Identify one usability, accessibility, performance or security consideration.
- Produce a small piece of evidence: code, diagram, checklist, test result or improvement note.

20. Version Control and Git Concepts

Version control records changes to files over time and supports collaboration, rollback and review. Git is a widely used distributed version-control system.

A repository contains project history. A commit records a meaningful set of changes. Branches allow work to develop separately before being reviewed and merged.

Commit messages should explain the purpose of a change. Large unrelated changes in one commit are harder to review and reverse.

Public repositories must not contain passwords, API secrets or confidential information. Removing a secret from the latest file does not necessarily remove it from repository history.

Practical Application

- Apply the concept to a fictional SME website or a local training project.
- Identify one usability, accessibility, performance or security consideration.
- Produce a small piece of evidence: code, diagram, checklist, test result or improvement note.

21. Website Performance

Web performance affects user experience, accessibility, conversion and resource consumption. Slow pages can particularly disadvantage users on mobile networks or older devices.

Large images, excessive JavaScript, blocking resources and unnecessary third-party scripts are common performance problems. Image dimensions and compression should match actual display needs.

Browsers can cache resources so repeated visits require less network transfer. Content delivery networks can serve static resources from geographically distributed locations.

Performance should be measured rather than guessed. Current tools such as Lighthouse and PageSpeed Insights can help identify opportunities, but recommendations should be interpreted in the context of the site.

Practical Application

- Apply the concept to a fictional SME website or a local training project.
- Identify one usability, accessibility, performance or security consideration.
- Produce a small piece of evidence: code, diagram, checklist, test result or improvement note.

22. Search-Friendly Web Development

Search-friendly development starts with useful content and crawlable, semantically structured pages. Clear titles, headings, internal links and descriptive URLs help both users and search systems.

Metadata can provide concise page descriptions and information for social sharing, but metadata cannot compensate for weak page content.

Duplicate URLs and accidental indexing of test pages can create maintenance problems. Canonical links, redirects and indexing controls should be used carefully.

Structured data can help machines understand certain content types, but it should accurately represent visible content and follow current search-engine guidelines.

Practical Application

- Apply the concept to a fictional SME website or a local training project.
- Identify one usability, accessibility, performance or security consideration.
- Produce a small piece of evidence: code, diagram, checklist, test result or improvement note.

23. Web Security Foundations

Web security protects confidentiality, integrity and availability of websites, data and users. Security begins with understanding trust boundaries: browsers, servers, databases, third-party services and users exchange information across different levels of trust.

Input from users, URLs, APIs and external files should be treated as untrusted until validated. Output encoding and context-aware handling reduce injection risks.

Authentication verifies identity; authorisation determines what an authenticated identity may do. Strong passwords, MFA for administrative accounts and least privilege reduce account risk.

Security headers, HTTPS, secure cookies, dependency updates, backups and logging are part of defensive web administration. Security must be maintained throughout the website lifecycle.

Practical Application

- Apply the concept to a fictional SME website or a local training project.
- Identify one usability, accessibility, performance or security consideration.
- Produce a small piece of evidence: code, diagram, checklist, test result or improvement note.

24. Common Web Application Risks

Cross-site scripting can occur when untrusted content is inserted into a page in a way that allows script execution. Prevention depends on safe frameworks, output encoding, sanitisation where appropriate and avoiding unsafe DOM operations.

Injection vulnerabilities occur when untrusted input changes the meaning of commands or queries. Parameterised queries and safe APIs are fundamental server-side controls.

Broken access control allows users to reach resources or actions beyond their permissions. The server must enforce authorisation rather than relying on hidden buttons or client-side logic.

Security misconfiguration, outdated components and exposed secrets are common operational risks. OWASP resources provide current defensive guidance and awareness of major web-application risk categories.

Practical Application

- Apply the concept to a fictional SME website or a local training project.
- Identify one usability, accessibility, performance or security consideration.
- Produce a small piece of evidence: code, diagram, checklist, test result or improvement note.

25. Privacy, Cookies and Responsible Data Collection

Websites may process contact details, account data, analytics identifiers and behavioural information. Developers and site owners should know what data is collected, why it is needed, where it is sent and how long it is retained.

Cookies can support sessions, preferences, analytics and advertising. Their legal treatment depends on purpose, context and jurisdiction. Developers should not assume that a generic banner automatically makes data collection compliant.

Data minimisation reduces both privacy and security risk. A form should not collect personal information simply because a field can be added.

Third-party scripts can transmit information to external providers and affect performance. They should be inventoried and justified.

Practical Application

- Apply the concept to a fictional SME website or a local training project.
- Identify one usability, accessibility, performance or security consideration.
- Produce a small piece of evidence: code, diagram, checklist, test result or improvement note.

26. Testing and Quality Assurance

Web quality assurance checks functionality, content, layout, accessibility, performance, compatibility and security before publication.

Testing should include different viewport sizes, keyboard use, form errors, broken links, missing assets and common browsers relevant to the audience.

Automated tools are useful but incomplete. Accessibility scanners, validators and performance tools cannot replace human testing and judgement.

A release checklist makes quality repeatable. High-risk changes should have rollback plans and backups.

Practical Application

- Apply the concept to a fictional SME website or a local training project.
- Identify one usability, accessibility, performance or security consideration.
- Produce a small piece of evidence: code, diagram, checklist, test result or improvement note.

27. Website Maintenance and Governance

A website is an ongoing service rather than a one-time design product. Content becomes outdated, links break, dependencies change and security updates are released.

Governance defines who owns content, who can publish, who administers the platform, how changes are approved and how incidents are handled.

Content review dates help identify pages that require updating or archiving. Unmaintained pages can mislead users and weaken trust.

Maintenance plans should include backups, restore testing, updates, security monitoring, analytics review, accessibility checks and domain/hosting renewal.

Practical Application

- Apply the concept to a fictional SME website or a local training project.
- Identify one usability, accessibility, performance or security consideration.
- Produce a small piece of evidence: code, diagram, checklist, test result or improvement note.

28. AI-Assisted Web Development

Generative AI can assist with explanations, code drafts, debugging hypotheses, content outlines, accessibility suggestions and test cases. It can accelerate learning and routine development.

AI-generated code can be insecure, outdated, inaccessible or inconsistent with the existing codebase. Developers must understand, test and review code before deployment.

Prompts should provide the relevant context, constraints and desired behaviour without exposing secrets or confidential source code when the service is not approved for such use.

AI is most useful as a supervised assistant. The developer remains responsible for requirements, architecture, security, testing and the final published result.

Practical Application

- Apply the concept to a fictional SME website or a local training project.
- Identify one usability, accessibility, performance or security consideration.
- Produce a small piece of evidence: code, diagram, checklist, test result or improvement note.

29. Practical Lab Series

Lab	Focus	Task	Evidence
Lab 1	First Web Page	Create a valid HTML page with headings, paragraphs, lists, links and images.	HTML file
Lab 2	Semantic Structure	Build header, navigation, main content, article/section and footer structure.	Semantic page
Lab 3	Forms	Create a contact form with labels, required fields and clear errors.	Accessible form
Lab 4	CSS Styling	Create reusable typography, spacing, buttons and card styles.	Stylesheet
Lab 5	Responsive Layout	Build a mobile-first page using Flexbox and Grid.	Responsive page
Lab 6	JavaScript Interaction	Add a menu toggle, disclosure or simple interface interaction.	JS-enhanced page
Lab 7	DOM & Validation	Read form values, validate them and provide accessible feedback.	Validated form
Lab 8	API/JSON	Fetch data from an approved public training API or local JSON file and render it safely.	Data-driven component
Lab 9	Accessibility Audit	Test keyboard navigation, headings, labels, alt text and contrast.	Accessibility report
Lab 10	Performance Review	Inspect network requests and performance; optimise images/scripts.	Before/after report
Lab 11	Security Review	Review a fictional site for exposed secrets, weak access, unsafe input and outdated components.	Security checklist
Lab 12	CMS Administration	Create a content model, roles, update and backup plan for a CMS site.	CMS governance plan
Lab 13	Deployment Plan	Document domain, DNS, HTTPS, staging, production and rollback.	Deployment checklist
Lab 14	Capstone	Build and document a small professional website.	Website + portfolio

30. HTML Starter Pattern

```
<!doctype html>
<html lang="en">
<head>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1">
 <title>Meaningful Page Title</title>
</head>
<body>
 <header>...</header>
 <nav aria-label="Primary">...</nav>
 <main id="main-content">...</main>
 <footer>...</footer>
</body>
</html>
```

Learners should understand what each line contributes rather than copying the pattern mechanically.

31. CSS Starter Pattern

```
* { box-sizing: border-box; }
body { margin: 0; font-family: system-ui, sans-serif; line-height: 1.5; }
.container { width: min(100% - 2rem, 70rem); margin-inline: auto; }
.cards { display: grid; gap: 1rem; grid-template-columns: repeat(auto-fit, minmax(16rem, 1fr)); }
img { max-width: 100%; height: auto; }
```

This pattern illustrates predictable sizing, readable defaults, a constrained content width, responsive Grid and flexible images.

32. JavaScript Starter Pattern

```
const button = document.querySelector('[data-toggle]');
const panel = document.querySelector('[data-panel]');

button?.addEventListener('click', () => {
  const expanded = button.getAttribute('aria-expanded') === 'true';
  button.setAttribute('aria-expanded', String(!expanded));
  panel.hidden = expanded;
});
```

The example connects behaviour to semantic state. Learners should test keyboard operation and initial HTML state.

33. Website Quality Checklist

Area	Core checks
Content	Current, accurate, clear, owned and reviewed
HTML	Semantic structure, logical headings, valid controls
Responsive	Usable at small and large viewport sizes
Accessibility	Keyboard, focus, labels, alternatives, contrast, zoom
Performance	Optimised images, limited unnecessary scripts, measured loading
Security	HTTPS, updates, least privilege, safe input, no exposed secrets
Privacy	Necessary data only, documented third parties and retention
SEO	Meaningful titles, headings, links, crawlable content
Operations	Backups, restore plan, monitoring, renewals, ownership

34. SME Case Studies

Case 1 – Outdated SME Website

A 12-person company has a fixed-width website that is difficult to use on phones.

Task: Create a mobile-first redesign plan covering layout, navigation, content priority and testing.

Case 2 – Inaccessible Form
A training provider's registration form uses placeholders without labels and reports errors only in red.

Task: Identify accessibility barriers and redesign the form.

Case 3 – Slow Homepage
A tourism website loads several large photographs and third-party widgets.

Task: Create a performance investigation plan and prioritise improvements.

Case 4 – CMS Plugin Risk
An SME website uses many outdated plugins installed by different administrators.

Task: Design a plugin inventory, update, backup and removal process.

Case 5 – Exposed API Key
A learner finds a privileged API credential inside public JavaScript.

Task: Explain containment, credential rotation and secure architectural alternatives without attempting to use the credential.

Case 6 – Website Redesign
Management wants a visually modern site but has not defined user goals.

Task: Convert the request into audience, task, content, accessibility and measurable outcome requirements.

35. Capstone Project – Professional SME Website
Create a small multi-page website for a fictional SME/project or an organisation for which you have explicit permission. The site must demonstrate standards-based implementation and documented quality assurance.

1. Project brief, target audience and user goals.
2. Site map and navigation model.
3. Low-fidelity wireframes for key pages.
4. Semantic HTML for at least four page/content types.
5. Reusable CSS design system with responsive layout.
6. At least one meaningful JavaScript interaction.
7. Accessible form with validation and error guidance.
8. Responsive images/media where relevant.
9. Accessibility audit and corrective actions.
10. Performance review and optimisation evidence.
11. Basic SEO/content metadata checklist.
12. Basic web-security and privacy checklist.
13. CMS or maintenance plan, even if the capstone is coded statically.
14. Deployment/hosting plan including HTTPS and backup.
15. README or maintenance documentation.

Capstone Assessment Rubric

Criterion	Weight
Requirements, information architecture and content	10%
Semantic HTML and code quality	15%
CSS, responsive design and visual consistency	15%
JavaScript functionality	10%
Accessibility	15%
Performance and SEO readiness	10%
Security and privacy	10%
Testing and maintenance	10%
Documentation and presentation	5%

36. Final Knowledge Assessment

1. What is the role of HTML?

2. What is the role of CSS?

3. What is the role of JavaScript in a webpage?

4. What does DNS do?

5. Why use semantic HTML?

6. What is responsive design?

7. What is the DOM?

8. Why is client-side validation not a security control?

9. What are the four WCAG principles?

10. Why should secrets not be stored in client-side JavaScript?

11. What is version control?

12. Why is HTTPS important?

13. Why is a CMS plugin update process necessary?

14. Why must AI-generated code be reviewed?

Answer Key

1. To provide structure and semantic meaning for web content.

2. To control presentation and layout.

3. To add logic, behaviour and dynamic interaction.

4. It resolves domain names to network destinations/addresses.

5. It improves meaning, maintainability, accessibility and machine interpretation.

6. Design that adapts layout and interaction to different viewport/device conditions.

7. The browser's object representation of the document that scripts can inspect and modify.

8. It can be bypassed; the server must independently validate input.

9. Perceivable, Operable, Understandable and Robust.

10. Browser-delivered code can be inspected by users.

11. A system for recording and managing changes to files over time.

12. It encrypts transport and authenticates the server through TLS.

13. Unsupported or vulnerable components can create security and compatibility risks.

14. It may be incorrect, insecure, inaccessible or outdated.

37. Career Transition Guide

Starting profile	Possible direction	Portfolio evidence
Web/content user	Website Administrator	CMS plan; content model; maintenance checklist
Digital content staff	Digital Content Specialist	Accessible pages; SEO/content QA; publishing workflow
Beginner coder	Junior Front-End Assistant	HTML/CSS/JS site; Git evidence; testing report
VET learner	Junior Web Developer pathway	Responsive capstone; forms; API/JSON lab
SME IT support	Web Operations Support	Hosting, DNS, HTTPS, backup and security checklist
Career changer	Entry-level Web Creator	Complete capstone portfolio and documentation

38. Glossary

Term	Definition
Browser	Client software used to request and render web resources.
Web Server	System that responds to web requests and delivers resources.
URL	Uniform Resource Locator identifying a web resource.
Domain	Human-readable name used to locate internet services.
DNS	Domain Name System.
HTTP	Hypertext Transfer Protocol.
HTTPS	HTTP protected by TLS.
HTML	HyperText Markup Language.
Semantic HTML	HTML elements used according to their meaning.
CSS	Cascading Style Sheets.
Cascade	Rules determining which CSS declarations apply.
Box Model	Content, padding, border and margin model used in CSS layout.
Flexbox	One-dimensional CSS layout system.
Grid	Two-dimensional CSS layout system.
Responsive Design	Design adapting to different viewport/device conditions.
JavaScript	Programming language widely used for web-page behaviour.

DOM	Document Object Model.
Event	An occurrence such as click, input or submit that code can respond to.
JSON	Text format for structured data.
API	Application Programming Interface.
CMS	Content Management System.
Repository	Version-controlled project storage/history.
Commit	Recorded set of changes in version control.
Accessibility	Design and implementation enabling people with diverse abilities to use a service.
WCAG	Web Content Accessibility Guidelines.
SEO	Search Engine Optimisation.
TLS	Protocol used to protect data in transit, including HTTPS.
Authentication	Verification of identity.
Authorisation	Determination of what an identity is permitted to do.
XSS	Cross-Site Scripting.
Injection	A vulnerability class in which untrusted input changes the meaning of a command/query.
Least Privilege	Granting only permissions necessary for a task.

39. Curated Learning Resources

MDN Web Docs – Learn Web Development
https://developer.mozilla.org/en-US/docs/Learn_web_development
Primary learner reference for HTML, CSS, JavaScript, accessibility and web development concepts.

WHATWG HTML Living Standard
https://html.spec.whatwg.org/
Authoritative current HTML specification; best used as a technical reference rather than a beginner tutorial.

W3C Web Accessibility Initiative – WCAG 2.2
https://www.w3.org/TR/WCAG22/
Current WCAG 2.2 recommendation and conformance criteria.

W3C WAI Tutorials
https://www.w3.org/WAI/tutorials/
Practical accessibility tutorials for page structure, images, tables, forms and related content.

OWASP Top 10
https://owasp.org/www-project-top-ten/
Awareness reference for major web application security risk categories.

OWASP Cheat Sheet Series
https://cheatsheetseries.owasp.org/
Defensive implementation guidance for developers; use according to the technology and context.

web.dev – Learn
https://web.dev/learn/
Practical learning on HTML, CSS, JavaScript, performance, accessibility and modern web practices.

Google Search Central – SEO Starter Guide
https://developers.google.com/search/docs/fundamentals/seo-starter-guide

Foundational search-friendly website guidance.

GitHub Skills
https://skills.github.com/
Optional interactive practice for GitHub and version-control workflows.

WordPress Learn
https://learn.wordpress.org/
Optional CMS learning for learners using WordPress; verify current course/module structure before publication.

40. Certification and Progression

DT-06 should be presented as a ReskillingLab pathway completion record rather than as equivalent to a third-party professional certification. The strongest evidence for entry-level web roles is a demonstrable portfolio combined with clear explanation of design, accessibility, testing and security decisions.

- Learners can progress to advanced JavaScript and front-end frameworks after mastering HTML, CSS and core JavaScript.
- Website administrators can deepen CMS administration, hosting, SEO, analytics and security.
- Developers can progress to back-end development, databases, APIs, DevOps and cloud deployment.
- Accessibility-focused learners can pursue deeper WCAG testing and accessible component development.
- Security-focused learners can continue toward defensive web application security and secure development.
- Any external certification or badge should be checked for current availability, syllabus and cost before publication.

41. Suggested 50–80 Hour Delivery Plan

Component	Indicative hours
Orientation and initial skills check	2
Web foundations and HTML	10
CSS and responsive design	12
JavaScript, DOM and forms	14
APIs, developer tools and debugging	6
Accessibility	7
CMS, hosting and version control	7
Performance, SEO and maintenance	6
Security and privacy	6
Labs, capstone and assessment	10
Total *(can be compressed toward 50 hours by reducing guided practice)*	80 hours;

42. Trainer / Facilitator Notes

- Teach semantic HTML before visual styling and frameworks.
- Require learners to explain code rather than merely copy it.

- Use local or explicitly authorised environments for all development and testing.
- Include keyboard and accessibility checks in every relevant lab.
- Treat security as defensive development: do not use real systems for exploitation exercises.
- Require source attribution and licensing awareness for images, fonts, code and other assets.
- Use browser developer tools to build systematic debugging habits.
- Require learners to document AI assistance and verify AI-generated code.
- Assess the final website on usability, accessibility, maintainability and security as well as appearance.

43. References

Mozilla Developer Network (MDN). Learn Web Development and Web Platform documentation.

WHATWG. HTML Living Standard.

World Wide Web Consortium (W3C). Web Content Accessibility Guidelines (WCAG) 2.2.

World Wide Web Consortium, Web Accessibility Initiative. Tutorials.

OWASP Foundation. OWASP Top 10 and Cheat Sheet Series.

Google. web.dev learning resources and Search Central SEO Starter Guide.

GitHub Skills. Version-control learning resources.

WordPress Learn. CMS learning resources.

44. Website Publishing Structure

For ReskillingLab, DT-06 should be published as modular learning content. Each unit should use a consistent interface: Learn | Practice | Check Your Understanding | Resources.

- Overview & Initial Skills Check
- How the Web Works
- HTML Foundations
- Links, Images & Tables
- Forms
- CSS Foundations
- Typography & Visual Hierarchy
- Flexbox & Grid
- Responsive Design
- JavaScript Foundations
- DOM & Events
- JavaScript Forms
- JSON & APIs
- Developer Tools
- Accessibility Foundations
- Accessible Interaction
- CMS

- Domains, DNS & Hosting
- Version Control
- Performance
- SEO
- Web Security
- Privacy & Cookies
- Testing & QA
- Maintenance & Governance
- AI-Assisted Web Development
- Labs & Cases
- Capstone
- Assessment
- Resources, Progression & Glossary