WDE™ – Certified Entry-Level Web Developer EXAM SYLLABUS

(Exam WDE-40-01)

Last revised: (September 18, 2025)

Module 1: HTML Fundamentals (6) (15%)

1.1 Document Skeleton & Doctype (1)

Objective 1.1.1 Describe the basic structure of an HTML document, including the *<!DOCTYPE>* declaration

A. Place `<!DOCTYPE html>` at the top to ensure standards mode and predictable rendering.
B. Wrap all content in a single `<html lang="...">` element with appropriate language.
C. Separate metadata in `<head>` (e.g., `<meta charset="utf-8">`, `title`) from content in `<body>`.

1.2 Block vs. Inline Elements (1)

Objective 1.2.1 Understand the distinction between block-level and inline elements

A. Differentiate default layout behavior: block elements create new lines; inline elements flow within text.
B. Recognize common examples (block: `<div>`, `<p>`; inline: `<span>`, `<a>`) and choose based on meaning.

1.3 Text Entities & Character Encoding (2)

Objective 1.3.1 Identify and correctly use basic HTML entities

A. Escape reserved characters with entities such as `&`, `<`, `>`, `&quot;`, `&apos;`.
B. Use non-breaking space ` ` only when needed; prefer CSS for spacing.
C. Ensure entities render correctly across encodings and user agents.

Objective 1.3.2 Explain the importance of character encoding

A. Specify UTF-8 early in `<head>` with `<meta charset="utf-8">` to avoid mojibake.
B. Keep editor, server, and document encodings consistent to prevent corruption.
C. Test pages with multilingual content and special symbols.

1.4 Comments in HTML (1)

Objective 1.4.1 Utilize comments to document HTML code

A. Use `<!-- comment -->` to explain structure or intent without affecting layout.
B. Avoid exposing secrets, credentials, or debug notes in shipped HTML.
C. Keep comments concise and relevant to future maintenance.

1.5 Core Tags: html, head, title, body (1)

Objective 1.5.1 Demonstrate the use of basic tags (*html*, *head*, *title*, *body*)

A. Provide a unique, descriptive `<title>` per page for usability and SEO.
B. Place metadata, icons, and styles in `<head>`; present page content within `<body>`.
C. Validate the basic skeleton to avoid parsing and rendering issues.

Module 2: Text Formatting and Structure (8) (20%)

2.1 Emphasis & Inline Semantics (1)

Objective 2.1.1 Implement text formatting tags such as *strong*, *em*, *u*, *del*, *sup*, *sub*, and *mark*

A. Use semantic emphasis (`<strong>`, `<em>`) to convey meaning, not just style.
B. Apply `<del>`, `<u>`, `<mark>`, `<sup>`, `<sub>` where appropriate and avoid overuse.
C. Style the visual appearance with CSS rather than misusing tags.

2.2 Headings, Paragraphs & Separation (3)

Objective 2.2.1 Demonstrate the use of headings and their importance in structuring content and enhancing accessibility

A. Create a logical outline with a single page-level `<h1>` and nested headings.
B. Use headings to label sections for screen readers and in-page navigation.
C. Avoid skipping levels purely for visual size; use CSS for sizing.

Objective 2.2.2 Structure content with paragraphs and manage text flow

A. Wrap each idea in a distinct `<p>` and avoid empty paragraphs for spacing.
B. Control flow and spacing with CSS rather than multiple `<br>` tags.
C. Use inline semantics for quotes, code, and emphasis within paragraphs.

Objective 2.2.3 Apply line breaks and horizontal rules for content separation

A. Insert `<br>` for meaningful single line breaks (e.g., addresses, poetry), not layout.
B. Use `<hr>` to indicate thematic shifts between sections.
C. Prefer CSS margins and borders for visual spacing.

2.3 Quotations, Citations, Abbreviations & Code Semantics (2)

Objective 2.3.1 Use the *blockquote*, *q*, *cite*, and *abbr* tags appropriately

A. Mark long quotations with `<blockquote>` and inline quotes with `<q>`.
B. Provide sources with `<cite>` and expansions with `<abbr title="..." >`.
C. Retain quotation punctuation as content, not styling.

Objective 2.3.2 Employ *code*, *pre*, *kbd*, and *samp* tags to display preformatted text and format code snippets and user input

A. Wrap code fragments with `<code>` and preserve whitespace with `<pre>` when needed.
B. Indicate user input with `<kbd>` and program output with `<samp>`.

2.4 Lists & Tables (2)

Objective 2.4.1 Create and manipulate ordered, unordered, and definition lists, including nesting

A. Choose list types that reflect meaning and keep nesting clear and minimal.
B. Use `<dl>` with paired `<dt>`/`<dd>` for term-definition content.

Objective 2.4.2 Create and manipulate tables with headers, captions, and merged cells

A. Use `<table>` for data, add `<caption>`, and group rows with `<thead>`, `<tbody>`, `<tfoot>`.
B. Provide header scope via `<th scope="col|row">` and merge cells with `colspan`/`rowspan` sparingly.

Module 3: Multimedia and Hyperlinks (8) (20%)

3.1 Images (1)

Objective 3.1.1 Embed images with *img*, focusing on *alt* text and basic responsive attributes

A. Provide informative `alt` text or empty `alt` for decorative images.

B. Use `srcset`/`sizes` and intrinsic `width`/`height` to improve responsiveness and stability.
C. Consider lazy loading with `loading="lazy"` for non-critical images.

3.2 Hyperlinks (1)

Objective 3.2.1 Create hyperlinks with the *a* tag, including internal, external, email, and telephone links

A. Use clear, descriptive link text and valid `href` values (e.g., `mailto:`, `tel:`).
B. Open new windows sparingly and add `rel="noopener noreferrer"` with `target="_blank"`.
C. Provide anchor links for in-page navigation and skip links for accessibility.

3.3 Audio & Video Embeds (1)

Objective 3.3.1 Embed multimedia with audio and video, including sources and responsiveness

A. Include controls and provide multiple `<source>` formats for compatibility.
B. Add captions/subtitles via `<track kind="captions">` and a poster for video.
C. Size media responsively and avoid autoplay unless essential and muted.

3.4 Image Maps (1)

Objective 3.4.1 Design interactive image maps with *map* and *area*

A. Define accurate coordinates and meaningful `alt`/`aria-label` for clickable regions.
B. Ensure keyboard accessibility and provide text alternatives for regions.
C. Test image maps across devices and resolutions.

3.5 Figure & Figcaption (1)

Objective 3.5.1 Use figure and figcaption to associate media with captions

A. Wrap related visuals in `<figure>` with a concise `<figcaption>`.
B. Keep captions close to media for context and accessibility.

3.6 Iframes & External Embeds (1)

Objective 3.6.1 Embed external web content with iframe, including responsive handling

A. Provide a descriptive title and consider security attributes like `sandbox` and `referrerpolicy`.
B. Maintain aspect ratio responsively via CSS or the aspect-ratio property.
C. Defer non-critical iframes and avoid performance bottlenecks.

3.7 Linked Media (1)

Objective 3.7.1 Integrate multimedia elements as clickable links

A. Wrap images or icons in `<a>` with clear alternative text describing the destination.
B. Provide focus styles and large tap targets for linked media.
C. Ensure linked media is distinguishable from decorative content.

3.8 Favicons (1)

Objective 3.8.1 Integrate favicons for brand identity and user recognition

A. Link appropriate icon formats and sizes in `<head>` (e.g., *ico*, *png*, *svg*).
B. Provide a web app manifest where applicable.
C. Verify appearance in tabs, pinned tiles, and bookmarks.

Module 4: Forms and Styling (10) (25%)

4.1 Forms: Inputs, Submission, Grouping, Validation & Selects (5)

Objective 4.1.1 Design forms with a variety of input types and understand their use cases

A. Choose appropriate controls (`input`, `textarea`, `button`) and types (`text`, `email`, `url`, `number`, `password`, `checkbox`, `radio`, `submit`, `reset`).
B. Assign meaningful `name`/`id` values and associate labels for accessibility.
C. Provide helpful placeholders and helper text without replacing proper labels.

Objective 4.1.2 Discuss form submission methods (GET vs. POST) and their appropriate use cases

A. Use `GET` for idempotent queries and bookmarkable results
B. Use `POST` for data creation or changes.
C. Set action, choose `enctype` when needed (e.g., file uploads), and consider autocomplete.
D. Handle server responses and errors gracefully.

Objective 4.1.3 Implement field grouping with fieldset and legend; use *form* attributes to customize behavior

A. Group related inputs within `<fieldset>` and label groups with a clear `<legend>`.
B. Apply attributes such as `value`, `placeholder`, `disabled`, and `readonly` appropriately.

Objective 4.1.4 Use validation techniques and attributes

A. Leverage built-in constraints (`required`, `min`, `max`, `maxlength`, `pattern`, `step`) before scripting.
B. Provide clear inline messages and preserve user input on errors.
C. Respect user privacy and avoid exposing sensitive data in URLs.

Objective 4.1.5 Create dropdown menus with select and option

A. Group related options with `<optgroup>` and provide a meaningful default.
B. Use `multiple` and `size` attributes when appropriate and ensure keyboard operability.
C. Pair selects with labels and helper text for clarity.

4.2 CSS Application: Inline/Internal Styles, Classes & IDs (2)

Objective 4.2.1 Apply CSS for styling HTML elements, including inline and internal styles

A. Embed small page-specific rules in `<style>` and reserve inline styles for one-off overrides.
B. Prefer external stylesheets for reuse, caching, and maintainability.
C. Keep specificity low and avoid overusing `!important`.

Objective 4.2.2 Explore the use of CSS classes and IDs for element styling

A. Use classes for reusable patterns and IDs for unique elements that require anchors or scripting.
B. Avoid styling by ID alone; prefer class-based selectors for flexibility.
C. Adopt consistent naming conventions.

4.3 Styling & Layout Fundamentals: Color, Typography, Box Model & Generic Containers (3)

Objective 4.3.1 Incorporate *color* and *font* properties for consistent website styling

A. Set readable typography (family, size, weight, line height) and establish a type scale.
B. Apply color and background color with sufficient contrast for accessibility.
C. Use CSS custom properties to centralize tokens for color and typography.

Objective 4.3.2 Differentiate and apply the use of *span* and *div* for styling and layout

A. Use `<span>` for inline semantics and `<div>` for block-level grouping when no semantic element fits.
B. Prefer semantic HTML5 elements over generic containers when meaning exists.
C. Keep markup lean and avoid unnecessary wrappers.

Objective 4.3.3 Incorporate basic styling attributes to manipulate container appearance (*border*, *padding*, *margin*)

A. Control spacing with the box model and consider `box-sizing: border-box;` for predictable layouts.
B. Use logical properties (e.g., `margin-inline`, `padding-block`) for internationalization.
C. Apply consistent spacing scales to improve rhythm and maintainability.

Module 4: Accessibility, Best Practices, and Modern HTML (8) (20%)

5.1 Accessibility Foundations & ARIA (2)

Objective 5.1.1 Explain the core principles of web accessibility and the importance of following WCAG guidelines

A. Apply POUR principles (*Perceivable*, *Operable*, *Understandable*, *Robust*) to content and UI.
B. Provide alternatives, keyboard access, visible focus, and sufficient contrast.
C. Document accessibility acceptance criteria during development.

Objective 5.1.2 Utilize ARIA roles, states, and properties to make web content more accessible

A. Use ARIA to enhance semantics only when native elements cannot express behavior.
B. Apply roles and states (e.g., `role="dialog"`, `aria-expanded`, `aria-hidden`) correctly and update them dynamically.

5.2 Semantic Structure (1)

Objective 5.2.1 Implement semantic markup and integrate modern HTML5 structural elements

A. Organize documents with `<header>`, `<nav>`, `<main>`, `<article>`, `<section>`, `<aside>`, `<footer>`.
B. Provide skip links and landmark regions for efficient navigation.
C. Pair structure with headings for a clear document outline.

5.3 Structured Data (1)

Objective 5.3.1 Apply microformats, microdata, and Schema.org for enriched content semantics

A. Annotate content with microdata.
B. Use consistent properties (e.g., `itemprop`, `itemscope`) and validate structured data.
C. Consider microformats when required by legacy tools or contexts.

5.4 Platform Features: HTML APIs & SVG (2)

Objective 5.4.1 Understand and apply the basics of HTML APIs (Geolocation, Web Storage)

A. Request location via the *Geolocation API* with explicit user consent and graceful fallbacks.
B. Store small key-value data with `localStorage`/`sessionStorage` and handle quota and privacy.
C. Avoid blocking the UI and handle permission errors robustly.

Objective 5.4.2 Utilize SVG for scalable vector graphics

A. Embed SVG inline for styling and accessibility, and set appropriate roles and titles when needed.
B. Style with CSS and manage icon systems via `<symbol>` and `<use>`.

5.5 Best Practices & Performance (1)

Objective 5.5.1 Discuss HTML5 development best practices focusing on code readability and performance

A. Keep markup semantic, minimal, and consistently formatted.
B. Defer non-critical resources, minimize blocking assets, and audit performance regularly.
C. Use version control and code reviews to sustain quality.

5.6 Accessibility Testing (1)

Objective 5.6.1 Understand and apply the basics of testing web accessibility

A. Explain how to run automated checks and fix common issues.
B. Perform manual keyboard and screen reader spot checks on key flows.
C. Document findings and regressions as part of the release process.

MQC Profile

A Minimally Qualified Candidate (MQC) for the **WDE-40-01** exam is an individual with foundational skills in semantic HTML and introductory CSS. The candidate can assemble valid pages, structure and format content clearly, embed links and media responsibly, build basic forms, and apply simple styling to achieve readable, consistent layouts.

The MQC understands the document skeleton and metadata, block vs. inline elements, entities and character encoding, common text and code semantics, lists and tables, hyperlinks and multimedia, form inputs and submission, basic validation attributes, and core CSS concepts (inline/internal styles, classes/IDs, color and typography, and the box model).

This profile represents a blend of standards awareness, practical markup, and foundational styling and accessibility skills needed to produce maintainable entry-level web content.

Block 1: HTML Fundamentals

Weight: 15% of total exam

Minimum Coverage – the Candidate can:

- Assemble a valid HTML page with `<!DOCTYPE html>`, language on `<html>`, and essential `<head>` metadata (e.g., `<meta charset="utf-8">`, `title`).
- Distinguish block-level and inline elements and choose appropriately for meaning, not appearance.
- Use entities for reserved characters and explain the role of UTF-8 character encoding.
- Document code with HTML comments.
- Validate the basic page skeleton.

Block 2: Text Formatting and Structure

Weight: 20% of total exam

Minimum Coverage – the Candidate can:

- Apply emphasis and inline semantics (e.g., `<strong>`, `<em>`, `<abbr>`, `<code>`) appropriately.
- Structure content with a clear heading outline, paragraphs, and meaningful separation (`<br>`, `<hr>` used sparingly).
- Create lists (ordered, unordered, definition) and simple data tables with captions and headers.
- Use quotation and code elements (`<blockquote>`, `<q>`, `<pre>`, `<kbd>`, `<samp>`) to convey intent.

Block 3: Multimedia and Hyperlinks

Weight: 20% of total exam

Minimum Coverage – the Candidate can:

- Embed images with informative alt text and basic responsiveness (`width`/`height`, `srcset`/`sizes` when appropriate).
- Create hyperlinks (internal, external, email, telephone) with clear anchor text and safe attributes (e.g., `rel="noopener"` with `target="_blank"`).
- Embed audio and video with controls, compatible sources, and captions/subtitles via `<track>`.
- Use figures and captions, image maps, and iframes responsibly
- Add favicons for basic site identity.

Block 4: Forms and Styling

Weight: 25% of total exam

Minimum Coverage – the Candidate can:

- Build forms with labels, common input types, selects, and textareas.
- Choose GET vs. POST appropriately and set `action`/`enctype` when needed.
- Group related fields with `<fieldset>`/`<legend>` and apply basic validation attributes (`required`, `min`/`max`, `pattern`, `maxlength`).
- Apply CSS via inline/internal styles for simple scenarios.
- Use classes and IDs to target elements predictably.
- Use color and typography for readability and apply the box model (`border`, `paddi`ng, `margin; box-sizing: border-box;`) for consistent spacing.

Block 5: Accessibility, Best Practices, and Modern HTML

Weight: 20% of total exam

Minimum Coverage – the Candidate can:

- Apply accessibility fundamentals: semantic landmarks, labels and associations, visible focus, and sufficient contrast.
- Use ARIA roles and states only when native semantics are insufficient, without creating conflicts.
- Organize documents with modern HTML5 structural elements and add basic structured data where appropriate.
- Demonstrate awareness of HTML platform features (e.g., Web Storage), SVG basics, simple performance practices, and accessibility testing methods.

Passing Requirement

To pass the WDE exam, a candidate must achieve a **cumulative average score of at least 75%** across all exam blocks.

WDE-40-01 Exam Structure Summary

The WDE™ exam consists of **40 single-select and multiple-select items**, each designed to assess a candidate's ability to write semantic HTML and apply foundational CSS to create well-structured, responsive, and accessible web pages.

Every item is worth a maximum of **10 points**, regardless of its format. After the exam is completed, the total raw score is normalized, and the candidate's performance is reported as a percentage.

The exam covers five blocks, each weighted proportionally based on the number and complexity of items. To pass the WDE exam, candidates need an overall average of **75% or higher** across all blocks.

Block Number	Block Name	Number of Items	Weight
1	HTML Fundamentals	6	15%
2	Text Formatting & Structure	8	20%
3	Multimedia & Hyperlinks	8	20%
4	Forms & Styling	10	25%
5	Accessibility, Best Practices, and Modern HTML	8	20%
	Total	**40**	**100%**