LEARNHUB / TECHNICAL COURSE LIBRARY

Web Development

Practical examples

Understand the pieces of a modern website: semantic HTML for structure, CSS for presentation, JavaScript for behavior, and responsive layouts for different screens.



Level: Beginner · Category: Programming

Original, beginner-friendly study content designed to accompany the course syllabus and its linked video lesson.

Contents

- How to use this resource
- Learning map
- Chapter 01 · HTML structure and semantics
- Chapter 02 · CSS foundations and layout
- Chapter 03 · JavaScript and interaction
- Chapter 04 · Responsive and accessible interfaces
- Chapter 05 · Project workflow and publishing
- Summary and next steps

Course purpose: Understand the pieces of a modern website: semantic HTML for structure, CSS for presentation, JavaScript for behavior, and responsive layouts for different screens.

How to use this resource

Read the chapter titles first, then study the explanations and examples. Pause at each practice prompt before reading its suggested answer. Write down questions that remain and revisit them during revision.

Learning map



Chapter 01 · HTML structure and semantics

Problem context

Outline a resource card using semantic HTML and identify its title, description, metadata, and action.

Concept to apply

HTML elements describe the meaning and structure of content. Semantic elements communicate the role of a region, such as navigation, main content, or an article.

Worked example

A course page uses a header, navigation, main region, article for the lesson, and a labeled list for learning outcomes.

Walk-through

Start each page with a clear document structure, a single descriptive main heading, and meaningful labels. Use links for navigation and buttons for actions. Alternative text describes informative images; decorative images should not add noisy text.

Check your result

Use an article or list item for the card, a heading for the resource name, a paragraph for its description, metadata text, and an anchor or button with a specific accessible label.

Chapter 02 · CSS foundations and layout

Problem context

Create a style rule for a course badge that remains legible in light and dark themes.

Concept to apply

CSS rules match elements and assign presentation such as color, spacing, typography, and layout. The box model includes content, padding, border, and margin.

Worked example

A navigation bar uses flex alignment for its logo and actions, while a content section uses a responsive grid for course cards.

Walk-through

Use reusable classes and design tokens rather than repeated ad hoc values. Flexbox is useful for one-dimensional alignment; Grid is useful for two-dimensional composition. Check contrast and focus states while styling.

Check your result

Define foreground and background tokens with sufficient contrast, add padding and rounded corners, and provide a visible focus outline when the badge is interactive.

Chapter 03 · JavaScript and interaction

Problem context

Describe the states needed for a video lesson’s play/pause control and how a screen reader should hear the action.

Concept to apply

JavaScript is a programming language used in the browser to transform data and respond to events. The DOM represents the page elements scripts can update.

Worked example

A search box filters course records as a learner types, matching the course name and topic tags.

Walk-through

Use event listeners for interaction, keep state changes explicit, and validate data received from forms. Prefer semantic controls with built-in keyboard behavior instead of clickable generic containers.

Check your result

> Track whether playback is active; change the button label or accessible pressed state to match. Keep focus on the same control after toggling.

Chapter 04 · Responsive and accessible interfaces

Problem context

Review a card layout for a 360-pixel screen and list three changes that improve usability.

Concept to apply

Responsive design adapts presentation to available space and input method. Accessibility makes information and controls usable by people with varied abilities.

Worked example

A desktop course detail page places the video beside the syllabus; on a narrow screen, the video and syllabus stack in reading order.

Walk-through

Begin with the smallest screen, add breakpoints when content needs them, and avoid fixed widths that force horizontal scrolling. Use headings in order, labels for form fields, keyboard navigation, and reduced-motion support.

Check your result

> Reduce columns to one, let controls wrap or stack, and ensure text/buttons remain large enough. Avoid cropping the course title or hiding essential actions.

Chapter 05 · Project workflow and publishing

Problem context

Plan the folders and pages for a small course library and name a practical way to test its build.

Concept to apply

A website project combines source files, assets, and build tooling into pages served from a web origin. A build converts source code into browser-ready assets.

Worked example

A portfolio links project cards to detail pages, shares a common navigation component, and includes an accurate route list for deployment.

Walk-through

Keep components focused and reuse data rather than copying markup. Test routes and missing states, optimize images, and make every action lead somewhere real. A static site can serve public educational content without a backend.

Check your result

Separate course data, shared components, page components, and public assets. Run the type check and production build, then verify representative routes and asset URLs.

Summary and next steps

Web Development brings together 5 connected topics. Review each definition, explain one example without looking, and then try the practice prompt again.

Key ideas to carry forward

HTML structure and semantics: HTML elements describe the meaning and structure of content. Semantic elements communicate the role of a region, such as navigation, main content, or an article.

CSS foundations and layout: CSS rules match elements and assign presentation such as color, spacing, typography, and layout. The box model includes content, padding, border, and margin.

JavaScript and interaction: JavaScript is a programming language used in the browser to transform data and respond to events. The DOM represents the page elements scripts can update.

Responsive and accessible interfaces: Responsive design adapts presentation to available space and input method. Accessibility makes information and controls usable by people with varied abilities.

Project workflow and publishing: A website project combines source files, assets, and build tooling into pages served from a web origin. A build converts source code into browser-ready assets.

A short review routine

1. Recall the main idea from each chapter before rereading. 2. Work one example by hand. 3. Explain a common mistake and how you would catch it. 4. Revisit a difficult topic after a short break.

Course outcomes

- Build semantic pages with HTML
- Style readable, consistent interfaces with CSS
- Use JavaScript to respond to user actions
- Create responsive and accessible project pages

This practical examples is an original LearnHub learning aid. Use the referenced ideas for study, then consult the course video and the rest of the library for additional practice.