Admissions & Support: support@mockattempt.com
Phone / WhatsApp: +91 95400 07917
Course Code: WEBD • **Updated:** 04 Nov 2026
https://mockattempt.com/courses/web-designing-training

WEBD • WEB DESIGNING & FRONTEND UI **BEGINNER TO JOB-READY WEB DESIGNER & UI ENGINEER**

100% LIVE INDUSTRY EXPERT COHORT + INTERNSHIP LETTER

Web Designing Training

Master modern Web Designing from Semantic HTML5, Accessible Forms & Canvas to CSS3 Flexbox, Grid, Responsive Media Queries, Keyframe Animations & Interactive ES6+ JavaScript DOM/Fetch — with runnable code, 396 Topic Mock Tests (18,360 MCQs), and Spoken Voice + MockAttempt AI Interviews at ₹1,499 (Incl. 18% GST).

Duration: 15–17 Weeks • Live + Hands-On Projects (214 Hrs) • **Modules:** 16 • **Live Topics:** 54 • **Subtopics:** 432+ • **Labs & Tasks:** 108 • **Mocks / AI Vivas:** 396 / 396 • **Projects:** 16 Module + 1 Live Client Prod

All-Inclusive Direct Fee (Incl. 18% GST • Zero Hidden Charges): Total Payable: ₹1,499 (Net Tuition ₹1,270 + Included 18% GST ₹229) • **Split Option:** 100% Credit to Job Track • **Effective ROI:** ~₹10/Guided Hr (~₹28/Live Topic)

Faculty: 100% Active Working IT Professionals • **Live Batch Tracks (7:00 AM – 8:00 PM IST Flexible Slots):** Weekday Live Online Batch (Mon – Fri • 1 Hr/Day • 7:00 AM – 8:00 PM) (Monday to Friday (1 Hour Live Session / Day) • 07:00 AM – 08:00 PM IST (Students Can Choose Any 1-Hour Slot Based on Trainer Availability) • 18–20 Weeks Fast-Track) | Weekend Live Online Batch (Sat – Sun • 2 Hrs/Day • 7:00 AM – 8:00 PM) (Saturday & Sunday (2 Hours Live Session / Day) • 07:00 AM – 08:00 PM IST (Students Can Choose Any 2-Hour Slot Based on Trainer Availability) • 20–22 Weeks Weekend Track)

✓ **Trainer LeetCode & HackerRank Mentorship (JavaScript ES6+): Live trainer support to crack LeetCode & HackerRank problems, optimize complexity, and handle edge cases.**

Core Technologies & Engineering Stack: HTML5 Semantics • CSS3 Box Model • Flexbox & CSS Grid • Responsive @media • CSS Animations • JavaScript ES6+ • DOM & Fetch API

1. PROGRAM MASTERY OUTCOMES & ELIGIBILITY

End-to-End Engineering Outcomes:

1. Craft semantic, SEO-friendly, WCAG-accessible HTML5 layouts, forms, tables, media, Canvas, and Web Storage apps
2. Engineer pixel-accurate responsive interfaces using CSS3 Specificity, Box Model, Flexbox, CSS Grid, Media Queries, and GPU Animations
3. Build interactive user experiences with ES6+ JavaScript, Closures, Array pipelines, DOM Event Delegation, Form Validation, and Async Fetch APIs
4. Practice 396 Timed Mock Exams (18,000 HTML/CSS/JS MCQs) and 396 Spoken Voice + MockAttempt AI Technical Interviews
5. Semantic, accessible HTML5 landmarks (`WCAG 2.1 AA`), keyboard navigation, and technical SEO metadata.
6. Crack LeetCode and HackerRank problems in JavaScript ES6+ with live trainer mentorship on algorithmic/query patterns, edge cases, and optimal complexity.
7. Fluid responsive layouts with CSS Grid, Flexbox, custom properties (design tokens), and mobile-first breakpoints.
8. Modern Vanilla ES2024+ JavaScript: DOM state architecture, `async/await` Fetch APIs, debouncing, and event delegation.

Eligibility & Prerequisites: Zero prior coding or design experience required — starts from your very first HTML5 webpage • Works on any laptop/PC with Chrome/Edge DevTools and VS Code • Ideal for B.Tech / BCA / MCA / B.Sc IT students, non-CS beginners, career switchers, and working developers upgrading to 2026 production standards. • Commitment to attend live Weekday or Weekend classes, complete daily hands-on lab checklists, and practice topic-wise Mock Tests & AI Interviews. • Ideal for B.Tech / BCA / MCA / B.Sc IT students, non-CS beginners, career switchers, and working developers upgrading to 2026 production standards.

2. DETAILED ZERO-TO-EXPERT MODULE, TOPIC & SUBTOPIC SYLLABUS (16 MODULES • 54 LIVE TOPICS • 432+ SUBTOPICS)

Progressive Zero-to-Expert Mastery Stages: Modules 1–4: Stage 1: Day-1 Foundations & Core Mental Models (16 Topics / 128 Subtopics) • Modules 5–8: Stage 2: Applied Engineering, Data & API Workflows (14 Topics / 112 Subtopics) • Modules 9–12: Stage 3: Internals, Concurrency, Security & Performance (12 Topics / 96 Subtopics) • Modules 13–16: Stage 4: Production Architecture, Testing, CI/CD & Expert Mastery (12 Topics / 96 Subtopics)

MODULE 1 OF 16 • STAGE 1 • 11 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 1: HTML5 Document Architecture, Building Blocks, Attributes & Typography

Master Standards Mode (<!DOCTYPE html>), DOM tree hierarchy, global/ARIA attributes, viewport meta configuration, and accessible heading structures.

Objectives: ✓ What is HTML5 — Standards Mode, <!DOCTYPE html> & Browser DOM Parsing — How browsers tokenize HTML5 markup into the Document Object Model (DOM), <!DOCTYPE html> vs Quirks Mode, <head> metadata, and responsive viewport setup. • ✓ HTML Building Blocks — Tags, Elements, Block vs Inline & Void Elements — Opening/closing tags, nested DOM trees, self-closing void elements (<img>, <br>, <hr>, <input>, <meta>), and block vs inline flow layout. • ✓ HTML Global Attributes, Boolean Attributes, Custom `data-*` & ARIA — id uniqueness vs reusable class tokens, boolean attributes (disabled, required, hidden, defer), custom data-* dataset API, title, tabindex, and aria-label. • ✓ HTML Headings (`<h1>`–`<h6>`), Paragraphs & Accessible Document Outlines — Semantic heading hierarchy for SEO crawlers and screen-reader rotor navigation, paragraph rhythm, semantic emphasis (/ vs <b>/<i>), and <pre>/<code>. • ✓ Build, verify, and ship Module Capstone: SEO-Optimized Personal Developer & Course Portfolio Landing Page (Pure HTML5).

1.1 What is HTML5 — Standards Mode, <!DOCTYPE html> & Browser DOM Parsing

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: DOCTYPE Standards Mode & DOM tree • Core Syntax, CLI & Production APIs: <!DOCTYPE html>, <html lang="en">, <meta charset="UTF-8">, <meta name="viewport"> • Deep-Dive Mechanics: Meta viewport & UTF-8 charset • Advanced Patterns: Lang attribute & Quirks Mode prevention • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Standards-Compliant HTML5 Responsive Boilerplate • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for What is HTML5

Purpose: To understand and apply What is HTML5 — Standards Mode, <!DOCTYPE html> & Browser DOM Parsing in real Web Designing systems — specifically: How browsers tokenize HTML5 ma...

What Topic Is Next → Next Topic 1.2: HTML Building Blocks — Tags, Elements, Block vs Inline & Void Elements

1.2 HTML Building Blocks — Tags, Elements, Block vs Inline & Void Elements

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Tags vs elements & Void elements • Core Syntax, CLI & Production APIs: , , <p>, <section>, void elements (<img />, <input />, <meta>) • Deep-Dive Mechanics: Block vs inline & DOM nesting rules • Advanced Patterns: Content model & Auto-closing p tag • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Block vs Inline Flow & Clean Element Nesting Demo • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for HTML Building Blocks

Purpose: To understand and apply HTML Building Blocks — Tags, Elements, Block vs Inline & Void Elements in real Web Designing systems — specifically: Opening/closing tags, nested...

What Topic Is Next → Next Topic 1.3: HTML Global Attributes, Boolean Attributes, Custom `data-*` & ARIA

1.3 HTML Global Attributes, Boolean Attributes, Custom `data-*` & ARIA

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Boolean attributes & Data-* dataset • Core Syntax, CLI & Production APIs: id, class, data-*, hidden, disabled, required, aria-label, element.dataset • Deep-Dive Mechanics: Unique id vs class & Aria-label • Advanced Patterns: Tabindex & Hidden attribute • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Custom data-* Attributes & Accessible ARIA Controls • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for HTML Global Attributes

Purpose: To understand and apply HTML Global Attributes, Boolean Attributes, Custom `data-*` & ARIA in real Web Designing systems — specifically: id uniqueness vs reusable class t...

What Topic Is Next → Next Topic 1.4: HTML Headings (`<h1>`-`<h6>`), Paragraphs & Accessible Document Outlines

1.4 HTML Headings (`<h1>`-`<h6>`), Paragraphs & Accessible Document Outlines

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Single H1 SEO & Heading hierarchy • Core Syntax, CLI & Production APIs: <h1> to <h6>, <p>, , , <blockquote>, <pre><code> • Deep-Dive Mechanics: Screen-reader outline & Strong vs b • Advanced Patterns: Em vs i & Blockquote cite • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Accessible SEO Article Heading Outline • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for HTML Headings (`<h1>`–`<h6>`)

Purpose: To understand and apply HTML Headings (`<h1>`–`<h6>`), Paragraphs & Accessible Document Outlines in real Web Designing systems — specifically: Semantic heading hierarchy...

What Topic Is Next → Module 1 Capstone (SEO-Optimized Personal Developer & Course Portfolio Landing Page (Pure HTML5)) → Next Topic 2.1: HTML Anchor (`<a>`) — Navigation, Fragment Links, `download` & `rel="noopener"`

Module 1 Hands-On Capstone Project (100 Marks): SEO-Optimized Personal Developer & Course Portfolio Landing Page (Pure HTML5) — Architect a complete, valid HTML5 developer portfolio document featuring proper OpenGraph/meta head tags, single H1 hierarchy, skip-to-content accessibility link, structured project sections...

MODULE 2 OF 16 • STAGE 1 • 12 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 2: HTML Hyperlinks, Responsive Images, Accessible Data Tables & Lists

Build secure navigation anchors, CLS-free lazy-loaded responsive images (<picture>/srcset), WCAG-accessible data tables, and structured navigation lists.

Objectives: ✓ HTML Anchor (`<a>`) — Navigation, Fragment Links, `download` & `rel="noopener"` — Absolute vs relative URLs, #section fragment scrolling, mailto:/tel: schemes, download attribute, and preventing tabnabbing with rel="noopener noreferrer". • ✓ HTML Responsive Images (`<img>`, `<picture>`, `srcset`), `alt` & Core Web Vitals (CLS/LCP) — Intrinsic aspect-ratio reservation via width/height attributes, meaningful alt vs decorative alt="", loading="lazy" vs fetchpriority="high", and WebP/AVIF <picture>. • ✓ HTML Accessible Data Tables — `<thead>`, `<th scope>`, `colspan` & `<caption>` — Semantic tabular data structure (<caption>, <thead>, <tbody>, <tfoot>), row/column header associations via scope="col" and scope="row", colspan, and rowspan. • ✓ HTML Ordered (`<ol>`), Unordered (`<ul>`) & Description Lists (`<dl>`) — Semantic bulleted navigation lists, numbered step sequences (start, reversed), key-value

glossary/metadata pairs (<dl>, <dt>, <dd>), and nested sublists. • ✓ Build, verify, and ship Module Capstone: Course Comparison Matrix, Media Gallery & Multi-Section Navigation Page.

2.1 HTML Anchor (`<a>`) — Navigation, Fragment Links, `download` & `rel="noopener"` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Rel="noopener noreferrer" & Links vs buttons • Core Syntax, CLI & Production APIs: <a href="...">, target="_blank", rel="noopener noreferrer", download, href="#id" • Deep-Dive Mechanics: Fragment #id scroll & Mailto and tel • Advanced Patterns: Download attribute & Relative vs absolute URLs • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Secure External, Tel & In-Page Anchor Navigation • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for HTML Anchor (`<a>`)

Purpose: To understand and apply HTML Anchor (`<a>`) — Navigation, Fragment Links, `download` & `rel="noopener"` in real Web Designing systems — specifically: Absolute vs relative...

What Topic Is Next → Next Topic 2.2: HTML Responsive Images (`<img>`, `<picture>`, `srcset`), `alt` & Core Web Vitals (CLS/LCP)

2.2 HTML Responsive Images (`<img>`, `<picture>`, `srcset`), `alt` & Core Web Vitals (CLS/LCP) [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: CLS width & height & Srcset & picture • Core Syntax, CLI & Production APIs: <img src alt width height loading="lazy">, <picture>, <source srcset sizes>, <figure>/<figca... • Deep-Dive Mechanics: Loading="lazy" & Fetchpriority="high" • Advanced Patterns: Meaningful alt vs alt="" & WebP & AVIF • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: CLS-Free Responsive Picture Element with Caption • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for HTML Responsive Images (`<img>`, `<picture>`, `srcse...

Purpose: To understand and apply HTML Responsive Images (`<img>`, `<picture>`, `srcset`), `alt` & Core Web Vitals (CLS/LCP) in real Web Designing systems — specifically: Intrinsic...

What Topic Is Next → Next Topic 2.3: HTML Accessible Data Tables — `<thead>`, `<th scope>`, `colspan` & `<caption>`

2.3 HTML Accessible Data Tables — `<thead>`, `<th scope>`, `colspan` & `<caption>` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Th scope="col" & Caption accessibility • Core Syntax, CLI & Production APIs: <table>, <caption>, <thead>, <tbody>, <tfoot>, <th scope="col">, colspan, rowspan • Deep-Dive Mechanics: Thead tbody tfoot & Colspan & rowspan • Advanced Patterns: Tabular data vs layout & Responsive overflow-x wrapper • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Accessible Course Fee & Assessment Comparison Table • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for HTML Accessible Data Tables

Purpose: To understand and apply HTML Accessible Data Tables — `<thead>`, `<th scope>`, `colspan` & `<caption>` in real Web Designing systems — specifically: Semantic tabular data...

What Topic Is Next → Next Topic 2.4: HTML Ordered (`<ol>`), Unordered (`<ul>`) & Description Lists (`<dl>`)

2.4 HTML Ordered (`<ol>`), Unordered (`<ul>`) & Description Lists (`<dl>`) [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Ul vs ol vs dl & Direct li child rule • Core Syntax, CLI & Production APIs: <ul>, <ol start="1" reversed>, <li>, <dl>, <dt>, <dd> • Deep-Dive Mechanics: Nested lists inside li & Dt & dd glossary • Advanced Patterns: Ol start & reversed & Nav ul menus • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Semantic Navigation Menu & Course Metadata Description List • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for HTML Ordered (`<ol>`)

Purpose: To understand and apply HTML Ordered (`<ol>`), Unordered (`<ul>`) & Description Lists (`<dl>`) in real Web Designing systems — specifically: Semantic bulleted navigation...

What Topic Is Next → Module 2 Capstone (Course Comparison Matrix, Media Gallery & Multi-Section Navigation Page) → Next Topic 3.1: HTML5 Semantic Landmarks (`<header>`, `<nav>`, `<main>`, `<article>`, `<section>`, `<footer>`)

Module 2 Hands-On Capstone Project (100 Marks): Course Comparison Matrix, Media Gallery & Multi-Section Navigation Page — Create a rich Academy Course Catalog & Comparison page featuring an accessible Jump-Link Navbar, responsive `<picture>` WebP course banners with zero layout shift, and a full `<table aria-de...

MODULE 3 OF 16 • STAGE 1 • 13 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 3: HTML5 Semantic Landmarks, Accessible Forms, `<canvas>` Graphics & Web Storage

Replace div-soup with semantic landmarks, engineer accessible HTML5 validated forms, draw 2D graphics on `<canvas>`, and persist client state in `localStorage`.

Objectives: ✓ HTML5 Semantic Landmarks (`<header>`, `<nav>`, `<main>`, `<article>`, `<section>`, `<footer>`) — Eliminating "div soup", ARIA landmark equivalency, <article> (independently syndicatable) vs <section> (thematic grouping), <aside>, <time datetime>, and <dialog>. • ✓ HTML5 Forms, Input Types, `<label>` Accessibility & Constraint Validation — GET vs POST, <label for>, <fieldset>/<legend>, HTML5 inputs (email, tel, number, date, range), required/pattern/minlength, autocomplete, and FormData. • ✓ HTML5 `<canvas>` 2D Drawing API vs Scalable Vector Graphics (`<svg>`) — Immediate-mode raster pixel drawing (`getContext("2d")`, fillRect, arc, strokeText, requestAnimationFrame) vs retained-mode DOM-accessible `<svg>` vectors. • ✓ HTML5 Web Storage (`localStorage` vs `sessionStorage`) & Security Rules — Origin-scoped ~5MB key-value string persistence, JSON.stringify/JSON.parse serialization, tab-scoped sessionStorage vs persistent localStorage, and XSS token risks. • ✓ Build, verify, and ship Module Capstone: Interactive Admission Lead Form, Signature Canvas & Persistent Draft Saver.

3.1 HTML5 Semantic Landmarks (`<header>`, `<nav>`, `<main>`, `<article>`, `<section>`, `<footer>`) [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Single visible <main> & Article vs section • Core Syntax, CLI & Production APIs: <header>, <nav>, <main>, <article>, <section>, <aside>, <footer>, <dialog>, <time> • Deep-Dive Mechanics: ARIA landmark roles & Avoiding div soup • Advanced Patterns: <time datetime> & Native <dialog> • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Complete Semantic HTML5 Application Shell • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for HTML5 Semantic Landmarks (`<header>`, `<nav>`, `<mai...

Purpose: To understand and apply HTML5 Semantic Landmarks (`<header>`, `<nav>`, `<main>`, `<article>`, `<section>`, `<footer>`) in real Web Designing systems — specifically: Elimi...

What Topic Is Next → Next Topic 3.2: HTML5 Forms, Input Types, `<label>` Accessibility & Constraint Validation

3.2 HTML5 Forms, Input Types, `<label>` Accessibility & Constraint Validation [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Label for binding & Name attribute submission • Core Syntax, CLI & Production APIs: <form method="POST">, <label for="id">, <input type="email" required pattern="...">, <fields... • Deep-Dive Mechanics: Required & pattern & Fieldset & legend • Advanced Patterns: GET vs POST & Constraint Validation API • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Accessible HTML5 Validated Lead Inquiry Form • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for HTML5 Forms

Purpose: To understand and apply HTML5 Forms, Input Types, `<label>` Accessibility & Constraint Validation in real Web Designing systems — specifically: GET vs POST, <label for>,...

What Topic Is Next → Next Topic 3.3: HTML5 `<canvas>` 2D Drawing API vs Scalable Vector Graphics (`<svg>`)

3.3 HTML5 `<canvas>` 2D Drawing API vs Scalable Vector Graphics (`<svg>`) [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: GetContext("2d") & Canvas raster vs SVG vector • Core Syntax, CLI & Production APIs: canvas.getContext("2d"), ctx.fillRect(), ctx.beginPath(), ctx.arc(), window.devicePixelRatio • Deep-Dive Mechanics: HTML width/height vs CSS & DevicePixelRatio HiDPI • Advanced Patterns: BeginPath & arc & Fallback inner text • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: 2D Canvas Progress Score Ring Renderer • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for HTML5 `<canvas>` 2D Drawing API vs Scalable Vector G...

Purpose: To understand and apply HTML5 `<canvas>` 2D Drawing API vs Scalable Vector Graphics (`<svg>`) in real Web Designing systems — specifically: Immediate-mode raster pixel dr...

What Topic Is Next → Next Topic 3.4: HTML5 Web Storage (`localStorage` vs `sessionStorage`) & Security Rules

3.4 HTML5 Web Storage (`localStorage` vs `sessionStorage`) & Security Rules [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: LocalStorage vs sessionStorage & JSON.stringify & parse • Core Syntax, CLI & Production APIs: localStorage.setItem(k, JSON.stringify(v)), JSON.parse(localStorage.getItem(k)), removeItem(... • Deep-Dive Mechanics: [object Object] bug & Same-origin isolation • Advanced Patterns: XSS HttpOnly contrast & Storage event • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Safe JSON Theme & Progress Persistence Helper • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for HTML5 Web Storage (`localStorage` vs `sessionStorage...

Purpose: To understand and apply HTML5 Web Storage (`localStorage` vs `sessionStorage`) & Security Rules in real Web Designing systems — specifically: Origin-scoped ~5MB key-value...

What Topic Is Next → Module 3 Capstone (Interactive Admission Lead Form, Signature Canvas & Persistent Draft Saver) → Next Topic 4.1: What is CSS3 — Cascade Order, External Stylesheets & CSS Custom Properties (`var()`)

Module 3 Hands-On Capstone Project (100 Marks): Interactive Admission Lead Form, Signature Canvas & Persistent Draft Saver — Build an accessible Course Enrollment & Lead Inquiry Portal in HTML5 + vanilla JS that validates email/phone/batch inputs natively, lets candidates sketch or preview a badge on `<canvas>`, a...

MODULE 4 OF 16 • STAGE 1 • 13 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 4: CSS3 Foundations, Specificity Cascade, Box Model & Visual Surfaces

Master CSS cascade layers, exact specificity calculation, `box-sizing: border-box`, margin collapse rules, CSS Custom Properties (`--var`), gradients, and shadows.

Objectives: ✓ What is CSS3 — Cascade Order, External Stylesheets & CSS Custom Properties (`var()`) — External `<link rel="stylesheet">` vs internal `<style>` vs inline styles, CSSOM construction (render-blocking resource), inheritance, and `:root` CSS variables. • ✓ CSS Selectors, Combinators (`>`, `+`, `~`) & Specificity Calculation `(ID, Class, Tag)` — Class, ID, attribute (`[type="email"]`), child (`>`), adjacent sibling (`+`), general sibling (`~`), modern `:is()`/`:where()`/`:has()`, and specificity weights. • ✓ CSS Box Model — `box-sizing: border-box`, Padding, Border & Margin Collapsing — Content, padding, border, and margin layers; `content-box` vs `border-box` math; vertical margin collapsing; and logical properties (`margin-inline`, `padding-block`). • ✓ CSS Backgrounds, Linear/Radial Gradients, `backdrop-filter` & Elevation Shadows — `background-size: cover` vs `contain`, multi-layer backgrounds, `linear-gradient` / `radial-gradient`, `object-fit: cover`, and subtle multi-layer `box-shadow`. • ✓ Build, verify, and ship Module Capstone: Design-Token Pricing Card & Glassmorphism Hero Component System.

4.1 What is CSS3 — Cascade Order, External Stylesheets & CSS Custom Properties (`var()`) [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: CSSOM render-blocking & External vs inline CSS • Core Syntax, CLI & Production APIs: <link rel="stylesheet" href="styles.css">, :root { --primary: #2563eb; }, var(--primary, fal... • Deep-Dive Mechanics: Cascade & inheritance & CSS Custom Properties var() • Advanced Patterns: Avoiding !important & Dark mode tokens • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Themeable Design Tokens with CSS Custom Properties • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for What is CSS3

Purpose: To understand and apply What is CSS3 — Cascade Order, External Stylesheets & CSS Custom Properties (`var()`) in real Web Designing systems — specifically: External `<link...

What Topic Is Next → Next Topic 4.2: CSS Selectors, Combinators (`>`, `+`, `~`) & Specificity Calculation `(ID, Class, Tag)`

4.2 CSS Selectors, Combinators (`>`, `+`, `~`) & Specificity Calculation `(ID, Class, Tag)` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Specificity (ID, Class, Tag) & Right-to-left matching • Core Syntax, CLI & Production APIs: .class, #id, [data-state="open"], parent > child, h2 + p, :where(), :is(), :has() • Deep-Dive Mechanics: Child > vs descendant & Adjacent sibling + • Advanced Patterns: :where() zero specificity & :has() parent selector • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Clean Low-Specificity BEM & Combinator Styling • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for CSS Selectors

Purpose: To understand and apply CSS Selectors, Combinators (`>`, `+`, `~`) & Specificity Calculation `(ID, Class, Tag)` in real Web Designing systems — specifically: Class, ID, a...

What Topic Is Next → Next Topic 4.3: CSS Box Model — `box-sizing: border-box`, Padding, Border & Margin Collapsing

4.3 CSS Box Model — `box-sizing: border-box`, Padding, Border & Margin Collapsing [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Box-sizing: border-box & Content-box overflow bug • Core Syntax, CLI & Production APIs: box-sizing: border-box, padding, border, margin, outline-offset, padding-inline • Deep-Dive Mechanics: Vertical margin collapse & Outline vs border • Advanced Patterns: Logical inline/block props & Margin: 0 auto centering • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Universal border-box Reset & Predictable Card Sizing • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for CSS Box Model

Purpose: To understand and apply CSS Box Model — `box-sizing: border-box`, Padding, Border & Margin Collapsing in real Web Designing systems — specifically: Content, padding, bord...

What Topic Is Next → Next Topic 4.4: CSS Backgrounds, Linear/Radial Gradients, `backdrop-filter` & Elevation Shadows

4.4 CSS Backgrounds, Linear/Radial Gradients, `backdrop-filter` & Elevation Shadows [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Background-size: cover & Linear-gradient overlay • Core Syntax, CLI & Production APIs: background: linear-gradient(...), background-size: cover, object-fit: cover, box-shadow • Deep-Dive Mechanics: Fallback background-color & Object-fit: cover • Advanced Patterns: Layered box-shadow & WCAG 4.5:1 contrast • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Accessible Gradient-Overlay Hero Banner • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for CSS Backgrounds

Purpose: To understand and apply CSS Backgrounds, Linear/Radial Gradients, `backdrop-filter` & Elevation Shadows in real Web Designing systems — specifically: `background-size: co...

What Topic Is Next → Module 4 Capstone (Design-Token Pricing Card & Glassmorphism Hero Component System) → Next Topic 5.1: CSS `display` Property — `block`, `inline`, `inline-block`, `none` vs `visibility: hidden`

Module 4 Hands-On Capstone Project (100 Marks): Design-Token Pricing Card & Glassmorphism Hero Component System — Design a reusable CSS3 Design-Token Component Sheet featuring CSS Custom Properties (`:root` light/dark theme tokens), `border-box` Course Cards with gradient badges, accessible contrast rat...

MODULE 5 OF 16 • STAGE 2 • 13 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 5: CSS Display Modes, Positioning, Stacking Contexts (`z-index`) & Pseudo-Selectors

Control document flow with `display`, anchor sticky headers and modal overlays with `position`, debug `z-index` stacking contexts, and craft interactive `:focus-visible` states.

Objectives: ✓ CSS `display` Property — `block`, `inline`, `inline-block`, `none` vs `visibility: hidden` — Normal flow box generation, why vertical width/height fail on `inline` spans, `inline-block` alignment, and accessibility/layout differences of `display: none` vs `visibility: hidden` vs `opacity: 0`. • ✓ CSS Positioning — `static`, `relative`, `absolute`, `fixed` & `sticky` Mechanics — Containing blocks, anchoring `position: absolute` children inside `position: relative` parents, viewport-pinned `fixed` drawers, and scroll-threshold `position: sticky; top: 0`. • ✓ CSS `z-index` & Stacking Contexts (`isolation: isolate`, `transform`, `opacity`) — Why `z-index` ignores `position: static` (except Flex/Grid items), what creates a new Stacking Context (`opacity < 1`, `transform`, `filter`, `isolation: isolate`), and clean z-index scales. • ✓ CSS Pseudo-Classes (`:hover`, `:focus-visible`, `:nth-child`) & Pseudo-Elements (`::before`, `::after`) — Interactive states (`:hover`, `:active`, `:focus-visible`, `:focus-within`), form states (`:checked`, `:invalid`), structural `:nth-child(even)`, and `::before`/`::after` decorative elements. • ✓ Build, verify, and ship Module Capstone: Sticky Glass Navbar, Notification Badge, Accessible Tooltip & Modal Overlay Kit.

5.1 CSS `display` Property — `block`, `inline`, `inline-block`, `none` vs `visibility: hidden` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Inline ignores width/height & Inline-block • Core Syntax, CLI & Production APIs: display: block | inline | inline-block | flex | grid | none, visibility: hidden • Deep-Dive Mechanics: Display: none vs visibility: hidden & Opacity: 0 & pointer-events • Advanced Patterns: .sr-only pattern & Flow layout • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Pill Badge (`inline-flex`) & Accessible `.sr-only` Helper • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for CSS `display` Property

Purpose: To understand and apply CSS `display` Property — `block`, `inline`, `inline-block`, `none` vs `visibility: hidden` in real Web Designing systems — specifically: Normal fl...

What Topic Is Next → Next Topic 5.2: CSS Positioning — `static`, `relative`, `absolute`, `fixed` & `sticky` Mechanics

5.2 CSS Positioning — `static`, `relative`, `absolute`, `fixed` & `sticky` Mechanics [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Relative containing block & Absolute out-of-flow • Core Syntax, CLI & Production APIs: position: relative | absolute | fixed | sticky, inset: 0, top / right / bottom / left • Deep-Dive Mechanics: Sticky + top: 0 & Overflow: hidden breaks sticky • Advanced Patterns: Inset: 0 shorthand & Fixed viewport overlay • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Sticky Header & Relative Card Corner Ribbon • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for CSS Positioning

Purpose: To understand and apply CSS Positioning — `static`, `relative`, `absolute`, `fixed` & `sticky` Mechanics in real Web Designing systems — specifically: Containing blocks,...

What Topic Is Next → Next Topic 5.3: CSS `z-index` & Stacking Contexts (`isolation: isolate`, `transform`, `opacity`)

5.3 CSS `z-index` & Stacking Contexts (`isolation: isolate`, `transform`, `opacity`) [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Stacking context & Transform/opacity trap • Core Syntax, CLI & Production APIs: z-index, isolation: isolate, stacking context rules • Deep-Dive Mechanics: Isolation: isolate & Z-index on static fails • Advanced Patterns: Flex/Grid item z-index & Tokenized z-scale • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Predictable Tokenized Z-Index Scale & Isolated Card • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for CSS `z-index`

Purpose: To understand and apply CSS `z-index` & Stacking Contexts (`isolation: isolate`, `transform`, `opacity`) in real Web Designing systems — specifically: Why `z-index` ignor...

What Topic Is Next → Next Topic 5.4: CSS Pseudo-Classes (`:hover`, `:focus-visible`, `:nth-child`) & Pseudo-Elements (`::before`, `::after`)

5.4 CSS Pseudo-Classes (`:hover`, `:focus-visible`, `:nth-child`) & Pseudo-Elements (`::before`, `::after`) [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: :focus-visible accessibility & ::before content: "" • Core Syntax, CLI & Production APIs: :hover, :focus-visible, :focus-within, :nth-child(2n), ::before, ::after, ::placeholder • Deep-Dive Mechanics: :nth-child striping & :focus-within • Advanced Patterns: :checked & :invalid & LVHA link order • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Keyboard-Accessible Button & Decorative Pseudo-Element Accent • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for CSS Pseudo-Classes (`:hover`, `:focus-visible`, `:nt...

Purpose: To understand and apply CSS Pseudo-Classes (`:hover`, `:focus-visible`, `:nth-child`) & Pseudo-Elements (`::before`, `::after`) in real Web Designing systems — specifical...

What Topic Is Next → Module 5 Capstone (Sticky Glass Navbar, Notification Badge, Accessible Tooltip & Modal Overlay Kit) → Next Topic 6.1: CSS Flexbox — Main vs Cross Axis, `justify-content`, `align-items`, `gap` & `flex` Shorthand

Module 5 Hands-On Capstone Project (100 Marks): Sticky Glass Navbar, Notification Badge, Accessible Tooltip & Modal Overlay Kit — Build an interactive UI Chrome Kit featuring a `position: sticky` top navigation bar, corner notification count badge (`::after` / `absolute`), keyboard-accessible `:focus-visible` button st...

MODULE 6 OF 16 • STAGE 2 • 15 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 6: Responsive Web Design — CSS Flexbox, CSS Grid, Media Queries & GPU Animations

Build fluid 1D & 2D responsive layouts with Flexbox and CSS Grid (`auto-fit`/`minmax`), mobile-first `@media` + `clamp()`, and 60fps GPU `@keyframes` motion.

Objectives: ✓ CSS Flexbox — Main vs Cross Axis, `justify-content`, `align-items`, `gap` & `flex` Shorthand — 1D flexible box layout, `flex-direction` axis swapping, `flex-wrap: wrap`, `gap`, `flex: 1 1 0%` (`flex-grow`, `flex-shrink`, `flex-basis`), and `min-width: 0` text truncation. • ✓ CSS Grid Layout — `fr` Units, `minmax()`, `auto-fit` vs `auto-fill` & Named Grid Areas — 2D rows-and-columns layout, fractional `fr` tracks, zero-media-query responsive card grids (`repeat(auto-fit, minmax(260px, 1fr))`), `grid-template-areas`, and `grid-column: span 2`. • ✓ Responsive `@media` Queries, Mobile-First Breakpoints, Container Queries & Fluid `clamp()` — Mobile-first `min-width` progression, fluid typography `clamp(min, preferred, max)`, `@container` component queries, `prefers-color-scheme`, and `prefers-reduced-motion`. • ✓ CSS Transitions, 2D/3D Transforms & 60fps GPU `@keyframes` Animations — `transition` timing functions, `transform: translate3d/scale/rotate`, `@keyframes` + `animation-fill-mode: forwards`, compositor-thread properties vs layout thrashing, and accessibility. • ✓ Build, verify, and ship Module Capstone: Responsive SaaS Bento Dashboard, Fluid Pricing Grid & Animated Hero Showcase.

6.1 CSS Flexbox — Main vs Cross Axis, `justify-content`, `align-items`, `gap` & `flex` Shorthand [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Main vs cross axis & Justify-content & align-items • Core Syntax, CLI & Production APIs: display: flex, justify-content, align-items, flex-wrap, gap, flex: 1 • Deep-Dive Mechanics: Flex: grow shrink basis & Min-width: 0 truncation • Advanced Patterns: Flex-wrap & gap & Axis swap on column • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Responsive Flexbox Navbar & Equal-Height Action Row • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for CSS Flexbox

Purpose: To understand and apply CSS Flexbox — Main vs Cross Axis, `justify-content`, `align-items`, `gap` & `flex` Shorthand in real Web Designing systems — specifically: 1D flex...

What Topic Is Next → Next Topic 6.2: CSS Grid Layout — `fr` Units, `minmax()`, `auto-fit` vs `auto-fill` & Named Grid Areas

6.2 CSS Grid Layout — `fr` Units, `minmax()`, `auto-fit` vs `auto-fill` & Named Grid Areas [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Fr fractional unit & Repeat(auto-fit, minmax()) • Core Syntax, CLI & Production APIs: display: grid, grid-template-columns: repeat(auto-fit, minmax(260px, 1fr)), gap, grid-templa... • Deep-Dive Mechanics: Auto-fit vs auto-fill & Minmax(0, 1fr) blowout fix • Advanced Patterns: Grid-template-areas & 2D Grid vs 1D Flexbox • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Intrinsic Responsive Card Grid (Zero Media Queries) • Idiomatic WEBD

Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for CSS Grid Layout

Purpose: To understand and apply CSS Grid Layout — `fr` Units, `minmax()`, `auto-fit` vs `auto-fill` & Named Grid Areas in real Web Designing systems — specifically: 2D rows-and-c...

What Topic Is Next → Next Topic 6.3: Responsive `@media` Queries, Mobile-First Breakpoints, Container Queries & Fluid `clamp()`

6.3 Responsive `@media` Queries, Mobile-First Breakpoints, Container Queries & Fluid `clamp()` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Mobile-first min-width & Fluid clamp() typography • Core Syntax, CLI & Production APIs: @media (min-width: 768px), clamp(1.5rem, 4vw, 3rem), @container, prefers-reduced-motion • Deep-Dive Mechanics: Rem breakpoints & Prefers-reduced-motion • Advanced Patterns: Prefers-color-scheme & @container queries • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Mobile-First Layout & Fluid `clamp()` Headline • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for Responsive `@media` Queries

Purpose: To understand and apply Responsive `@media` Queries, Mobile-First Breakpoints, Container Queries & Fluid `clamp()` in real Web Designing systems — specifically: Mobile-fi...

What Topic Is Next → Next Topic 6.4: CSS Transitions, 2D/3D Transforms & 60fps GPU `@keyframes` Animations

6.4 CSS Transitions, 2D/3D Transforms & 60fps GPU `@keyframes` Animations [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: GPU transform & opacity & Avoiding top/left reflow • Core Syntax, CLI & Production APIs: transition: transform 200ms ease, @keyframes, animation: fadeUp 400ms ease forwards • Deep-Dive Mechanics: @keyframes & forwards & Cubic-bezier easing • Advanced Patterns: Prefers-reduced-motion & Transition vs animation • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: 60fps GPU Card Lift & Accessible Reduced-Motion Guard • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for CSS Transitions

Purpose: To understand and apply CSS Transitions, 2D/3D Transforms & 60fps GPU `@keyframes` Animations in real Web Designing systems — specifically: `transition` timing functions,...

What Topic Is Next → Module 6 Capstone (Responsive SaaS Bento Dashboard, Fluid Pricing Grid & Animated Hero Showcase) → Next Topic 7.1: Features of Modern JavaScript (ES6+), `<script defer>` vs `async` & V8 Engine Basics

Module 6 Hands-On Capstone Project (100 Marks): Responsive SaaS Bento Dashboard, Fluid Pricing Grid & Animated Hero Showcase — Design and code a mobile-first (360px -> 768px -> 1440px) SaaS Academy Landing Page combining a Flexbox Navbar, a 2D CSS Grid Bento Feature Showcase (`auto-fit`/`minmax`), fluid `clamp()` ty...

MODULE 7 OF 16 • STAGE 2 • 13 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 7: JavaScript Foundations, Block Scoping (`let`/`const`), Data Types & Operators

Understand `<script defer>`, V8 execution contexts, `var` vs `let`/`const` Temporal Dead Zone (TDZ), strict equality (`===`), optional chaining (`?.`), and nullish coalescing (`??`).

Objectives: ✓ Features of Modern JavaScript (ES6+), `<script defer>` vs `async` & V8 Engine Basics — JIT-compiled dynamic language, first-class functions, single-threaded event loop, and how `<script defer>` vs `<script async>` vs blocking `<script>` affect HTML parsing. • ✓ JavaScript `var` vs `let` vs `const` — Block Scope, Hoisting & Temporal Dead Zone (TDZ) — Function-scoped `var` hoisting (`undefined`) vs block-scoped `let`/`const`, Temporal Dead Zone (`ReferenceError`), `const` reference immutability vs object property mutation, and `Object.freeze()`. • ✓ JavaScript Data Types, `typeof` Quirks, `===` vs `==`, Optional Chaining (`?.`) & `??` — 7 primitives (`string`, `number`, `bigint`, `boolean`, `undefined`, `symbol`, `null`) + `object`, historic `typeof null === "object"`, `NaN !== NaN`, `===`, `?.`, and `??` vs `||`. • ✓ Build, verify, and ship Module Capstone: Interactive Course Fee, GST, EMI & Scholarship Calculator Widget.

7.1 Features of Modern JavaScript (ES6+), `<script defer>` vs `async` & V8 Engine Basics [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: <script defer> vs async & ES Modules type="module" • Core Syntax, CLI & Production APIs: <script type="module" src="app.js">, <script defer>, "use strict", console.table() • Deep-Dive Mechanics: V8 JIT compilation & First-class functions • Advanced Patterns: Strict mode & Non-blocking HTML parse • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: ES6 Template Literals & Indian Rupee Currency Formatter • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for Features of Modern JavaScript (ES6+)

Purpose: To understand and apply Features of Modern JavaScript (ES6+), `<script defer>` vs `async` & V8 Engine Basics in real Web Designing systems — specifically: JIT-compiled dy...

What Topic Is Next → Next Topic 7.2: JavaScript `var` vs `let` vs `const` — Block Scope, Hoisting & Temporal Dead Zone (TDZ)

7.2 JavaScript `var` vs `let` vs `const` — Block Scope, Hoisting & Temporal Dead Zone (TDZ) [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Block scope vs function scope & Temporal Dead Zone (TDZ) • Core Syntax, CLI & Production APIs: const, let, Object.freeze(), block scope { ... } • Deep-Dive Mechanics: Const reference vs Object.freeze & Hoisting behavior • Advanced Patterns: For(let i) closure fix & Window global pollution • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Block Scope Isolation & `Object.freeze()` Immutability Check • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for JavaScript `var` vs `let` vs `const`

Purpose: To understand and apply JavaScript `var` vs `let` vs `const` — Block Scope, Hoisting & Temporal Dead Zone (TDZ) in real Web Designing systems — specifically: Function-sco...

What Topic Is Next → Next Topic 7.3: JavaScript Data Types, `typeof` Quirks, `===` vs `==`, Optional Chaining (`?.`) & `??`

7.3 JavaScript Data Types, `typeof` Quirks, `===` vs `==`, Optional Chaining (`?.`) & `??` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: 7 primitives + object & Typeof null === "object" • Core Syntax, CLI & Production APIs: ===, typeof, Array.isArray(), Number.isNaN(), obj?.prop, count ?? 0 • Deep-Dive Mechanics: === vs == coercion & ?? vs || falsy zero • Advanced Patterns: Optional chaining ?. & Number.isNaN() • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Safe Nullish Coalescing (`??`) & Optional Chaining (`?.`) Guard • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for JavaScript Data Types

Purpose: To understand and apply JavaScript Data Types, `typeof` Quirks, `===` vs `==`, Optional Chaining (`?.`) & `??` in real Web Designing systems — specifically: 7 primitives...

What Topic Is Next → Module 7 Capstone (Interactive Course Fee, GST, EMI & Scholarship Calculator Widget) → Next Topic 8.1: JavaScript Arrow Functions (`=>`), Lexical `this`, Rest/Spread (`...`) & Destructuring

Module 7 Hands-On Capstone Project (100 Marks): Interactive Course Fee, GST, EMI & Scholarship Calculator Widget — Build an interactive real-time Course Fee & EMI Calculator in HTML5 + CSS3 + ES6 JavaScript supporting coupon codes, 18% GST breakdown, 3/6/12-month EMI computation, and `Intl.NumberFormat("...

MODULE 8 OF 16 • STAGE 2 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 8: JavaScript Functions, Arrow Lexical `this`, Closures & Array Functional Pipelines

Master arrow functions vs lexical `this`, private stateful closures (debounce/throttle), and immutable `map`/`filter`/`reduce` data transformations.

Objectives: ✓ JavaScript Arrow Functions (`=>`), Lexical `this`, Rest/Spread (`...`) & Destructuring — Concise implicit-return arrow functions, why arrow functions inherit `this` lexically from enclosing scope, no `arguments` or `new` constructor, rest parameters `(...args)`, and destructuring. • ✓ JavaScript Closures, Lexical Environment, Private State & `debounce()` / `once()` Patterns — How inner functions retain live references to outer lexical environments after the outer function returns; data privacy, `once()`, `memoize()`, and UI input `debounce()`. • ✓ JavaScript Array Methods — `map`, `filter`, `reduce`, `find`, `some`/`every` & Immutability — Non-mutating pipelines (`map`, `filter`, `reduce`, `find`, `some`, `every`, `flatMap`, `toSorted`) vs in-place mutating traps (`sort`, `splice`, `reverse`, `push`). • ✓ Build, verify, and ship Module Capstone: Live Debounced Course Search, Multi-Tag Filter & Cart Totalizer Engine.

8.1 JavaScript Arrow Functions (`=>`), Lexical `this`, Rest/Spread (`...`) & Destructuring [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Lexical this binding & X => ({ key: x }) parens • Core Syntax, CLI & Production APIs: (a, b) => a + b, ({ title, fee = 999 }) => ..., ...rest, ...spread • Deep-Dive Mechanics: No new constructor & Rest ...args vs spread • Advanced Patterns: Object destructuring defaults & Callbacks & timers • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Destructuring Defaults, Rest Aggregation & Implicit Object Return • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for JavaScript Arrow Functions (`=>`)

Purpose: To understand and apply JavaScript Arrow Functions (`=>`), Lexical `this`, Rest/Spread (`...`) & Destructuring in real Web Designing systems — specifically: Concise impli...

What Topic Is Next → Next Topic 8.2: JavaScript Closures, Lexical Environment, Private State & `debounce()` / `once()` Patterns

8.2 JavaScript Closures, Lexical Environment, Private State & `debounce()` / `once()` Patterns [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Lexical environment & Private state encapsulation • Core Syntax, CLI & Production APIs: Lexical [[Environment]], closure factory functions, clearTimeout / setTimeout debounce • Deep-Dive Mechanics: Debounce(fn, delay) & Memoization cache • Advanced Patterns: Detached DOM leak & Factory function • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Private Counter & Reusable `once()` Initialization Closure • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for JavaScript Closures

Purpose: To understand and apply JavaScript Closures, Lexical Environment, Private State & `debounce()` / `once()` Patterns in real Web Designing systems — specifically: How inner...

What Topic Is Next → Next Topic 8.3: JavaScript Array Methods — `map`, `filter`, `reduce`, `find`, `some`/`every` & Immutability

8.3 JavaScript Array Methods — `map`, `filter`, `reduce`, `find`, `some`/`every` & Immutability [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Sort((a,b) => a - b) trap & Reduce initial value 0 • Core Syntax, CLI & Production APIs: .filter(), .map(), .reduce((acc, item) => ..., 0), .find(), .some(), .every() • Deep-Dive Mechanics: Immutable map & filter & Find vs filter • Advanced Patterns: Some & every short-circuit & ToSorted() non-mutating • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Numeric Sort Fix & Chained `filter` -> `map` -> `reduce` Pipeline • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for JavaScript Array Methods

Purpose: To understand and apply JavaScript Array Methods — `map`, `filter`, `reduce`, `find`, `some`/`every` & Immutability in real Web Designing systems — specifically: Non-muta...

What Topic Is Next → Module 8 Capstone (Live Debounced Course Search, Multi-Tag Filter & Cart Totalizer Engine) → Next Topic 9.1: JavaScript DOM Manipulation — `querySelector`, `classList`, `DocumentFragment` & XSS Defense

Module 8 Hands-On Capstone Project (100 Marks): Live Debounced Course Search, Multi-Tag Filter & Cart Totalizer Engine — Build an interactive Course Catalog Filter & Mini-Cart engine using a closure-backed `debounce(fn, 250)` search bar, immutable `.filter()`/`.map()` card rendering, and `.reduce()` dynamic su...

MODULE 9 OF 16 • STAGE 3 • 13 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 9: Browser DOM Manipulation, Event Bubbling/Delegation & Secure Form Validation

Manipulate the live DOM safely (`textContent` vs XSS-prone `innerHTML`), scale dynamic UIs with single-listener Event Delegation, and validate forms with accessible ARIA feedback.

Objectives: ✓ JavaScript DOM Manipulation — `querySelector`, `classList`, `DocumentFragment` & XSS Defense — Selecting nodes (`querySelector`, `querySelectorAll` static NodeList), toggling classes (`classList.toggle`), batching inserts via `DocumentFragment`, and why `textContent` prevents XSS. • ✓ DOM Events — `addEventListener`, Capturing vs Bubbling, Event Delegation & `preventDefault()` — 3-phase W3C event flow (Capture -> Target -> Bubble), Event Delegation via `e.target.closest()`, `e.preventDefault()` vs `e.stopPropagation()`, and `{ passive: true }` scroll listeners. • ✓ JavaScript Client-Side Form Validation, `FormData`, `setCustomValidity` & `aria-live` UX — Intercepting `submit` events cleanly, reading `new FormData(form)`, regex & constraint checks, accessible `aria-invalid` + `aria-live="polite"` feedback, and why server validation is still mandatory. • ✓ Build, verify, and ship Module Capstone: Interactive Kanban Study Task Board with Event Delegation & Validated Modal Form.

9.1 JavaScript DOM Manipulation — `querySelector`, `classList`, `DocumentFragment` & XSS Defense [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: TextContent vs innerHTML XSS & DocumentFragment 1 reflow • Core Syntax, CLI & Production APIs: document.querySelector(), el.textContent, el.classList.toggle(), document.createDocumentFrag... • Deep-Dive Mechanics: ClassList.toggle() & Static NodeList vs live HTMLCollection • Advanced Patterns: Dataset attributes & CreateElement • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: XSS-Safe DOM Node Builder Logic Verification • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for JavaScript DOM Manipulation

Purpose: To understand and apply JavaScript DOM Manipulation — `querySelector`, `classList`, `DocumentFragment` & XSS Defense in real Web Designing systems — specifically: Selecti...

What Topic Is Next → Next Topic 9.2: DOM Events — `addEventListener`, Capturing vs Bubbling, Event Delegation & `preventDefault()`

9.2 DOM Events — `addEventListener`, Capturing vs Bubbling, Event Delegation & `preventDefault()` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Capture -> Target -> Bubble & Event Delegation closest() • Core Syntax, CLI & Production APIs: el.addEventListener("click", handler), e.target.closest(selector), e.preventDefault(), { onc... • Deep-Dive Mechanics: PreventDefault vs stopPropagation & E.target vs e.currentTarget • Advanced Patterns: Passive scroll listeners & AbortController cleanup • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Event Delegation Action Dispatcher Pattern • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for DOM Events

Purpose: To understand and apply DOM Events — `addEventListener`, Capturing vs Bubbling, Event Delegation & `preventDefault()` in real Web Designing systems — specifically: 3-phas...

What Topic Is Next → Next Topic 9.3: JavaScript Client-Side Form Validation, `FormData`, `setCustomValidity` & `aria-live` UX

9.3 JavaScript Client-Side Form Validation, `FormData`, `setCustomValidity` & `aria-live` UX [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: FormData & Object.fromEntries & E.preventDefault() on submit • Core Syntax, CLI & Production APIs: form.addEventListener("submit", ...), new FormData(form), Object.fromEntries(fd), input.chec... • Deep-Dive Mechanics: Client UX vs server security & Aria-invalid & aria-live • Advanced Patterns: Reward early punish late & Trim() & regex check • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Clean Lead Form Validator Returning Structured Errors • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for JavaScript Client-Side Form Validation

Purpose: To understand and apply JavaScript Client-Side Form Validation, `FormData`, `setCustomValidity` & `aria-live` UX in real Web Designing systems — specifically: Interceptin...

What Topic Is Next → Module 9 Capstone (Interactive Kanban Study Task Board with Event Delegation & Validated Modal Form) → Next Topic 10.1: JavaScript Promises — Lifecycle States, Microtask Queue, `Promise.all` & `Promise.allSettled`

Module 9 Hands-On Capstone Project (100 Marks): Interactive Kanban Study Task Board with Event Delegation & Validated Modal Form — Build an interactive browser Study Task & Module Tracker using `DocumentFragment` batch rendering, a single delegated `click` listener (`e.target.closest("[data-action]")`) for Complete/Dele...

MODULE 10 OF 16 • STAGE 3 • 13 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 10: Asynchronous JavaScript — Promises, `async`/`await` & REST `fetch()` API Integration

Master Microtask Promise states, `Promise.all` vs `Promise.allSettled`, clean `async`/`await` error boundaries, and resilient `fetch()` REST API UI states (Loading, Empty, Error, AbortController).

Objectives: ✓ JavaScript Promises — Lifecycle States, Microtask Queue, `Promise.all` & `Promise.allSettled` — Eliminating callback hell, Pending / Fulfilled / Rejected states, `.then()/.catch()/.finally()` chaining, fail-fast `Promise.all` vs resilient `Promise.allSettled`, and `Promise.race` timeouts. • ✓ JavaScript `async` & `await` — Sequential Waterfall Trap vs Parallel Concurrency & `try/catch` — How `async` functions always wrap return values in a Promise, pausing function execution non-blockingly at `await`, `try/catch/finally` error handling, and avoiding sequential `await` waterfalls. • ✓ JavaScript `fetch()` REST API — `response.ok` Trap, JSON Headers, CORS & `AbortController` — GET/POST JSON requests with `fetch()`, why HTTP 404/500 does NOT reject `fetch()` automatically, `await response.json()`, `AbortController` request cancellation, and CORS preflight basics. • ✓ Build, verify, and ship Module Capstone: Live Course Explorer & Certificate Verifier SPA with `fetch()`, Skeletons & `AbortController`.

10.1 JavaScript Promises — Lifecycle States, Microtask Queue, `Promise.all` & `Promise.allSettled` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Pending Fulfilled Rejected & Microtask Queue priority • Core Syntax, CLI & Production APIs: new Promise((resolve, reject) => ...), Promise.all(), Promise.allSettled(), .then().catch().... • Deep-Dive Mechanics: Promise.all fail-fast & Promise.allSettled resilience • Advanced Patterns: Returning inner Promise & .finally() cleanup • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Resilient Batch Widget Loader with `Promise.allSettled` • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop

Execution & Memory Lifecycle for JavaScript Promises

Purpose: To understand and apply JavaScript Promises — Lifecycle States, Microtask Queue, `Promise.all` & `Promise.allSettled` in real Web Designing systems — specifically: Elimin...

What Topic Is Next → Next Topic 10.2: JavaScript `async` & `await` — Sequential Waterfall Trap vs Parallel Concurrency & `try/catch`

10.2 JavaScript `async` & `await` — Sequential Waterfall Trap vs Parallel Concurrency & `try/catch` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: ForEach(async) bug & Waterfall vs Promise.all • Core Syntax, CLI & Production APIs: async function load() { try { const [a, b] = await Promise.all([...]); } catch (e) {} } • Deep-Dive Mechanics: Try / catch / finally & Async always returns Promise • Advanced Patterns: For...of sequential await & Non-blocking UI • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Parallel `await Promise.all` Dashboard Hydrator • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for JavaScript `async`

Purpose: To understand and apply JavaScript `async` & `await` — Sequential Waterfall Trap vs Parallel Concurrency & `try/catch` in real Web Designing systems — specifically: How `...

What Topic Is Next → Next Topic 10.3: JavaScript `fetch()` REST API — `response.ok` Trap, JSON Headers, CORS & `AbortController`

10.3 JavaScript `fetch()` REST API — `response.ok` Trap, JSON Headers, CORS & `AbortController` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Response.ok check & 404/500 does not reject • Core Syntax, CLI & Production APIs: fetch(url, { method, headers, body, signal }), if (!res.ok) throw new Error(res.status), Abo... • Deep-Dive Mechanics: AbortController signal & Race-condition prevention • Advanced Patterns: Content-Type application/json & CORS preflight • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Production-Grade `fetchJson()` Wrapper with Status & Timeout Guard • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for JavaScript `fetch()` REST API

Purpose: To understand and apply JavaScript `fetch()` REST API — `response.ok` Trap, JSON Headers, CORS & `AbortController` in real Web Designing systems — specifically: GET/POST...

What Topic Is Next → Module 10 Capstone (Live Course Explorer & Certificate Verifier SPA with `fetch()`, Skeletons & `AbortController`) → Next Topic 11.1: CSS Custom Properties, Fluid clamp() Scales & Dark/Light Themes

Module 10 Hands-On Capstone Project (100 Marks): Live Course Explorer & Certificate Verifier SPA with `fetch()`, Skeletons & `AbortController` — Build a responsive Single-Page Web Application in HTML5 + CSS3 Grid + ES2024 `async`/`await` `fetch()` that loads live JSON data, renders shimmer loading skeletons, cancels superseded search...

MODULE 11 OF 16 • STAGE 3 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Advanced CSS Architecture: Design Tokens, Container Queries & Motion

Production-grade Web Designing Training engineering module covering Advanced CSS Architecture: Design Tokens, Container Queries & Motion with hands-on labs and capstone build.

Objectives: ✓ CSS Custom Properties, Fluid clamp() Scales & Dark/Light Themes — Maintain pixel-consistent spacing, typography, and dark mode across 50+ storefront pages. • ✓ @container Queries, CSS Subgrid & Modern :has() Selectors — Build modular product and booking cards that adapt to any sidebar or grid slot cleanly. • ✓ GPU-Accelerated 60fps Transitions & prefers-reduced-motion — Create polished micro-interactions without layout thrashing or motion discomfort. • ✓ Build, verify, and ship Module Capstone: Multi-Brand Design Token System & Responsive Component Kit.

11.1 CSS Custom Properties, Fluid clamp() Scales & Dark/Light Themes [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Fluid clamp() Scales • Integration with @container Queries, CSS Subgrid & Modern • Live Debugging, Telemetry & Technical Interview Trade • What Is CSS Custom Properties, Fluid clamp() Scales & Dark/Light Themes & Why Industry Teams Use It • In Web Designing Training, CSS Custom Properties, Fluid clamp() Scales & Dark/Light Themes exists for one clear rea... • Even if you come from a non • Runnable Code Implementation (JAVASCRIPT)

Purpose: To understand and apply CSS Custom Properties, Fluid clamp() Scales & Dark/Light Themes in real Web Designing systems — specifically: Maintain pixel-consistent spacing, t...

What Topic Is Next → Next Topic 11.2: @container Queries, CSS Subgrid & Modern :has() Selectors

11.2 @container Queries, CSS Subgrid & Modern :has() Selectors [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of CSS Subgrid • Integration with GPU • Live Debugging, Telemetry & Technical Interview Trade • What Is @container Queries, CSS Subgrid & Modern • In Web Designing Training, @container Queries, CSS Subgrid & Modern • Even if you come from a non • Runnable Code Implementation (JAVASCRIPT)

Purpose: To understand and apply @container Queries, CSS Subgrid & Modern :has() Selectors in real Web Designing systems — specifically: Build modular product and booking cards th...

What Topic Is Next → Next Topic 11.3: GPU-Accelerated 60fps Transitions & prefers-reduced-motion

11.3 GPU-Accelerated 60fps Transitions & prefers-reduced-motion [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of prefers • Integration with CSS Custom Properties, Fluid clamp() Scales & Dark/Light Themes • Live Debugging, Telemetry & Technical Interview Trade • What Is GPU • In Web Designing Training, GPU • Even if you come from a non • Runnable Code Implementation (JAVASCRIPT)

Purpose: To understand and apply GPU-Accelerated 60fps Transitions & prefers-reduced-motion in real Web Designing systems — specifically: Create polished micro-interactions withou...

What Topic Is Next → Module 11 Capstone (Multi-Brand Design Token System & Responsive Component Kit) → Next Topic 12.1: Async/Await Fetch Pipelines, HTTP Error Handling & Retry Backoff

Module 11 Hands-On Capstone Project (100 Marks): Multi-Brand Design Token System & Responsive Component Kit — Design, implement, test, and document 'Multi-Brand Design Token System & Responsive Component Kit' applying every production concept from Module 11 (Advanced CSS Architecture: Design Tokens,...

MODULE 12 OF 16 • STAGE 3 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Async ES2024 JavaScript: Fetch API, AbortController & Web Storage

Production-grade Web Designing Training engineering module covering Async ES2024 JavaScript: Fetch API, AbortController & Web Storage with hands-on labs and capstone build.

Objectives: ✓ Async/Await Fetch Pipelines, HTTP Error Handling & Retry Backoff — Connect HTML/CSS interfaces to live REST APIs with skeleton loaders and clean error states. • ✓ Debounced Search Autocomplete & AbortController Race Cancellation — Prevent stale slow network responses from overwriting newer search results. • ✓ LocalStorage, SessionStorage, History API & FormData Validation — Persist shopping carts across refreshes and validate multi-step forms in real time. • ✓ Build, verify, and ship Module Capstone: Live Search, Cart Sync & Offline-Resilient Checkout UI.

12.1 Async/Await Fetch Pipelines, HTTP Error Handling & Retry Backoff [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of HTTP Error Handling • Integration with Debounced Search Autocomplete & AbortController Race Cancellation • Live Debugging, Telemetry & Technical Interview Trade • What Is Async/Await Fetch Pipelines, HTTP Error Handling & Retry Backoff & Why Industry Teams Use It • In Web Designing Training, Async/Await Fetch Pipelines, HTTP Error Handling & Retry Backoff exists for one clear re... • Even if you come from a non • Runnable Code Implementation (JAVASCRIPT)

Purpose: To understand and apply Async/Await Fetch Pipelines, HTTP Error Handling & Retry Backoff in real Web Designing systems — specifically: Connect HTML/CSS interfaces to live...

What Topic Is Next → Next Topic 12.2: Debounced Search Autocomplete & AbortController Race Cancellation

12.2 Debounced Search Autocomplete & AbortController Race Cancellation [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of AbortController Race Cancellation • Integration with LocalStorage, SessionStorage, History API & FormData Validation • Live Debugging, Telemetry & Technical Interview Trade • What Is Debounced Search Autocomplete & AbortController Race Cancellation & Why Industry Teams Use It • In Web Designing Training, Debounced Search Autocomplete & AbortController Race Cancellation exists for one clear r... • Even if you come from a non • Runnable Code Implementation (JAVASCRIPT)

Purpose: To understand and apply Debounced Search Autocomplete & AbortController Race Cancellation in real Web Designing systems — specifically: Prevent stale slow network respons...

What Topic Is Next → Next Topic 12.3: LocalStorage, SessionStorage, History API & FormData Validation

12.3 LocalStorage, SessionStorage, History API & FormData Validation [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of SessionStorage • Integration with Async/Await Fetch Pipelines, HTTP Error Handling & Retry Backoff • Live Debugging, Telemetry & Technical Interview Trade • What Is LocalStorage, SessionStorage, History API & FormData Validation & Why Industry Teams Use It • In Web Designing Training, LocalStorage, SessionStorage, History API & FormData Validation exists for one clear rea... • Even if you come from a non • Runnable Code Implementation (JAVASCRIPT)

Purpose: To understand and apply LocalStorage, SessionStorage, History API & FormData Validation in real Web Designing systems — specifically: Persist shopping carts across refres...

What Topic Is Next → Module 12 Capstone (Live Search, Cart Sync & Offline-Resilient Checkout UI) → Next Topic 13.1: Keyboard Focus Management, ARIA Live Regions & Contrast Ratios

Module 12 Hands-On Capstone Project (100 Marks): Live Search, Cart Sync & Offline-Resilient Checkout UI — Design, implement, test, and document 'Live Search, Cart Sync & Offline-Resilient Checkout UI' applying every production concept from Module 12 (Async ES2024 JavaScript: Fetch API, AbortCont...

MODULE 13 OF 16 • STAGE 4 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

WCAG 2.2 Web Accessibility, Keyboard Navigation & Technical SEO

Production-grade Web Designing Training engineering module covering WCAG 2.2 Web Accessibility, Keyboard Navigation & Technical SEO with hands-on labs and capstone build.

Objectives: ✓ Keyboard Focus Management, ARIA Live Regions & Contrast Ratios — Ensure screen-reader and keyboard-only customers can browse and purchase effortlessly. • ✓ Semantic Meta Tags, OpenGraph Cards, JSON-LD Schema & Sitemaps — Win rich Google search snippets and high-converting WhatsApp/LinkedIn link previews. • ✓ Cross-Browser DevTools Auditing (Mobile Throttling & Layout Inspector) — Eliminate overflow scrollbars and touch-target bugs across iOS Safari and Android Chrome. • ✓ Build, verify, and ship Module Capstone: 98+ Lighthouse Accessible & SEO-Optimized Business Portal.

13.1 Keyboard Focus Management, ARIA Live Regions & Contrast Ratios [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of ARIA Live Regions • Integration with Semantic Meta Tags, OpenGraph Cards, JSON • Live Debugging, Telemetry & Technical Interview Trade • What Is Keyboard Focus Management, ARIA Live Regions & Contrast Ratios & Why Industry Teams Use It • In Web Designing Training, Keyboard Focus Management, ARIA Live Regions & Contrast Ratios exists for one clear real • Even if you come from a non • Runnable Code Implementation (JAVASCRIPT)

Purpose: To understand and apply Keyboard Focus Management, ARIA Live Regions & Contrast Ratios in real Web Designing systems — specifically: Ensure screen-reader and keyboard-onl...

What Topic Is Next → Next Topic 13.2: Semantic Meta Tags, OpenGraph Cards, JSON-LD Schema & Sitemaps

13.2 Semantic Meta Tags, OpenGraph Cards, JSON-LD Schema & Sitemaps [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of OpenGraph Cards • Integration with Cross • Live Debugging, Telemetry & Technical Interview Trade • What Is Semantic Meta Tags, OpenGraph Cards, JSON • In Web Designing Training, Semantic Meta Tags, OpenGraph Cards, JSON • Even if you come from a non • Runnable Code Implementation (JAVASCRIPT)

Purpose: To understand and apply Semantic Meta Tags, OpenGraph Cards, JSON-LD Schema & Sitemaps in real Web Designing systems — specifically: Win rich Google search snippets and h...

What Topic Is Next → Next Topic 13.3: Cross-Browser DevTools Auditing (Mobile Throttling & Layout Inspector)

13.3 Cross-Browser DevTools Auditing (Mobile Throttling & Layout Inspector) [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Layout Inspector) • Integration with Keyboard Focus Management, ARIA Live Regions & Contrast Ratios • Live Debugging, Telemetry & Technical Interview Trade • What Is Cross • In Web Designing Training, Cross • Even if you come from a non • Runnable Code Implementation (JAVASCRIPT)

Purpose: To understand and apply Cross-Browser DevTools Auditing (Mobile Throttling & Layout Inspector) in real Web Designing systems — specifically: Eliminate overflow scrollbars...

What Topic Is Next → Module 13 Capstone (98+ Lighthouse Accessible & SEO-Optimized Business Portal) → Next Topic 14.1: Responsive <picture>, AVIF/WebP Compression & Lazy Loading

Module 13 Hands-On Capstone Project (100 Marks): 98+ Lighthouse Accessible & SEO-Optimized Business Portal — Design, implement, test, and document '98+ Lighthouse Accessible & SEO-Optimized Business Portal' applying every production concept from Module 13 (WCAG 2.2 Web Accessibility, Keyboard Navig...

MODULE 14 OF 16 • STAGE 4 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Core Web Vitals Optimization, Vite Asset Bundling & Live CDN Deploy

Production-grade Web Designing Training engineering module covering Core Web Vitals Optimization, Vite Asset Bundling & Live CDN Deploy with hands-on labs and capstone build.

Objectives: ✓ Responsive <picture>, AVIF/WebP Compression & Lazy Loading — Cut page weight by 75% so mobile 4G visitors see hero content in under 1.5 seconds. • ✓ Font Loading (font-display: swap), Critical CSS & CLS Zeroing — Stop text flashes and layout shifts that hurt conversions and Core Web Vitals scores. • ✓ Vite Bundling, Asset Fingerprinting, Security Headers & HTTPS Launch — Deploy cache-busted, minified production web builds to live custom domains. • ✓ Build, verify, and ship Module Capstone: Sub-1.5s High-Conversion Storefront Deployed on HTTPS CDN.

14.1 Responsive <picture>, AVIF/WebP Compression & Lazy Loading [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of AVIF/WebP Compression • Integration with Font Loading (font • Live Debugging, Telemetry & Technical Interview Trade • What Is Responsive , AVIF/WebP Compression & Lazy Loading & Why Industry Teams Use It • In Web Designing Training, Responsive , AVIF/WebP Compression & Lazy Loading exists for one clear real • Even if you come from a non • Runnable Code Implementation (JAVASCRIPT)

Purpose: To understand and apply Responsive <picture>, AVIF/WebP Compression & Lazy Loading in real Web Designing systems — specifically: Cut page weight by 75% so mobile 4G visit...

What Topic Is Next → Next Topic 14.2: Font Loading (font-display: swap), Critical CSS & CLS Zeroing

14.2 Font Loading (font-display: swap), Critical CSS & CLS Zeroing [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Critical CSS • Integration with Vite Bundling, Asset Fingerprinting, Security Headers & HTTPS Launch • Live Debugging, Telemetry & Technical Interview Trade • What Is Font Loading (font • In Web Designing Training, Font Loading (font • Even if you come from a non • Runnable Code Implementation (JAVASCRIPT)

Purpose: To understand and apply Font Loading (font-display: swap), Critical CSS & CLS Zeroing in real Web Designing systems — specifically: Stop text flashes and layout shifts th...

What Topic Is Next → Next Topic 14.3: Vite Bundling, Asset Fingerprinting, Security Headers & HTTPS Launch

14.3 Vite Bundling, Asset Fingerprinting, Security Headers & HTTPS Launch [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Asset Fingerprinting • Integration with Responsive <picture>, AVIF/WebP Compression & Lazy Loading • Live Debugging, Telemetry & Technical Interview Trade • Integration with Responsive , AVIF/WebP Compression & Lazy Loading • What Is Vite Bundling, Asset Fingerprinting, Security Headers & HTTPS Launch & Why Industry Teams Use It • In Web Designing Training, Vite Bundling, Asset Fingerprinting, Security Headers & HTTPS Launch exists for one clea... • Even if you come from a non

Purpose: To understand and apply Vite Bundling, Asset Fingerprinting, Security Headers & HTTPS Launch in real Web Designing systems — specifically: Deploy cache-busted, minified p...

What Topic Is Next → Module 14 Capstone (Sub-1.5s High-Conversion Storefront Deployed on HTTPS CDN) → Next Topic 15.1: CSS Custom Properties (Design Tokens), Fluid clamp() Typography & Container Queries

Module 14 Hands-On Capstone Project (100 Marks): Sub-1.5s High-Conversion Storefront Deployed on HTTPS CDN — Design, implement, test, and document 'Sub-1.5s High-Conversion Storefront Deployed on HTTPS CDN' applying every production concept from Module 14 (Core Web Vitals Optimization, Vite Asset B...

MODULE 15 OF 16 • STAGE 4 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Design Tokens, WCAG 2.2 Accessibility & Modern CSS Container Queries

Self-explanatory list-and-bullet production mastery module covering Design Tokens, WCAG 2.2 Accessibility & Modern CSS Container Queries in Web Designing Training, paired with hands-on labs, capstone repository, Topic Mock Tests, and Spoken MockAttempt AI Viva.

Objectives: ✓ CSS Custom Properties (Design Tokens), Fluid clamp() Typography & Container Queries — Purpose: Build modular UI components that adapt cleanly to any card width, viewport, and brand theme. • ✓ WCAG 2.2 AA Accessibility: Semantic Landmarks, Focus Traps & ARIA Live Regions — Purpose: Ensure keyboard users and screen readers can navigate forms, modals, and menus effortlessly. • ✓ Progressive Enhancement, Form Validation UX & Client Storage Security — Purpose: Deliver resilient forms with instant accessible error feedback and XSS-safe DOM updates. • ✓ Build, verify, and document 'Accessible Component System with CSS Variables, Container Queries & Keyboard Nav' with clean Git commits and edge-case protection. • ✓ Clear 100% of Topic Timed Mock Tests and Spoken Voice + MockAttempt AI Technical Interviews.

15.1 CSS Custom Properties (Design Tokens), Fluid clamp() Typography & Container Queries [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for CSS Custom Properties (Design Tokens), Fluid clamp() Typography & Containe... • Performance Optimization, Complexity & Latency Impact of CSS Custom Properties (Design Tokens), Fluid clamp() Typog... • Live Debugging, Stack • Runnable Code Implementation (JAVASCRIPT) • Hands • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for CSS Custom Properties (Design Tokens) • Production Hardening & Security: XSS-safe textContent updates, CSP headers & accessible form validation

Purpose: To understand and apply CSS Custom Properties (Design Tokens), Fluid clamp() Typography & Container Queries in real Web Designing systems — specifically: Build modular UI...

What Topic Is Next → Next Topic 15.2: WCAG 2.2 AA Accessibility: Semantic Landmarks, Focus Traps & ARIA Live Regions

15.2 WCAG 2.2 AA Accessibility: Semantic Landmarks, Focus Traps & ARIA Live Regions [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for WCAG 2.2 AA Accessibility • Performance Optimization, Complexity & Latency Impact of WCAG 2.2 AA Accessibility • Live Debugging, Stack • Runnable Code Implementation (JAVASCRIPT) • Hands • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for WCAG 2.2 AA Accessibility • Production Hardening & Security: XSS-safe textContent updates, CSP headers & accessible form validation

Purpose: To understand and apply WCAG 2.2 AA Accessibility: Semantic Landmarks, Focus Traps & ARIA Live Regions in real Web Designing systems — specifically: Ensure keyboard users...

What Topic Is Next → Next Topic 15.3: Progressive Enhancement, Form Validation UX & Client Storage Security

15.3 Progressive Enhancement, Form Validation UX & Client Storage Security [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Progressive Enhancement, Form Validation UX & Client Storage Security • Performance Optimization, Complexity & Latency Impact of Progressive Enhancement, Form Validation UX & Client Stora... • Live Debugging, Stack • Runnable Code Implementation (JAVASCRIPT) • Hands • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for Progressive Enhancement • Production Hardening & Security: XSS-safe textContent updates, CSP headers & accessible form validation

Purpose: To understand and apply Progressive Enhancement, Form Validation UX & Client Storage Security in real Web Designing systems — specifically: Deliver resilient forms with i...

What Topic Is Next → Module 15 Capstone (Accessible Component System with CSS Variables, Container Queries & Keyboard Nav) → Next Topic 16.1: Optimizing Largest Contentful Paint (LCP), Interaction to Next Paint (INP) & Zero CLS

Module 15 Hands-On Capstone Project (100 Marks): Accessible Component System with CSS Variables, Container Queries & Keyboard Nav — Architect, implement, test, and document 'Accessible Component System with CSS Variables, Container Queries & Keyboard Nav' applying every bullet-wise engineering concept from Module 15 (Des...

MODULE 16 OF 16 • STAGE 4 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Core Web Vitals (LCP, INP, CLS), Asset Optimization & Frontend Deployment

Self-explanatory list-and-bullet production mastery module covering Core Web Vitals (LCP, INP, CLS), Asset Optimization & Frontend Deployment in Web Designing Training, paired with hands-on labs, capstone repository, Topic Mock Tests, and Spoken MockAttempt AI Viva.

Objectives: ✓ Optimizing Largest Contentful Paint (LCP), Interaction to Next Paint (INP) & Zero CLS — Purpose: Eliminate layout jumps, sluggish button clicks, and render-blocking CSS/JS on mobile networks. • ✓ Responsive AVIF/WebP Images, Lazy Loading, Font Subsetting & Caching Headers — Purpose: Cut page payload weight dramatically while keeping visuals razor-sharp on retina screens. • ✓ Technical SEO Metadata, Cross-Browser QA & Live HTTPS Storefront Launch — Purpose: Ship search-engine-ready, cross-device verified websites with OpenGraph and JSON-LD schemas. • ✓ Build, verify, and document '98+ Lighthouse Production Storefront Audit & Zero-CLS Asset Pipeline' with clean Git commits and edge-case protection. • ✓ Clear 100% of Topic Timed Mock Tests and Spoken Voice + MockAttempt AI Technical Interviews.

16.1 Optimizing Largest Contentful Paint (LCP), Interaction to Next Paint (INP) & Zero CLS [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Optimizing Largest Contentful Paint (LCP), Interaction to Next Paint (INP)... • Performance Optimization, Complexity & Latency Impact of Optimizing Largest Contentful Paint (LCP), Interaction to.... • Live Debugging, Stack • Runnable Code Implementation (JAVASCRIPT) • Hands • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for Optimizing Largest Contentful Paint (LCP) • Production Hardening & Security: XSS-safe textContent updates, CSP headers & accessible form validation

Purpose: To understand and apply Optimizing Largest Contentful Paint (LCP), Interaction to Next Paint (INP) & Zero CLS in real Web Designing systems — specifically: Eliminate layo...

What Topic Is Next → Next Topic 16.2: Responsive AVIF/WebP Images, Lazy Loading, Font Subsetting & Caching Headers

16.2 Responsive AVIF/WebP Images, Lazy Loading, Font Subsetting & Caching Headers [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Responsive AVIF/WebP Images, Lazy Loading, Font Subsetting & Caching Heade... • Performance Optimization, Complexity & Latency Impact of Responsive AVIF/WebP Images, Lazy Loading, Font Subsetting... • Live Debugging, Stack • Runnable Code Implementation (JAVASCRIPT) • Hands • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for Responsive AVIF/WebP Images • Production Hardening & Security: XSS-safe textContent updates, CSP headers & accessible form validation

Purpose: To understand and apply Responsive AVIF/WebP Images, Lazy Loading, Font Subsetting & Caching Headers in real Web Designing systems — specifically: Cut page payload weight...

What Topic Is Next → Next Topic 16.3: Technical SEO Metadata, Cross-Browser QA & Live HTTPS Storefront Launch

16.3 Technical SEO Metadata, Cross-Browser QA & Live HTTPS Storefront Launch [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Technical SEO Metadata, Cross • Performance Optimization, Complexity & Latency Impact of Technical SEO Metadata, Cross • Live Debugging, Stack • Runnable Code Implementation (JAVASCRIPT) • Hands • Idiomatic WEBD Implementation: semantic HTML5 landmarks, CSS Grid/Flexbox tokens & modular ES6+ DOM code • Browser Critical Rendering Path, CSSOM & ES2024 Event Loop Execution & Memory Lifecycle for Technical SEO Metadata • Production Hardening & Security: XSS-safe textContent updates, CSP headers & accessible form validation

Purpose: To understand and apply Technical SEO Metadata, Cross-Browser QA & Live HTTPS Storefront Launch in real Web Designing systems — specifically: Ship search-engine-ready, cr...

What Topic Is Next → Module 16 Capstone → End-of-Course Live Client Project: OmniStore High-Conversion Responsive E-Commerce & Interactive Booking Storefront

Module 16 Hands-On Capstone Project (100 Marks): 98+ Lighthouse Production Storefront Audit & Zero-CLS Asset Pipeline — Architect, implement, test, and document '98+ Lighthouse Production Storefront Audit & Zero-CLS Asset Pipeline' applying every bullet-wise engineering concept from Module 16 (Core Web Vitals...

3. LIVE CLIENT PRODUCTION PROJECT & CAREER SCOPE

End-of-Course Live Client Production Project + Official Internship Confirmation Letter

Project: OmniStore High-Conversion Responsive E-Commerce & Interactive Booking Storefront • **Confirmed Role:** Frontend Web Designing & JavaScript Engineering Intern

Domain & Production Infra: D2C Retail & Hospitality Client Production Web Launch (Semantic HTML5 + Modern CSS3 Grid/Flexbox + ES2024 JS + CDN HTTPS Edge)

- Crafted pixel-accurate, WCAG-accessible responsive layouts using semantic HTML5, CSS Grid, Flexbox, and custom variables.
- Programmed interactive cart state, dynamic product filtering, form validation, and REST API integration in vanilla ES6+ JavaScript.
- Achieved 95+ Google Lighthouse Performance, Accessibility, and Technical SEO scores on mobile and desktop.
- Shipped asset-minified storefront builds to live HTTPS production domain for real client user traffic.

Target Job Roles: Frontend Web Developer, UI / Responsive Web Designer, HTML5/CSS3/JavaScript Developer, Junior Web Application Engineer

Realistic Annual CTC Scope: **Fresher / Entry-Level (0–1 Yr):** ₹3.8 LPA – ₹7.2 LPA **Mid-Level UI / Frontend Engineer (1–3 Yrs):** ₹7.5 LPA – ₹14.5 LPA **Senior Frontend / Full-Stack Lead (3–6+ Yrs):** ₹15.0 LPA – ₹30.0+ LPA

MockAttempt Academy Pvt. Ltd. • Email: support@mockattempt.com • Phone/WhatsApp: +91 95400 07917 • Web: https://mockattempt.com/courses/web-designing-training

Try 2 Free Topic-Wise Timed Topic Mock Tests & Spoken MockAttempt AI Interviews daily on every topic above before enrolling.