# The Definitive Guide to the Top 100 HTML Interview Questions

## Introduction

HyperText Markup Language (HTML) has evolved significantly from its inception as a simple system for structuring academic documents. Today, it stands as the fundamental cornerstone of the Open Web Platform, providing the structural and semantic foundation for nearly every web page and application. A deep understanding of HTML is no longer a trivial prerequisite for web developers but a critical competency that distinguishes proficient engineers from novices. Modern technical interviews reflect this reality, moving beyond simple tag memorization to probe a candidate's comprehension of the deeper principles that govern the web: semantics, accessibility, and performance.

This guide is designed to provide an exhaustive and scholarly exploration of the 100 most important and frequently asked HTML interview questions. Its purpose is to equip aspiring developers with a profound understanding of modern HTML principles. The following sections are structured to build knowledge systematically, from the core anatomy of an HTML document to the advanced APIs and optimization techniques that power contemporary web applications. The objective is not merely to provide answers but to foster a comprehensive understanding of the "what" and the "why" behind each concept, thereby enabling developers to articulate their knowledge with confidence and precision in any technical interview setting.

## Section I: Core Concepts & Document Structure

This foundational section deconstructs the essential anatomy of an HTML document. It covers the syntax, core components, and the hierarchical model that browsers use to interpret and render web pages. A firm grasp of these fundamentals is the absolute prerequisite for building any content for the web.

### 1. What does HTML stand for and what is its purpose?

HTML stands for **HyperText Markup Language**.1 It is the standard language used to create and structure the content of web pages.4 Its purpose is not to program logic but to "mark up" or describe the content's structure and semantic meaning.

* **HyperText** refers to the ability to create links that connect web pages to one another, forming the interconnected "web" of information.4
* **Markup Language** refers to the system of using tags to define elements, such as headings, paragraphs, images, and lists, which tells the browser how to organize and display the content.3

Essentially, HTML serves as the skeleton of a webpage, providing the foundational structure upon which styling (CSS) and interactivity (JavaScript) are applied.

HTML

<!DOCTYPE **html**>  
<html>  
<head>  
 <title>Page Title</title>  
</head>  
<body>  
 <h1>This is a Heading</h1>  
 <p>This is a paragraph of text.</p>  
 <a href="https://example.com">This is a link.</a>  
</body>  
</html>

### 2. Describe the basic structure of an HTML document.

A well-formed HTML document follows a consistent, hierarchical structure. This structure is essential for browsers to parse and render the page correctly.1 The primary components are:

1. **<!DOCTYPE html>:** A declaration that defines the document type.
2. **<html> Element:** The root element that wraps all other content on the page.
3. **<head> Element:** A container for metadata about the page.
4. **<body> Element:** A container for all the visible content of the page.

HTML

<!DOCTYPE **html**>  
<html lang="en">  
 <head>  
 <meta charset="UTF-8">  
 <title>Document Title</title>  
 </head>  
 <body>  
 <h1>My First Web Page</h1>  
 <p>Content for the user to see.</p>  
 </body>  
</html>

### 3. What do the <!DOCTYPE> declaration and lang attribute do?

The <!DOCTYPE> declaration is a critical instruction for the web browser, not an HTML tag itself.4 Its primary function is to inform the browser about the version of HTML the document is written in.1 For modern web pages, the declaration is simple:

<!DOCTYPE html>.2

This declaration is the foundational switch that dictates the browser's entire rendering behavior. Its presence ensures the browser uses **"standards mode,"** which interprets the HTML and CSS according to the latest W3C specifications, leading to consistent and predictable rendering across different browsers.11 If the

<!DOCTYPE> is omitted or incorrect, the browser falls back into **"quirks mode,"** a backward-compatibility mode that emulates the non-standard behaviors of older browsers like Internet Explorer 5.11 This can cause significant layout inconsistencies and bugs. Therefore, this single line of code can be seen as the developer's contract with the browser, promising to adhere to standards in exchange for predictable rendering.

The lang attribute, typically placed on the <html> tag (e.g., <html lang="en">), specifies the primary language of the document's content.1 This is crucial for:

* **Accessibility:** It allows screen readers to switch to the correct language profile to pronounce the text accurately.
* **Search Engines:** It helps search engines understand the language of the content, which can improve search results for users filtering by language.
* **Browsers:** It can influence browser functions like translation prompts or spell-checking.

HTML

<!DOCTYPE **html**>  
<html lang="en-US">  
<head>  
 <title>A Properly Declared Page</title>  
</head>  
<body>  
 <p>This content is in American English.</p>  
</body>  
</html>

### 4. What is the difference between the <head> and <body> tags?

The <head> and <body> elements are the two main children of the <html> root element, and they serve entirely different purposes.1

* **The <head> Element:** This section contains metadata—information *about* the HTML document that is not displayed directly on the page.1 It is processed by the browser before the visible content is rendered. Common elements found within the  
  <head> include:
  + <title>: The title of the page, shown in the browser tab and search engine results.
  + <meta>: Metadata such as character encoding, viewport settings, and page description.
  + <link>: Links to external resources, most commonly CSS stylesheets.
  + <script>: Can be used to include JavaScript, though it is often placed elsewhere for performance reasons.
  + <style>: For including internal CSS.
* **The <body> Element:** This section contains all the content that is visible to the user on the web page.1 This includes text, headings, paragraphs, images, videos, links, forms, and all other structural components that form the user interface. There can only be one  
  <body> element in a document.15

HTML

<!DOCTYPE **html**>  
<html lang="en">  
<head> <meta charset="UTF-8">  
 <title>Head vs. Body</title>  
 <link rel="stylesheet" href="styles.css">  
</head>  
<body> <h1>Welcome to the Page</h1>  
 <p>This text is visible to the user.</p>  
</body>  
</html>

### 5. Can you explain the purpose of <meta> tags in HTML?

Meta tags (<meta>) provide structured metadata about an HTML document.1 This information is not displayed on the page itself but is machine-readable and used by browsers, search engines, and other web services.7 Key meta tags include:

* **charset:** Specifies the character encoding for the document. UTF-8 is the universal standard and is essential for ensuring text displays correctly across all languages and symbols.1
  + Example: <meta charset="UTF-8">
* **viewport:** This is critical for responsive web design. It tells the browser how to control the page's dimensions and scaling, ensuring the site looks good on all devices, from desktops to mobile phones.1
  + Example: <meta name="viewport" content="width=device-width, initial-scale=1.0">
* **description:** Provides a brief summary of the page's content. Search engines often use this description in their search results snippets, making it vital for SEO.1
  + Example: <meta name="description" content="An in-depth guide to HTML interview questions.">
* **author:** Specifies the author of the document.
  + Example: <meta name="author" content="John Doe">
* **Open Graph & Twitter Cards:** These are specialized meta tags used by social media platforms (like Facebook and X, formerly Twitter) to create rich previews when a link to the page is shared.

HTML

<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <meta name="description" content="This page explains the purpose of meta tags.">  
 <meta name="author" content="Web Development Experts">  
</head>

### 6. How do you link a CSS file to an HTML document?

The standard and most effective way to apply styles to an HTML document is by linking to an external CSS stylesheet. This is done using the <link> tag placed within the <head> section of the document.1

The <link> tag requires two key attributes:

* rel="stylesheet": This attribute specifies the relationship between the HTML document and the linked file. In this case, it indicates that the file is a stylesheet.
* href="path/to/styles.css": This attribute provides the path to the CSS file. The path can be relative (to the current HTML file) or absolute (a full URL).

Placing the <link> tag in the <head> is a critical performance and user experience practice. The browser parses HTML from top to bottom. When it encounters a stylesheet link in the <head>, it begins fetching the CSS file in parallel while continuing to parse the rest of the HTML. This allows the browser to construct the CSS Object Model (CSSOM) alongside the Document Object Model (DOM). Having both ready early enables the browser to build the render tree and display the styled page to the user more quickly, avoiding a flash of unstyled content (FOUC).

HTML

<!DOCTYPE **html**>  
<html>  
<head>  
 <title>Linking CSS</title>  
 <link rel="stylesheet" href="styles.css">  
</head>  
<body>  
 <h1>This heading will be styled by styles.css</h1>  
</body>  
</html>

### 7. How do you link a JavaScript file to an HTML document?

JavaScript is included in an HTML document using the <script> tag.1 The most common and recommended method is to link to an external JavaScript file using the

src attribute.

The conventional best practice is to place <script> tags just before the closing </body> tag.14 This placement is a direct consequence of the browser's rendering pipeline and is a fundamental performance optimization. When the HTML parser encounters a

<script> tag (without async or defer attributes), it must halt parsing, fetch the script from the network, execute it, and only then resume parsing the rest of the document. If a script is placed in the <head>, it blocks the rendering of all visible content below it. This can result in a blank page for the user until the script is fully processed, leading to a poor perceived performance. By placing scripts at the end of the <body>, the browser can parse and render the entire visible page first, ensuring the user sees content as quickly as possible. The script can then load and execute afterward, adding interactivity to the already-visible elements.

HTML

<!DOCTYPE **html**>  
<html>  
<head>  
 <title>Linking JavaScript</title>  
</head>  
<body>  
 <p id="demo">This text will be changed by JavaScript.</p>  
  
 <script src="app.js"></script>  
</body>  
</html>

### 8. What is the difference between an HTML element, tag, and attribute?

These three terms are the fundamental building blocks of HTML, but they are not interchangeable.2

* **Tag:** An HTML tag is the markup syntax used to denote the start and end of an element. It consists of a keyword enclosed in angle brackets. For example, <p> is a start tag and </p> is an end tag.5
* **Element:** An HTML element is the complete component, consisting of the start tag, the content, and the end tag.5 For example,  
  <p>This is some text.</p> is a paragraph element. Some elements, known as void elements, are self-contained and do not have content or an end tag (e.g., <img>).
* **Attribute:** An attribute provides additional information about an element and is always specified in the start tag.4 Attributes come in name/value pairs, like  
  name="value". For example, in <a href="https://example.com">Click me</a>, href is an attribute of the <a> element, and its value is the URL.

HTML

<p class="info-text">This is a paragraph.</p>

### 9. What are void elements in HTML?

Void elements, also known as empty or self-closing elements, are HTML elements that cannot have any child nodes (i.e., they cannot contain any content or other elements).5 Because they are empty, they only have a start tag and do not require a closing tag.10

In HTML5, the trailing slash (/) is optional, but it is required in stricter syntaxes like XHTML. Common void elements include:

* <br>: A line break.
* <hr>: A thematic break (horizontal rule).
* <img>: An image.
* <input>: A form input control.
* <link>: A link to an external resource.
* <meta>: Metadata about the document.

HTML

<p>This is the first line.<br>This is the second line.</p>  
<hr>  
<img src="logo.png" alt="Company Logo">

### 10. How do you add a comment in HTML and why are they used?

HTML comments are added using the syntax ``.1 Anything between these delimiters will not be rendered by the browser or processed by parsers.

Comments are a crucial tool for developers and serve several purposes 1:

* **Documentation:** To explain complex or non-obvious parts of the markup, making the code easier for other developers (or a future self) to understand.
* **Instructions:** To leave notes or reminders for team members about tasks to be completed.
* **Debugging:** To temporarily "comment out" a block of HTML to see how the page renders without it, which is a common technique for isolating issues.
* **Organization:** To add labels that visually separate large sections of code, improving readability.

Best practices for commenting include writing clear, concise comments that explain the "why" behind the code, not just the "what," and ensuring comments are kept up-to-date as the code evolves.

HTML

<header>  
 <h1>Website Title</h1>  
</header>

### 11. What are HTML entities?

HTML entities are reserved strings of text that are used to display characters that have special meaning in HTML or are not easily typed on a standard keyboard.4 An entity starts with an ampersand (

&) and ends with a semicolon (;).

They are necessary for two primary reasons:

1. **To display reserved characters:** Characters like < and > are part of the HTML syntax itself. To display them as literal text on the page, they must be "escaped" using their corresponding entities.
2. **To display special symbols:** Entities provide a way to display symbols like the copyright sign (©), the registered trademark symbol (®), or mathematical symbols that may not be present on all keyboards.

Common HTML entities include:

* < for the less-than sign (<)
* > for the greater-than sign (>)
* & for the ampersand (&)
* " for a double quote (")
* © for the copyright symbol (©)
* for a non-breaking space, which prevents an automatic line break at its position.

HTML

<p>To write the <h1> tag, you must use <h1>.</p>  
<p>Copyright © 2025. All rights reserved.</p>

### 12. What is the Document Object Model (DOM)?

The Document Object Model (DOM) is a programming interface for web documents.4 When a web browser loads an HTML document, it creates an in-memory, tree-like representation of that document's structure. This tree is the DOM. Each node in the tree represents a part of the document, such as an element, an attribute, or a piece of text.

The DOM is not the HTML source code itself; it is a live, dynamic model of the page. It serves as the crucial bridge between the static HTML document and scripting languages like JavaScript. Using the DOM, JavaScript can:

* Access and select any element on the page.
* Modify the structure, style, and content of the page.
* Create and delete elements.
* React to user events (like clicks or key presses).

Essentially, the DOM makes the web page interactive. Any change made to the DOM by a script is immediately reflected in what the user sees in the browser.

HTML

<!DOCTYPE **html**>  
<html>  
<body>  
 <h1 id="title">Hello World</h1>  
</body>  
</html>  
  
<script>  
 // Using the DOM to access an element and change its content  
 const heading = document.getElementById('title');  
 heading.textContent = 'Hello DOM!';  
</script>

### 13. What is the advantage of collapsing white space?

In HTML, browsers treat any sequence of one or more whitespace characters (spaces, tabs, newlines) as a single space.5 This behavior is known as "collapsing white space."

The primary advantage of this feature is that it gives developers the freedom to format their HTML source code for maximum readability without affecting the final rendered output.15 Developers can use indentation and line breaks to neatly structure their code and reflect the nesting of elements, making the markup much easier to read, understand, and maintain. If whitespace were not collapsed, this kind of source code formatting would result in large, unwanted gaps in the rendered page.

HTML

<p>  
 This text  
 is formatted  
 for readability in the source code.  
</p>  
  
<p>This text is formatted for readability in the source code.</p>

### 14. How do you serve a page in multiple languages?

Serving a page with content in multiple languages involves several layers of implementation, from the HTML markup to server-side logic.1

1. **Language Declaration (lang attribute):** As mentioned earlier, the lang attribute on the <html> tag is the first step. It should be set to the language of the content being displayed (e.g., <html lang="es"> for Spanish).
2. **Alternate Language Links (hreflang):** For SEO, it is crucial to tell search engines about the different language versions of a page. This is done using <link> tags with the hreflang attribute in the <head> section. Each link points to a URL for a specific language version of the current page.
3. **Content Delivery:** The server must be configured to deliver the correct language version to the user. This can be based on:
   * **URL Structure:** Using different URLs for each language (e.g., example.com/en/page and example.com/es/page). This is the recommended approach for SEO.
   * **User Preference:** Allowing the user to select their preferred language via a dropdown menu, with the choice stored in a cookie or localStorage.
   * **Browser Settings:** Detecting the user's preferred language from the Accept-Language HTTP header sent by their browser.

HTML

<head>  
 <title>My Page</title>  
 <link rel="alternate" hreflang="en" href="https://example.com/en/page">  
 <link rel="alternate" hreflang="de" href="https://example.com/de/page">  
 <link rel="alternate" hreflang="x-default" href="https://example.com/en/page">  
</head>

### 15. What is the difference between HTML and XHTML?

HTML (HyperText Markup Language) and XHTML (Extensible HyperText Markup Language) are both markup languages used for creating web pages, but they have a key difference in their syntax rules.2

* **HTML:** Has a more lenient, forgiving syntax. Browsers are designed to parse and render HTML even if it contains minor errors, such as unclosed tags.
* **XHTML:** Is an application of XML, which means it must follow the strict, well-formed syntax rules of XML.2 These rules include:
  + All elements must be properly nested.
  + All elements must be closed (void elements must be self-closed, e.g., <br />).
  + Element and attribute names must be in lowercase.
  + Attribute values must be quoted.

XHTML was developed to create a more standardized and interoperable web. However, with the advent of HTML5, which adopted a clear parsing algorithm that handles errors gracefully while encouraging well-formed markup, the need for the strictness of XHTML has largely diminished. Today, HTML5 is the dominant standard for web development.

HTML

<!DOCTYPE **html** **PUBLIC** "-//W3C//DTD XHTML 1.0 Strict//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">  
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">  
<head>  
 <title>An XHTML Document</title>  
</head>  
<body>  
 <p>This is a paragraph.<br />It has a line break.</p>  
</body>  
</html>

## Section II: Semantic Markup for Meaningful Content

This section champions the "why" behind semantic HTML, transitioning from a purely structural view to one that imbues content with meaning. It will cover the benefits for accessibility (A11y), search engine optimization (SEO), and code maintainability. Using elements for their intended purpose is a hallmark of a modern, professional web developer.

### 16. What is semantic HTML?

Semantic HTML is the practice of using HTML markup to reinforce the meaning and structure of the information in a webpage, rather than simply defining its presentation or appearance.1 It involves choosing the HTML element that best describes the content it contains.

For example, using <h1> for the main page heading, <p> for a paragraph, and <li> for a list item are all examples of semantic markup. In contrast, using a <div> for everything and relying solely on CSS classes to describe content (e.g., <div class="main-heading">) is non-semantic. While it might look the same visually, it lacks the intrinsic meaning that semantic tags provide.

HTML

<div class="header">My Website</div>  
<div class="nav">  
 <span class="nav-item">Home</span>  
 <span class="nav-item">About</span>  
</div>  
  
<header>  
 <h1>My Website</h1>  
</header>  
<nav>  
 <ul>  
 <li><a href="/">Home</a></li>  
 <li><a href="/about">About</a></li>  
 </ul>  
</nav>

### 17. Why are semantic HTML tags important?

The importance of using semantic HTML tags extends far beyond code aesthetics; it is a foundational practice with tangible benefits for multiple stakeholders.1

1. **Accessibility (A11y):** Semantic HTML is the primary API for assistive technologies like screen readers. A screen reader is a program that interprets the DOM, not the visual layout. A <nav> element explicitly tells the screen reader, "This is a block of navigation links," allowing the user to easily find or skip it. An <h1> tag identifies the main topic of the page. Without these semantic cues, a user with a visual impairment would be presented with a flat, undifferentiated block of text, making the page difficult or impossible to navigate.
2. **Search Engine Optimization (SEO):** Search engine crawlers are automated bots that analyze a page's structure to understand its content hierarchy and relevance. They use semantic elements like <article>, <h1>, and <strong> to weigh the importance of different pieces of content, which directly influences search rankings.11 A well-structured, semantic page is more easily understood by search engines and is therefore more likely to rank higher.
3. **Code Maintainability:** Semantic markup makes the code self-documenting and easier for developers to read and understand. When another developer encounters a <header>, <main>, or <footer> tag, its purpose is immediately clear, which is not the case with a sea of generic <div> elements. This improves collaboration and reduces the time needed to make future updates.
4. **Future-Proofing:** Semantic HTML is more likely to be compatible with future web technologies and devices, as it is based on a standardized description of content rather than a specific visual presentation.

In essence, writing semantic HTML is not just a "best practice"; it is a core responsibility of a web developer to ensure their content is universally accessible, discoverable, and maintainable. It is a form of API design for non-human user agents.

### 18. Explain the purpose of key HTML5 semantic elements like <header>, <footer>, <nav>, <main>, <article>, <section>, and <aside>.

HTML5 introduced a suite of new elements designed to give developers a standardized vocabulary for describing the main sections of a webpage.3

* **<header>:** Represents introductory content for its nearest ancestor sectioning content or for the page as a whole. This typically includes a logo, the site name, a search form, or the main navigation.8
* **<footer>:** Represents the footer for its nearest ancestor sectioning content or for the page as a whole. This usually contains information like copyright notices, contact information, and links to related documents.8
* **<nav>:** Represents a section of the page whose purpose is to provide navigation links, either within the current document or to other documents. It is intended for major navigation blocks.8
* **<main>:** Represents the dominant, main content of the <body> of a document. There should only be one <main> element per page, and it should not be a descendant of an <article>, <aside>, <footer>, <header>, or <nav> element.8
* **<article>:** Represents a self-contained, complete composition in a document that is intended to be independently distributable or reusable (e.g., in syndication). Examples include a forum post, a magazine or newspaper article, or a blog entry.6
* **<section>:** Represents a generic standalone section of a document, which doesn't have a more specific semantic element to represent it. It typically has a heading and groups a thematic set of content.6
* **<aside>:** Represents a portion of a document whose content is only tangentially related to the document's main content. Asides are often represented as sidebars or call-out boxes.8

HTML

<body>  
 <header>  
 <h1>My Blog</h1>  
 <nav>  
 <ul>  
 <li><a href="/">Home</a></li>  
 <li><a href="/archive">Archive</a></li>  
 </ul>  
 </nav>  
 </header>  
 <main>  
 <article>  
 <h2>The Importance of Semantic HTML</h2>  
 <section>  
 <h3>For Accessibility</h3>  
 <p>Details about accessibility...</p>  
 </section>  
 <section>  
 <h3>For SEO</h3>  
 <p>Details about SEO...</p>  
 </section>  
 </article>  
 <aside>  
 <h3>About the Author</h3>  
 <p>Bio goes here...</p>  
 </aside>  
 </main>  
 <footer>  
 <p>© 2025 My Blog</p>  
 </footer>  
</body>

### 19. What is the difference between <article> and <section>?

This is a common point of confusion, but the distinction is crucial for correct semantic structuring.6

The key difference is **self-containment**.

* An **<article>** should make sense on its own, as a complete unit of content. If you were to syndicate it in an RSS feed, it should be fully understandable without the context of the rest of the page. A blog post is the canonical example. An <article> can contain <section>s.
* A **<section>** is a way to group related content together within a larger document. It is not necessarily self-contained. Its content is thematically related, but it might not make sense if removed from the context of the page. Think of chapters in a book or different tabbed panels on a settings page. A <section> can contain <article>s (e.g., a "Latest News" section containing multiple article summaries).

A simple test is to ask: "Would this content make sense if it were the only thing on the page or in an RSS feed?" If the answer is yes, it's likely an <article>. If not, it's likely a <section>.

### 20. When should you use <figure> and <figcaption>?

The <figure> element is used to enclose a unit of content, such as an image, illustration, diagram, code snippet, or quote, that is referenced from the main content but can be moved to another part of the page (or to an appendix) without affecting the main flow.1

The <figcaption> element provides a caption or legend for the content within the <figure>. It is semantically linked to the <figure> element, which provides a powerful accessibility benefit: screen readers will announce the caption as being directly associated with the figure's content. This is a significant improvement over placing a <p> tag next to an <img>, where the association is only visual.

HTML

<figure>  
 <img src="chart.png" alt="A bar chart showing quarterly profits.">  
 <figcaption>Fig. 1 - Quarterly Profits for Fiscal Year 2025.</figcaption>  
</figure>  
  
<figure>  
 <pre><code>function hello() {  
 console.log("Hello, world!");  
}</code></pre>  
 <figcaption>A simple JavaScript function to log a greeting.</figcaption>  
</figure>

### 21. What is the difference between <b> and <strong>, and <i> and <em>?

This question tests the understanding of semantic meaning versus presentational styling.1 Visually, a browser renders

<b> and <strong> text as bold, and <i> and <em> text as italicized. However, their semantic meanings are entirely different.

* **<b> (Bold) vs. <strong> (Strong Importance):**
  + The <b> tag is a **presentational** element. It simply applies a bold font weight to the text for visual emphasis without implying any extra importance.1 It should be used when you want to draw attention to text stylistically, such as for a product name in a review.
  + The <strong> tag is a **semantic** element. It indicates that the enclosed text has strong importance, seriousness, or urgency.1 Screen readers may change their voice inflection to convey this importance, and search engines may give the text slightly more weight.
* **<i> (Italic) vs. <em> (Emphasis):**
  + The <i> tag is **presentational**. It renders text in an italic style and is typically used for things like a thought, a technical term, a foreign phrase, or a ship's name, where the text is offset from the normal prose but doesn't carry added emphasis.4
  + The <em> tag is **semantic**. It indicates stress emphasis on the enclosed text.4 Like  
    <strong>, screen readers may alter their pronunciation to reflect this emphasis.

**Rule of thumb:** If the meaning of the sentence changes when the tag is removed, use <strong> or <em>. If only the styling is lost, <b> or <i> may be appropriate.

HTML

<p>This is a <b>bold</b> word for visual effect.</p>  
<p><strong>Warning:</strong> Do not touch the live wire.</p>  
  
<p>The term <i>Homo sapiens</i> is Latin.</p>  
<p>You <em>must</em> complete this task today.</p>

### 22. What is the difference between <div> and <span>?

<div> and <span> are two of the most commonly used HTML elements, but they serve as generic containers with one fundamental difference: their default display behavior.7

* **<div> (Division):**
  + This is a **block-level** element.
  + By default, it starts on a new line and takes up the full width available to it, pushing subsequent content to the next line.
  + It is used to group larger chunks of content or other block-level elements together, often to create major layout sections of a page (e.g., a container, a row, a card).
* **<span>:**
  + This is an **inline** element.
  + By default, it does not start on a new line and only takes up as much width as its content requires.
  + It is used to group a small, inline portion of content, such as a few words within a paragraph, typically to apply specific styling (like a different color) or to hook into with JavaScript, without disrupting the flow of the text.

HTML

<div>This is a div.</div>  
<div>This is another div.</div>  
  
<p>This is a paragraph with a <span style="color: red;">red span</span> inside it.</p>

### 23. What are physical and logical tags?

This is a broader way of categorizing the distinction between presentational and semantic tags.10

* **Physical Tags:** These are tags that define the visual appearance of the text. They are presentational in nature. Examples include <b> (bold), <i> (italic), <u> (underline), and <font> (deprecated). They tell the browser *how* to display the content.
* **Logical Tags:** These are tags that define the semantic meaning or structure of the content. They describe *what* the content is. Examples include <strong> (strong importance), <em> (emphasis), <code> (a piece of computer code), and <address> (contact information).

Modern web development standards strongly favor the use of logical tags over physical tags. Styling should be handled by CSS, while HTML should focus on describing the content's meaning and structure.

### 24. What is the purpose of the small, s, and mark tags?

These are less common but useful semantic text-level tags:

* **<small>:** Represents side comments or "small print," such as copyright notices or legal disclaimers. It renders the text in a smaller font size but semantically indicates that the content is of a lower importance or is supplementary.1
* **<s> (Strikethrough):** Represents content that is no longer accurate or relevant. It is semantically different from <del> (deleted text), which is used to indicate text that has been removed from a document. The <s> tag is for things that are still visible but marked as incorrect, like an old price on an e-commerce site.
* **<mark>:** Represents a run of text in one document which is marked or highlighted for reference purposes, due to its relevance in another context. A common use case is to highlight search terms within a results page.1 By default, browsers render this with a yellow background.

HTML

<footer>  
 <p><small>© 2025 Company Inc.</small></p>  
</footer>  
<p>Price: <s>$99.99</s> $79.99</p>  
<p>Your search for "semantic" returned the following result: The use of <mark>semantic</mark> HTML is a best practice.</p>

### 25. How would you semantically mark up an interview or conversation?

This question tests the ability to apply semantic principles to a less common content structure. There is no single <interview> tag, so one must compose a solution from existing semantic elements. A good approach would be to use a description list (<dl>) or a series of paragraphs with strong or cite tags.22

Using a <dl> is often a strong choice because a conversation is fundamentally a series of terms (the speaker) and descriptions (what they said).

HTML

<dl>  
 <dt>Interviewer</dt>  
 <dd>What is your greatest strength?</dd>  
  
 <dt>Candidate</dt>  
 <dd>My ability to apply semantic HTML principles to solve complex structural problems.</dd>  
</dl>

Another valid approach could involve using <p> tags and using <strong> or <b> to identify the speaker.

HTML

<p><strong>Interviewer:</strong> What is your greatest strength?</p>  
<p><strong>Candidate:</strong> My ability to apply semantic HTML principles to solve complex structural problems.</p>

### 26. What is the purpose of the main element?

The <main> element represents the dominant, primary content of the <body> of a document.7 The content inside the

<main> element should be unique to that document and should not include content that is repeated across multiple pages, such as sidebars, site navigation, headers, or footers.

By definition, there must be only one <main> element in a document, and it must not be a descendant of an <article>, <aside>, <footer>, <header>, or <nav> element.

Using <main> provides a powerful landmark for assistive technologies. It allows screen reader users to jump directly to the primary content of the page, bypassing repetitive navigation links and headers.

HTML

<body>  
 <header>...</header>  
 <nav>...</nav>  
 <main>  
 <h1>Main Article Title</h1>  
 <p>This is the core content of the page...</p>  
 </main>  
 <footer>...</footer>  
</body>

### 27. What is the difference between id and class attributes?

id and class are global attributes used to provide identifiers for elements, but they have a fundamental difference in their uniqueness constraint.5

* **id (Identifier):**
  + The id attribute specifies a **unique** identifier for an element within the entire HTML document.
  + Each id value must be used only once per page.
  + It is primarily used as a hook for JavaScript to select a specific, single element (document.getElementById()) or as a target for internal page links (fragment identifiers, e.g., <a href="#section1">).
* **class:**
  + The class attribute specifies one or more class names for an element.
  + The same class name can be used on **multiple** elements across the page.
  + An element can also have multiple class names, separated by spaces.
  + It is primarily used as a hook for CSS to apply the same styles to a group of elements or for JavaScript to select a group of elements (document.getElementsByClassName()).

**In summary:** Use id when you need to uniquely identify one element. Use class when you want to group multiple elements to share styling or behavior.

HTML

<h1 id="main-title">My Page</h1>  
  
<p class="highlight">This is important text.</p>  
<ul>  
 <li class="highlight">This is an important item.</li>  
</ul>

### 28. Are HTML tags and elements the same thing?

No, they are distinct concepts, though often used interchangeably in casual conversation.2

* An **HTML tag** is the syntax used to mark the beginning and end of an element (e.g., <h1> and </h1>).
* An **HTML element** is the entire unit, which includes the start tag, the content inside, and the end tag (e.g., <h1>Main Heading</h1>).

For void elements like <img>, the element consists of just the single tag and its attributes.

### 29. What is the purpose of the hidden attribute?

The hidden attribute is a boolean attribute that, when present, indicates that an element is not yet, or is no longer, relevant. Browsers will not render elements that have the hidden attribute specified.2

However, the element is still part of the DOM and can be accessed and manipulated by JavaScript. This makes it a useful tool for hiding content that should only be revealed in response to a user action (e.g., showing a dialog box) or for keeping content available to screen readers while being visually hidden (though CSS techniques are often preferred for this). It is semantically more appropriate than using CSS display: none; when the content's relevance is conditional.

HTML

<p>You can see this paragraph.</p>  
<p hidden>You cannot see this paragraph until the 'hidden' attribute is removed by JavaScript.</p>  
  
<script>  
 // Example: A button could trigger this code to reveal the hidden content  
 setTimeout(() => {  
 const hiddenP = document.querySelector('p[hidden]');  
 if (hiddenP) {  
 hiddenP.removeAttribute('hidden');  
 }  
 }, 3000);  
</script>

### 30. What is the lang attribute used for?

The lang global attribute is used to specify the language of an element's content.1 It is most commonly used on the root

<html> element to declare the primary language for the entire page.

However, it can also be used on any other element to indicate that a specific part of the content is in a different language. This is important for:

* **Accessibility:** It signals to screen readers to switch language profiles for correct pronunciation.
* **Search Engines:** It helps search engines correctly index content in different languages.
* **Browser Behavior:** It can affect hyphenation, spell-checking, and translation services offered by the browser.

HTML

<html lang="en">  
<body>  
 <p>This paragraph is in English.</p>  
 <p lang="fr">Ce paragraphe est en français.</p>  
</body>  
</html>

## Section III: Structuring Content - Text, Lists, and Tables

This section focuses on the workhorse elements for organizing the majority of web content: text, lists, and tabular data. The emphasis will be on correct structural usage and accessibility, ensuring that content is not only visually organized but also logically coherent for all users and machines.

### 31. How do you create headings, and why is the hierarchy important?

Headings are created using the <h1> through <h6> tags, where <h1> represents the highest level (most important) and <h6> represents the lowest level.1

The heading hierarchy is critically important for two main reasons:

1. **Accessibility:** Screen reader users rely on the heading structure to understand the layout of a page and to navigate through it. They can listen to a list of all headings to get an outline of the content and jump directly to the section that interests them. A logical and consistent hierarchy (<h1> followed by <h2>, which is followed by <h3>, etc., without skipping levels) is essential for this to work effectively.
2. **SEO:** Search engines analyze the heading hierarchy to determine the main topics and subtopics of a page. The content within an <h1> tag is considered the most important descriptor of the page's content. A clear, logical heading structure helps search engines index the page more accurately, which can lead to better search rankings.

As a best practice, a page should have only one <h1> element, which serves as the main title for the entire page.7

HTML

<h1>Main Page Title</h1>  
<p>Introduction to the topic...</p>  
<h2>Section 1: A Major Subtopic</h2>  
<p>Content for section 1...</p>  
<h3>Subsection 1.1: A Detail of the Subtopic</h3>  
<p>Details for subsection 1.1...</p>  
<h2>Section 2: Another Major Subtopic</h2>  
<p>Content for section 2...</p>

### 32. How do you create a paragraph and a line break?

Paragraphs are created with the <p> tag. This is a block-level element that groups related sentences and is the most common way to structure blocks of text.1 Browsers automatically add some vertical space (margin) before and after each paragraph.

A line break is created with the <br> tag. This is a void element used to force a line break within a block of text, without starting a new paragraph.2 It should be used sparingly, typically only when the division of lines is semantically important, such as in poetry or mailing addresses. It should not be used to create vertical space between elements; CSS margins and padding are the correct tools for that purpose.

HTML

<p>This is a full paragraph of text. It will be separated from other paragraphs by a margin.</p>  
<p>This is a second paragraph.</p>  
  
<p>123 Web Dev Lane<br>  
 Codeville, HTML 54321</p>

### 33. What are the different types of lists in HTML?

HTML provides three types of lists for structuring content, each with a specific semantic purpose.1

1. **Unordered List (<ul>):** Used for a collection of items where the order does not matter. Browsers typically render these items with bullet points. Each item in the list is defined by an <li> (list item) tag.
2. **Ordered List (<ol>):** Used for a collection of items where the order is significant (e.g., a list of steps in a recipe or a ranked list). Browsers typically render these items with numbers. Each item is also defined by an <li> tag.
3. **Description List (<dl>):** Used for a list of terms and their corresponding descriptions (like a glossary or dictionary). It consists of three parts:
   * <dl>: The description list container.
   * <dt>: The description term.
   * <dd>: The description details.

HTML

<ul>  
 <li>Apples</li>  
 <li>Oranges</li>  
 <li>Bananas</li>  
</ul>  
  
<ol>  
 <li>First, gather your ingredients.</li>  
 <li>Next, mix them together.</li>  
 <li>Finally, bake for 30 minutes.</li>  
</ol>  
  
<dl>  
 <dt>HTML</dt>  
 <dd>HyperText Markup Language</dd>  
 <dt>CSS</dt>  
 <dd>Cascading Style Sheets</dd>  
</dl>

### 34. How do you create a nested list?

A nested list is a list that is contained within another list item. Any type of list can be nested inside another. To create one, you simply place a new <ul> or <ol> element inside an <li> element.12 This is a common way to create outlines or multi-level navigation menus.

HTML

<ul>  
 <li>Fruit  
 <ul>  
 <li>Apple</li>  
 <li>Banana</li>  
 </ul>  
 </li>  
 <li>Vegetables  
 <ol>  
 <li>Carrot</li>  
 <li>Broccoli</li>  
 </ol>  
 </li>  
</ul>

### 35. What is the difference between <blockquote> and <q>?

Both elements are used for quotations, but they differ in their scope and presentation.4

* **<blockquote>:**
  + This is a **block-level** element.
  + It is used for long, multi-line quotations that are set off from the main text.
  + Browsers typically render this element with indented margins on the left and right.
  + It can contain other block-level elements, like <p> tags.
* **<q> (Quote):**
  + This is an **inline** element.
  + It is used for short, in-sentence quotations.
  + Browsers automatically add quotation marks around the content of the <q> element, so you do not need to type them manually.

Both elements can take a cite attribute, whose value should be a URL pointing to the source of the quotation.

HTML

<p>As the W3C states:</p>  
<blockquote cite="https://www.w3.org/TR/html5/grouping-content.html#the-blockquote-element">  
 <p>The blockquote element represents content that is quoted from another source.</p>  
</blockquote>  
  
<p>He said, <q>Semantic HTML is essential</q>, and I agree.</p>

### 36. How do you create a table in HTML?

HTML tables are used to display tabular data in a structured grid of rows and columns. Creating an accessible and well-structured table involves several key elements.2

* <table>: The main container for the entire table.
* <thead>: (Optional but recommended) Groups the header content of the table.
* <tbody>: (Optional but recommended) Groups the main body content of the table.
* <tfoot>: (Optional but recommended) Groups the footer content of the table.
* <tr> (Table Row): Defines a row of cells in the table.
* <th> (Table Header): Defines a header cell. The content is typically bold and centered by default. It should be used for column and row titles.
* <td> (Table Data): Defines a standard data cell.
* <caption>: Provides a title or caption for the table, which is important for accessibility.

Using <thead>, <tbody>, and <tfoot> provides semantic structure and allows browsers to scroll the table body independently of the header and footer.

HTML

<table border="1">  
 <caption>Monthly Sales Figures</caption>  
 <thead>  
 <tr>  
 <th>Month</th>  
 <th>Sales</th>  
 </tr>  
 </thead>  
 <tbody>  
 <tr>  
 <td>January</td>  
 <td>$10,000</td>  
 </tr>  
 <tr>  
 <td>February</td>  
 <td>$12,000</td>  
 </tr>  
 </tbody>  
 <tfoot>  
 <tr>  
 <td>Total</td>  
 <td>$22,000</td>  
 </tr>  
 </tfoot>  
</table>

### 37. What are rowspan and colspan attributes used for?

The rowspan and colspan attributes are used on <th> or <td> elements to make a single cell span across multiple rows or columns, respectively.2

* **colspan="number":** Merges a cell with one or more cells to its right in the same row. The value "number" indicates how many columns the cell should span.
* **rowspan="number":** Merges a cell with one or more cells below it in subsequent rows. The value "number" indicates how many rows the cell should span.

These are useful for creating complex table layouts where headers or data points apply to multiple categories.

HTML

<table border="1">  
 <caption>Employee Schedule</caption>  
 <thead>  
 <tr>  
 <th>Name</th>  
 <th colspan="2">Availability</th>  
 </tr>  
 <tr>  
 <th></th> <th>Morning</th>  
 <th>Afternoon</th>  
 </tr>  
 </thead>  
 <tbody>  
 <tr>  
 <td>Alice</td>  
 <td rowspan="2">Available</td>  
 <td>Not Available</td>  
 </tr>  
 <tr>  
 <td>Bob</td>  
 <td>Available</td>  
 </tr>  
 </tbody>  
</table>

### 38. How can you improve the accessibility of HTML tables?

Making tables accessible is crucial for screen reader users to understand the relationships between data cells and their headers. The correct use of table elements is not about visual layout but about defining these data relationships.

1. **Use <caption>:** Always provide a <caption> for your table to describe its purpose.
2. **Use <th> for Headers:** Use <th> elements for all row and column headers, not <td> styled to look like a header.
3. **Use the scope Attribute:** This is the most important step for complex tables. The scope attribute on a <th> element explicitly tells assistive technologies what the header is for.2
   * scope="col": The header applies to all cells in that column.
   * scope="row": The header applies to all cells in that row.

A screen reader uses this structure to provide context. When it encounters a <td>, it can announce the corresponding <th> from that row and column, telling the user, for example, "Row: 'Chicago', Column: 'Population', Value: '2.7 million'". Without this structure, the user just hears a jumble of disconnected data points.

HTML

<table border="1">  
 <caption>City Information</caption>  
 <thead>  
 <tr>  
 <th scope="col">City</th>  
 <th scope="col">State</th>  
 <th scope="col">Population</th>  
 </tr>  
 </thead>  
 <tbody>  
 <tr>  
 <th scope="row">Chicago</th>  
 <td>Illinois</td>  
 <td>2.7 Million</td>  
 </tr>  
 <tr>  
 <th scope="row">Los Angeles</th>  
 <td>California</td>  
 <td>3.9 Million</td>  
 </tr>  
 </tbody>  
</table>

### 39. What is the difference between cellpadding and cellspacing?

These are older, presentational attributes for the <table> element that have been deprecated in HTML5 in favor of CSS properties.15

* **cellspacing:** Defines the space *between* adjacent cells in a table. In CSS, this is controlled by the border-spacing property on the <table> element.
* **cellpadding:** Defines the space *between* the content of a cell and the cell's border. In CSS, this is controlled by the padding property on <td> and <th> elements.

Understanding their historical purpose is useful, but modern development practice dictates that all styling, including table spacing and padding, should be managed with CSS.

### 40. How do you create a horizontal line in HTML?

A horizontal line, representing a thematic break or a separation between sections of content, is created using the <hr> (horizontal rule) tag.7 It is a void element, so it does not require a closing tag.

Semantically, <hr> should be used to indicate a shift in topic or a transition between different subjects within a piece of content. It should not be used purely for decorative purposes; for visual lines, CSS borders are the more appropriate tool.

HTML

<p>This is the first topic of discussion. It covers several points related to web standards.</p>  
<hr>  
<p>This is the second topic. Now we shift our focus to server-side technologies.</p>

### 41. What is the <pre> element used for?

The <pre> (preformatted text) element is used to display a block of text in which whitespace is preserved exactly as it is written in the HTML source code.19 Normally, browsers collapse multiple whitespace characters into a single space. The

<pre> tag overrides this behavior.

It is most commonly used for displaying computer code, poetry, or ASCII art, where indentation and line breaks are significant to the meaning and presentation of the content. Browsers typically render the content of a <pre> tag in a monospaced font. For marking up code, it is best practice to wrap the code itself in a <code> tag inside the <pre> tag for added semantic clarity.

HTML

<pre><code>  
function greet(name) {  
 return `Hello, ${name}!`;  
}  
  
console.log(greet('World'));  
</code></pre>

### 42. How do you create superscript and subscript text?

Superscript and subscript text can be created using the <sup> and <sub> tags, respectively.15

* **<sup> (Superscript):** Renders text smaller and raised above the baseline. It is used for things like exponents in mathematical formulas (e.g., E=mc2) or ordinal indicators (e.g., 1st).
* **<sub> (Subscript):** Renders text smaller and lowered below the baseline. It is used for things like footnotes or chemical formulas (e.g., H2O).

HTML

<p>The famous equation is E = mc<sup>2</sup>.</p>  
  
<p>The chemical formula for water is H<sub>2</sub>O.</p>

### 43. How do you align list elements?

Historically, list alignment might have been attempted with HTML attributes, but the modern and correct way to align list elements is with CSS.5 The

text-align property can be applied to the <ul> or <ol> element to align the text within each list item. For more complex alignment, such as aligning the bullet points or numbers themselves, properties like list-style-position or Flexbox/Grid on the list container can be used. Indentation of nested lists is handled automatically by the browser's default styles but can be customized with CSS padding or margin on the nested list element.

HTML

<style>  
 .centered-list {  
 text-align: center;  
 list-style-position: inside; /\* Aligns markers inside the text block \*/  
 }  
</style>  
  
<ul class="centered-list">  
 <li>Item one</li>  
 <li>Item two</li>  
 <li>Item three</li>  
</ul>

### 44. What would happen if there is no text between HTML tags?

If there is no content between the start and end tags of a standard element (like <p></p> or <div></div>), the element will still exist in the DOM, but it will typically have no height and thus will not be visible on the page.5 It will occupy zero space in the document flow. However, it can still be styled with CSS (e.g., given a height, border, or background color) or manipulated with JavaScript.

For void elements like <img>, which don't have closing tags, the concept doesn't apply as they cannot contain text content by definition.18

HTML

<p></p>  
  
<div style="height: 50px; width: 50px; border: 1px solid black;"></div>

### 45. How can you club two or more rows or columns into a single cell in a table?

This is achieved using the rowspan and colspan attributes on a <td> or <th> element, as detailed in question 37.15

* colspan merges a cell horizontally with cells to its right.
* rowspan merges a cell vertically with cells in the rows below it.

When using these attributes, you must omit the corresponding <td> or <th> elements from the subsequent columns or rows to avoid breaking the table structure.

HTML

<table border="1">  
 <tr>  
 <td>Cell 1.1</td>  
 <td colspan="2">Cell 1.2 (spans 2 columns)</td>  
 </tr>  
 <tr>  
 <td rowspan="2">Cell 2.1 (spans 2 rows)</td>  
 <td>Cell 2.2</td>  
 <td>Cell 2.3</td>  
 </tr>  
 <tr>  
 <td>Cell 3.2</td>  
 <td>Cell 3.3</td>  
 </tr>  
</table>

## Section IV: Linking & Embedding Media

This section covers how to connect documents and embed rich media. The evolution of media handling in HTML5 reflects a major shift towards reducing dependency on third-party plugins (like Flash) and prioritizing native browser performance and accessibility. A developer's knowledge of these modern media tags is a proxy for their understanding of the contemporary web development landscape, which emphasizes performance optimization, responsive design, and a standardized, plugin-free user experience.

### 46. How do you create a hyperlink in HTML?

Hyperlinks are the essence of the "hypertext" in HTML and are created using the anchor (<a>) tag.1 The most crucial attribute is

href (hypertext reference), which specifies the destination URL of the link. The content between the opening <a> and closing </a> tags becomes the clickable text or element.

HTML

<a href="https://www.w3.org/">Visit the World Wide Web Consortium</a>  
  
<a href="/about.html">About Us</a>  
  
<a href="#section-two">Jump to Section Two</a>

### 47. What is the purpose of the target attribute in an anchor tag?

The target attribute specifies where to open the linked document.2 The most common values are:

* \_self (Default): Opens the document in the same window or tab where it was clicked.
* \_blank: Opens the document in a new window or tab. This is frequently used for links to external sites to keep the user on the original site.
* \_parent: Opens the document in the parent frame (used with <iframe>s).
* \_top: Opens the document in the full body of the window, breaking out of any frames.

When using target="\_blank", it is a critical security best practice to also include rel="noopener noreferrer".

* noopener prevents the new page from being able to access the original page's window object via window.opener, which prevents a potential security vulnerability.
* noreferrer prevents the browser from sending the Referer HTTP header to the new page, which can be a privacy consideration.

HTML

<a href="https://externalsite.com" target="\_blank" rel="noopener noreferrer">  
 Visit External Site  
</a>

### 48. What is the difference between the <link> tag and the <a> tag?

Although both tags relate to linking, they serve fundamentally different purposes.5

* **<a> (Anchor) Tag:**
  + Creates a clickable hyperlink that users can interact with.
  + It is placed within the <body> of the document.
  + Its primary purpose is user navigation.
* **<link> Tag:**
  + Defines a relationship between the current document and an external resource.
  + It is a void element and is placed within the <head> of the document.
  + It is not interactive. Its most common use is to link to external CSS stylesheets (rel="stylesheet"), but it can also be used for favicons (rel="icon"), preloading resources (rel="preload"), and defining canonical URLs.

In short, <a> is for users, and <link> is for the browser.

### 49. How do you embed an image, and what is the purpose of the alt attribute?

Images are embedded using the <img> tag, which is a void element.1 It requires two essential attributes:

* src (Source): Specifies the path or URL to the image file.
* alt (Alternative Text): Provides a textual description of the image.

The alt attribute is **not optional**; it is a cornerstone of web accessibility.7 Its purposes are:

1. **Screen Readers:** Assistive technologies read the alt text aloud to users with visual impairments, allowing them to understand the image's content and purpose.
2. **Broken Images:** If the image fails to load for any reason (e.g., a wrong path or network issue), the browser will display the alt text in its place.
3. **SEO:** Search engines cannot "see" images, so they rely on alt text to understand the image's content, which helps in indexing the image for image search results.

An alt attribute should be descriptive but concise. If an image is purely decorative and provides no informational content, the alt attribute should still be present but left empty (alt="") so that screen readers know to skip it.

HTML

<img src="golden-retriever.jpg" alt="A golden retriever puppy playing in a field of grass.">  
  
<img src="decorative-border.png" alt="">

### 50. How do you create responsive images?

Responsive images are a crucial performance optimization technique for modern websites that are viewed on a wide range of devices with different screen sizes and resolutions. The goal is to serve the most appropriately sized image file for the user's context, avoiding the download of unnecessarily large images on small screens. There are two primary HTML techniques for this 2:

1. **Resolution Switching with srcset:** The srcset attribute allows you to provide a list of different-sized versions of the same image. The browser can then choose the most suitable one based on the user's screen resolution and viewport size. The sizes attribute can be used alongside srcset to give the browser more context about how large the image will be displayed at different viewport widths.
2. **Art Direction with <picture>:** The <picture> element is used when you need to serve entirely different images for different contexts, not just different sizes of the same image. This is known as "art direction." For example, you might show a wide, landscape-oriented image on a desktop and a cropped, portrait-oriented version of the same subject on a mobile device. The <picture> element contains multiple <source> elements, each with a media query, and a final <img> element as a fallback.

HTML

<img src="cat-small.jpg"  
 srcset="cat-medium.jpg 1000w, cat-large.jpg 2000w"  
 sizes="(max-width: 600px) 480px, 800px"  
 alt="A cute cat.">  
  
<picture>  
 <source media="(min-width: 800px)" srcset="landscape.jpg">  
 <source media="(min-width: 400px)" srcset="portrait.jpg">  
 <img src="square.jpg" alt="A beautiful landscape.">  
</picture>

### 51. What is an image map?

An image map allows you to define clickable areas, or "hotspots," within a single image.5 Each hotspot can be linked to a different URL. This is created using a

<map> element, which is associated with an <img> via the usemap attribute. Inside the <map>, <area> elements define the shape (rect, circle, poly), coordinates, and href for each clickable region.

Image maps are less common today, as CSS and JavaScript often provide more flexible and accessible solutions for creating complex interactive graphics. However, they can still be useful for certain applications, like geographical maps or diagrams.

HTML

<img src="solar-system.png" alt="Planets of the solar system." usemap="#planetmap">  
  
<map name="planetmap">  
 <area shape="rect" coords="0,0,82,126" href="sun.html" alt="Sun">  
 <area shape="circle" coords="90,58,3" href="mercury.html" alt="Mercury">  
 <area shape="circle" coords="124,58,8" href="venus.html" alt="Venus">  
</map>

### 52. How do you embed audio and video in HTML5?

HTML5 introduced the native <audio> and <video> elements, which revolutionized media embedding on the web by eliminating the need for third-party plugins like Flash.4

* **<audio> Element:** Used to embed sound content.
* **<video> Element:** Used to embed video content.

Both elements share common attributes and features:

* src: Specifies the path to the media file.
* controls: A boolean attribute that, if present, displays the browser's default playback controls (play/pause, volume, etc.).
* autoplay: A boolean attribute to make the media play automatically on load (Note: most modern browsers block autoplay with sound unless the user has interacted with the page first).
* loop: A boolean attribute to make the media loop continuously.
* **<source> Element:** To ensure cross-browser compatibility, it is best practice to provide multiple media formats using nested <source> elements. The browser will use the first format it supports.

HTML

<video controls width="640" height="360">  
 <source src="movie.mp4" type="video/mp4">  
 <source src="movie.webm" type="video/webm">  
 Your browser does not support the video tag.  
</video>  
  
<audio controls>  
 <source src="sound.mp3" type="audio/mpeg">  
 <source src="sound.ogg" type="audio/ogg">  
 Your browser does not support the audio element.  
</audio>

### 53. How do you embed another webpage within an HTML document?

Another webpage or an external piece of content (like a map or a YouTube video) can be embedded into an HTML document using the <iframe> (inline frame) element.4

The <iframe> creates a nested browsing context, effectively loading a separate HTML document within a rectangular region of the current page. The src attribute specifies the URL of the page to embed. The width and height attributes can be used to control the size of the frame.

It is important to be aware of security implications when using <iframe>s, such as clickjacking. The sandbox attribute can be used to apply a set of restrictions to the content within the frame.

HTML

<h2>Embedded Content</h2>  
<iframe src="https://www.openstreetmap.org/export/embed.html?bbox=-0.1,51.5,-0.09,51.51"  
 width="600"  
 height="450"  
 style="border:0;"  
 allowfullscreen=""  
 loading="lazy">  
</iframe>

### 54. What is SVG and how do you use it in HTML?

SVG stands for **Scalable Vector Graphics**.5 Unlike raster image formats like JPEG or PNG which are based on pixels, SVG is an XML-based vector image format. This means it describes images using mathematical shapes, paths, and text.

The primary advantages of SVG are 10:

* **Scalability:** SVGs can be scaled to any size without losing quality, making them perfect for responsive design and high-resolution (Retina) displays.
* **Small File Size:** For simple graphics like logos and icons, SVG files are often smaller than their raster counterparts.
* **Manipulability:** Since SVG is XML, it can be styled with CSS and manipulated with JavaScript. Every element within an SVG is part of the DOM.
* **Accessibility:** Text within an SVG remains as text, which is accessible to screen readers and search engines.

There are two main ways to use SVG in HTML:

1. **As an Image:** Link to an .svg file using an <img> tag, just like any other image format. This is the simplest method.
2. **Inline SVG:** Paste the SVG code directly into the HTML document. This allows you to style the individual parts of the SVG with CSS and interact with them using JavaScript.

HTML

<img src="logo.svg" alt="Company Logo">  
  
<svg width="100" height="100" viewBox="0 0 100 100" xmlns="http://www.w3.org/2000/svg">  
 <circle cx="50" cy="50" r="45" stroke="blue" stroke-width="5" fill="none" />  
</svg>

### 55. What is the <marquee> tag?

The <marquee> tag is a non-standard, deprecated HTML element used to create a scrolling area of text or images.5 It was popular in the early days of the web but is now considered obsolete.

It should not be used in modern web development for several reasons:

* It is not part of the W3C HTML standard.
* It presents accessibility issues, as moving text can be difficult for people with cognitive disabilities to read and for screen readers to process.
* It is a presentational element, and its functionality can be better and more accessibly replicated using CSS animations or JavaScript.

### 56. Can you make an image a clickable link?

Yes. To make an image a clickable link, you simply need to wrap the <img> element within an <a> (anchor) tag.19 The

href attribute of the <a> tag will define the link's destination.

HTML

<a href="https://example.com">  
 <img src="logo.png" alt="Visit our homepage">  
</a>

### 57. How do you add a video background to an HTML element?

Creating a video background involves using the HTML5 <video> element and positioning it behind other content with CSS.2

1. **HTML Structure:** Place a <video> element within a container div. The video should have the autoplay, loop, and muted attributes. The muted attribute is often necessary for autoplay to work in modern browsers.
2. **CSS Styling:** Use CSS to position the video absolutely within its container, set its width and height to cover the entire container, and use z-index: -1 to place it behind the other content. The object-fit: cover property is useful for ensuring the video covers the area without being distorted.

HTML

<style>  
 .video-container {  
 position: relative;  
 height: 100vh;  
 width: 100%;  
 overflow: hidden;  
 color: white;  
 text-align: center;  
 }  
 #background-video {  
 position: absolute;  
 top: 50%;  
 left: 50%;  
 min-width: 100%;  
 min-height: 100%;  
 width: auto;  
 height: auto;  
 z-index: -1;  
 transform: translateX(-50%) translateY(-50%);  
 background-size: cover;  
 }  
 .content {  
 position: relative;  
 z-index: 1;  
 padding-top: 20vh;  
 }  
</style>  
  
<div class="video-container">  
 <video autoplay muted loop id="background-video">  
 <source src="background.mp4" type="video/mp4">  
 </video>  
 <div class="content">  
 <h1>Welcome to Our Site</h1>  
 <p>Experience the motion.</p>  
 </div>  
</div>

### 58. What is the purpose of the download attribute in an anchor tag?

The download attribute, when added to an <a> tag, instructs the browser to download the linked URL instead of navigating to it.

When a user clicks on the link, a "Save As" dialog will appear, prompting them to save the file locally. The value of the download attribute can be used to suggest a filename for the downloaded file. If the value is empty, the browser will use the original filename. This is particularly useful for forcing the download of file types that the browser would normally try to display, such as images or PDF files.

HTML

<a href="/images/report.pdf" download>Download the Report</a>  
  
<a href="/images/report.pdf" download="annual-report.pdf">Download the Annual Report</a>

### 59. What is the difference between an absolute and a relative URL?

The href attribute in an <a> tag or the src attribute in an <img> tag can take either an absolute or a relative URL.

* **Absolute URL:** This is a full web address that includes the protocol (http:// or https://), domain name, and path. It points to a specific location on the web, regardless of the location of the current page. Absolute URLs are used for linking to external websites.
* **Relative URL:** This is a partial address that is relative to the current page's URL. It does not include the protocol or domain name. Relative URLs are used for linking to other pages or resources within the same website. This makes the site more portable, as the links will not break if the domain name changes.

HTML

<a href="https://www.mozilla.org/">Mozilla Homepage</a>  
  
<a href="about.html">About Us</a>  
  
<img src="images/logo.png" alt="Logo">

### 60. How do you add a favicon to your website?

A favicon (favorite icon) is a small icon displayed in the browser tab, bookmarks bar, and other places. It is added to a website using a <link> tag in the <head> section of the HTML document.19

The rel attribute should be set to icon, and the href attribute should point to the location of the icon file. While historically .ico was the standard format, modern browsers support other formats like PNG, GIF, and SVG. It is good practice to also specify the type of the image file.

HTML

<head>  
 <title>My Website</title>  
 <link rel="icon" href="/favicon.ico" type="image/x-icon">  
 <link rel="icon" href="/favicon.png" type="image/png">  
</head>

## Section V: Interactive Elements - Forms & User Input

This section provides an in-depth look at HTML forms, the primary mechanism for collecting user input. HTML5 forms represent a significant stride in shifting validation logic from the server to the client, improving user experience and reducing unnecessary server load. This demonstrates a core principle of modern web development: progressive enhancement. The browser provides a baseline level of functionality and validation, creating a faster, more responsive, and user-friendly interface, while server-side validation remains essential for security. An interviewer asking about these features is testing a candidate's understanding of modern UX principles.

### 61. How do you create a form in HTML?

An HTML form is created using the <form> element, which acts as a container for various input controls.4 These controls allow users to enter data, which can then be submitted to a server for processing.

The <form> element has two essential attributes:

* action: Specifies the URL of the server-side script that will process the form data (e.g., /submit-form.php).8
* method: Specifies the HTTP method to be used when submitting the form data. The two most common methods are GET and POST.8

Inside the <form> tags, you place form controls like <input>, <textarea>, <select>, and <button>.

HTML

<form action="/register" method="post">  
 <label for="username">Username:</label>  
 <input type="text" id="username" name="username">  
  
 <button type="submit">Submit</button>  
</form>

### 62. What is the difference between the GET and POST methods?

GET and POST are the two most common HTTP methods used in form submissions, and they handle data very differently.10

* **GET:**
  + Appends the form data to the URL as a query string (e.g., .../search?query=html).
  + Data is visible in the URL, browser history, and server logs, making it **insecure** for sensitive information like passwords.
  + There is a limit to the length of the URL, so GET can only handle small amounts of data.
  + GET requests are idempotent, meaning multiple identical requests should have the same effect as a single one. They are suitable for retrieving data, such as search queries or filtering results.
* **POST:**
  + Sends the form data in the body of the HTTP request, separate from the URL.
  + Data is not visible in the URL, making it more **secure** for submitting sensitive information.
  + There is no size limit on the amount of data that can be sent.
  + POST requests are not idempotent; submitting the same form multiple times will typically create multiple new resources. They are used for actions that change the state on the server, such as creating a new user, submitting a comment, or uploading a file.

### 63. What are the different types of <input> elements?

The <input> element is the most versatile form control, and its behavior is determined by its type attribute.15 Common input types include:

* text: A single-line text field.
* password: A text field where the characters are obscured.
* checkbox: A checkbox that can be toggled on or off.
* radio: A radio button, typically used in a group where only one can be selected at a time.
* submit: A button that submits the form.
* button: A generic clickable button, usually controlled by JavaScript.
* file: A control for uploading files.
* hidden: An input field that is not visible to the user but whose value is submitted with the form.
* reset: A button that resets all form controls to their initial values.

### 64. What are some of the new input types introduced in HTML5?

HTML5 added several new input types that provide better user experience, especially on mobile devices, and enable built-in browser validation.3

* email: For email addresses. The browser may provide validation to ensure the input is a valid email format.
* url: For web addresses.
* tel: For telephone numbers. Mobile browsers may display a numeric keypad.
* number: For numerical values. Provides spinner controls and can be constrained with min, max, and step attributes.
* date: Provides a date picker interface.
* color: Provides a color picker interface.
* range: A slider control for selecting a value within a range.
* search: A text field specifically for search terms.

Using these types allows the browser to provide a more appropriate user interface and perform client-side validation without requiring JavaScript.

HTML

<form action="/submit" method="post">  
 <label for="email-addr">Email:</label>  
 <input type="email" id="email-addr" name="email" required>  
  
 <label for="quantity">Quantity (1-5):</label>  
 <input type="number" id="quantity" name="quantity" min="1" max="5">  
  
 <label for="appt-date">Appointment Date:</label>  
 <input type="date" id="appt-date" name="appointment\_date">  
  
 <button type="submit">Submit</button>  
</form>

### 65. What is the purpose of the <label> element?

The <label> element is essential for form accessibility. It provides a human-readable caption for a form control.15

A <label> is programmatically associated with its corresponding <input> (or other form control) in one of two ways:

1. **Wrapping:** The <input> is placed inside the <label> element.
2. **Using for and id:** The <label> has a for attribute whose value is the same as the id of the <input> element. This is the more robust and common method.

This association provides two key benefits:

* **Accessibility:** Screen readers will read the label text when the user focuses on the input field, providing context for what information should be entered.
* **Usability:** On all devices, a user can click on the label text to focus on or activate the corresponding input field. This is especially helpful for small targets like checkboxes and radio buttons.

A form without properly associated labels is not accessible.

HTML

<label for="user-name">Name:</label>  
<input type="text" id="user-name" name="name">  
  
<label>  
 <input type="checkbox" name="subscribe">  
 Subscribe to newsletter  
</label>

### 66. What are <textarea>, <select>, and <option> elements used for?

These are other essential form controls for different types of user input.2

* **<textarea>:** Defines a multi-line text input control. Unlike the <input type="text">, it can hold an unlimited number of characters, and the text can wrap. The size of the text area can be specified by the rows and cols attributes, or more flexibly with CSS.
* **<select>:** Creates a dropdown list. It acts as a container for a list of <option> elements.
* **<option>:** Defines an option in a <select> list. The text between the <option> tags is what the user sees, and the value attribute specifies the data that is sent to the server when that option is selected.

HTML

<label for="comments">Comments:</label>  
<textarea id="comments" name="comments" rows="4" cols="50"></textarea>  
  
<label for="country">Country:</label>  
<select id="country" name="country">  
 <option value="us">United States</option>  
 <option value="ca">Canada</option>  
 <option value="mx">Mexico</option>  
</select>

### 67. How do you group related form controls?

Related form controls can be grouped together using the <fieldset> element. The <legend> element provides a caption or title for the group.7

Grouping controls this way is beneficial for both visual organization and accessibility. Visually, browsers typically draw a box around the <fieldset>. For screen reader users, the <legend> text is announced when the user focuses on the first control within the group, providing context for the entire set of related inputs (e.g., "Shipping Address").

HTML

<fieldset>  
 <legend>Contact Information</legend>  
  
 <label for="name">Name:</label>  
 <input type="text" id="name" name="name">  
  
 <label for="email">Email:</label>  
 <input type="email" id="email" name="email">  
</fieldset>

### 68. What are some important HTML5 form validation attributes?

HTML5 introduced several attributes that allow for declarative, client-side validation directly in the markup, reducing the need for JavaScript.2

* required: A boolean attribute that specifies that an input field must be filled out before submitting the form.
* placeholder: Provides a short hint or example text that is displayed in the input field before the user enters a value.
* pattern: Specifies a regular expression that the input field's value is checked against. This is powerful for validating complex formats like phone numbers or postal codes.
* min and max: Specify the minimum and maximum values for an input of type number, range, or date.
* step: Specifies the legal number intervals for an input of type number or range.
* readonly: A boolean attribute that specifies that an input field is read-only (cannot be changed) but its value is still submitted with the form.
* disabled: A boolean attribute that specifies that an input field should be disabled. A disabled input is unusable and un-clickable, and its value is not submitted with the form.

### 69. What is the purpose of the <datalist> element?

The <datalist> element provides an "autocomplete" feature for <input> elements.3 It contains a set of

<option> elements that represent pre-defined suggestions for the user.

To associate a <datalist> with an <input>, the id of the <datalist> must match the list attribute of the <input>. The user will see a dropdown list of the suggestions as they type, but they are still free to enter a value that is not on the list.

HTML

<label for="browser-choice">Choose your browser from the list:</label>  
<input list="browsers" id="browser-choice" name="browser">  
  
<datalist id="browsers">  
 <option value="Chrome">  
 <option value="Firefox">  
 <option value="Safari">  
 <option value="Edge">  
 <option value="Opera">  
</datalist>

### 70. What does enctype="multipart/form-data" mean?

The enctype attribute of a <form> element specifies how the form data should be encoded when submitting it to the server. The default value is application/x-www-form-urlencoded.

However, when a form includes an <input type="file"> for file uploads, the enctype must be set to multipart/form-data.13 This encoding method does not encode the data and instead sends the form data in multiple parts, with one part for each form control and a separate part for the file content itself. This is necessary to handle the binary data of the uploaded file.

HTML

<form action="/upload" method="post" enctype="multipart/form-data">  
 <label for="profile-pic">Upload a profile picture:</label>  
 <input type="file" id="profile-pic" name="picture">  
 <button type="submit">Upload</button>  
</form>

### 71. What is the difference between readonly and disabled attributes?

While both attributes prevent a user from modifying an input field, they have key differences in behavior.2

* **disabled:**
  + The input is completely unusable and un-clickable.
  + The input's value is **not** submitted with the form.
  + The input cannot receive focus.
  + It is typically used for fields that are conditionally unavailable.
* **readonly:**
  + The input cannot be modified by the user, but they can still focus on it and select/copy its text.
  + The input's value **is** submitted with the form.
  + It is typically used when you want to display a value that the user should see but not be able to change, like a pre-filled username or a calculated total.

### 72. How do you create a radio button group?

To create a group of radio buttons where only one option can be selected, all the <input type="radio"> elements in the group must share the same name attribute. The value attribute for each radio button should be unique, as this is the value that will be submitted to the server if that option is selected.

HTML

<fieldset>  
 <legend>Choose your favorite language:</legend>  
 <input type="radio" id="html" name="language" value="html">  
 <label for="html">HTML</label><br>  
 <input type="radio" id="css" name="language" value="css">  
 <label for="css">CSS</label><br>  
 <input type="radio" id="js" name="language" value="javascript">  
 <label for="js">JavaScript</label>  
</fieldset>

### 73. How do you create a checkbox?

A checkbox is created using <input type="checkbox">. Unlike radio buttons, multiple checkboxes in a group can be selected simultaneously. If you want to group them logically for server-side processing, you can give them the same name attribute, often with square brackets (``) to indicate an array of values to the server-side language.

HTML

<fieldset>  
 <legend>Select your toppings:</legend>  
 <input type="checkbox" id="topping1" name="toppings" value="pepperoni">  
 <label for="topping1">Pepperoni</label><br>  
 <input type="checkbox" id="topping2" name="toppings" value="mushrooms">  
 <label for="topping2">Mushrooms</label><br>  
 <input type="checkbox" id="topping3" name="toppings" value="onions">  
 <label for="topping3">Onions</label>  
</fieldset>

### 74. What is the purpose of the formnovalidate attribute?

The formnovalidate is a boolean attribute that can be applied to a submit button (<input type="submit"> or <button type="submit">). When present, it specifies that the form should not be validated when submitted.2

This overrides the form's default validation behavior and any validation attributes on the input fields (like required or pattern). It is useful for implementing a "Save as Draft" feature, where you want to allow the user to save their progress without having to fill out the entire form correctly.

HTML

<form action="/submit" method="post">  
 <label for="title">Title (required):</label>  
 <input type="text" id="title" name="title" required>  
  
 <button type="submit">Submit</button>  
 <button type="submit" formnovalidate>Save as Draft</button>  
</form>

### 75. What is the purpose of the hidden input type?

An <input type="hidden"> is a form control that is not visible to the user on the webpage but whose value is still sent to the server when the form is submitted.

It is used to store data that needs to be sent with the form but that the user does not need to see or interact with. Common use cases include:

* Storing a session ID or user token for security.
* Passing a unique ID for the item being edited in an "edit" form.
* Tracking the source of a form submission for analytics.

HTML

<form action="/update-post" method="post">  
 <input type="hidden" name="post\_id" value="12345">  
  
 <label for="post-content">Edit your post:</label>  
 <textarea id="post-content" name="content">Original content here...</textarea>  
  
 <button type="submit">Update Post</button>  
</form>

## Section VI: HTML5 APIs & Advanced Features

This section explores the powerful client-side capabilities introduced with HTML5, which transform the browser from a simple document viewer into a rich application platform. The HTML5 APIs collectively represent the "application-ization" of the web, providing native browser capabilities that were previously only achievable through complex JavaScript libraries or server-side technologies. This shift empowers developers to build more powerful, responsive, and feature-rich applications that run directly in the browser, blurring the lines between traditional websites and native applications. Understanding these APIs is essential for anyone working on Single-Page Applications (SPAs) or Progressive Web Apps (PWAs).

### 76. What is Web Storage, and what is the difference between localStorage and sessionStorage?

Web Storage is an HTML5 API that provides a way for web pages to store key/value data locally within the user's browser, offering a more modern and powerful alternative to cookies.8 There are two types of Web Storage:

1. **sessionStorage:**
   * Stores data for only one session.
   * The data is cleared as soon as the user closes the browser tab or window.
   * The data is specific to that tab; data stored in one tab is not accessible in another tab, even if it's from the same origin.
   * Useful for storing temporary data related to a single user journey, like information in a multi-step form.
2. **localStorage:**
   * Stores data with no expiration date.
   * The data persists even after the browser tab and window are closed and will be available the next time the user visits the site.
   * Data is shared across all tabs and windows from the same origin.
   * Useful for storing user preferences, application state, or data that should be remembered between visits.

The following table provides a detailed comparison of client-side storage mechanisms.13

| Feature | cookie | sessionStorage | localStorage |
| --- | --- | --- | --- |
| **Initiator** | Client or Server | Client | Client |
| **Expiry** | Manually set | On tab close | Never expires |
| **Persistence** | Yes (if expiry is set) | No | Yes |
| **Sent with HTTP Requests** | Yes, automatically | No | No |
| **Capacity** | ~4 KB | ~5-10 MB | ~5-10 MB |
| **Accessibility** | Any window (same origin) | Same tab only | Any window (same origin) |

### 77. What are data-\* attributes and when should they be used?

data-\* attributes are a standardized way to store custom, private data directly on an HTML element.1 The attribute name must start with

data-, followed by at least one character.

Their primary purpose is to store extra information that is relevant to the page's scripts but is not intended for display. This data can then be easily accessed and manipulated using JavaScript via the dataset property of the element object.

This is particularly useful for associating metadata with elements without resorting to non-standard attributes or misusing class or id. For example, you could store a product's ID or a user's role directly on an element.

HTML

<div id="user-card" data-user-id="987" data-role="editor">  
 John Doe  
</div>  
  
<script>  
 const card = document.getElementById('user-card');  
 const userId = card.dataset.userId; // "987"  
 const userRole = card.dataset.role; // "editor"  
  
 console.log(`User ${userId} has the role of ${userRole}.`);  
</script>

### 78. What is the Geolocation API?

The HTML5 Geolocation API provides a way for web applications to retrieve the geographical location (latitude and longitude) of a user's device.10

For privacy reasons, the browser will always ask for the user's permission before sharing their location with a website. The API is accessed through the navigator.geolocation object. The primary method is getCurrentPosition(), which takes a success callback function, an optional error callback function, and optional position options.

This API is the foundation for location-aware web applications, such as mapping services, local search, and check-in features.

HTML

<button id="find-me">Show my location</button>  
<p id="location-info"></p>  
  
<script>  
 const findMeButton = document.getElementById('find-me');  
 const locationInfo = document.getElementById('location-info');  
  
 findMeButton.addEventListener('click', () => {  
 if (!navigator.geolocation) {  
 locationInfo.textContent = 'Geolocation is not supported by your browser.';  
 } else {  
 locationInfo.textContent = 'Locating…';  
 navigator.geolocation.getCurrentPosition(success, error);  
 }  
 });  
  
 function success(position) {  
 const latitude = position.coords.latitude;  
 const longitude = position.coords.longitude;  
 locationInfo.textContent = `Latitude: ${latitude}°, Longitude: ${longitude}°`;  
 }  
  
 function error() {  
 locationInfo.textContent = 'Unable to retrieve your location.';  
 }  
</script>

### 79. What is the difference between Canvas and SVG?

Canvas and SVG are the two primary technologies for creating graphics on the web, but they operate on fundamentally different principles.3

* **<canvas>:**
  + **Pixel-based (Bitmap):** Provides a drawing surface where you can draw graphics pixel by pixel using JavaScript.
  + **Immediate Mode:** Once a shape is drawn on the canvas, the browser forgets about it. If you want to move it, you must clear the canvas and redraw the entire scene.
  + **Performance:** Generally has better performance for rendering a large number of objects or complex, fast-moving animations.
  + **Use Cases:** Ideal for dynamic graphics like games, data visualizations, and image manipulation. It is not accessible as the content is not part of the DOM.
* **<svg> (Scalable Vector Graphics):**
  + **Vector-based (XML):** Defines graphics as a set of shapes, paths, and text in an XML format.
  + **Retained Mode:** Each shape drawn in an SVG is an object in the DOM. You can attach event listeners to individual shapes and manipulate them with CSS and JavaScript.
  + **Scalability:** Being vector-based, SVGs scale to any size without losing quality.
  + **Use Cases:** Ideal for static or interactive graphics that need to be scalable and accessible, such as logos, icons, and charts.

### 80. What are Web Workers?

A Web Worker is a JavaScript script that runs in the background on a separate thread from the main browser UI thread.10

The main purpose of Web Workers is to allow long-running or computationally intensive tasks to be performed without freezing or slowing down the user interface. The main thread remains responsive to user interactions (like clicks and scrolling) while the worker thread performs the heavy lifting in the background.

Communication between the main thread and the worker thread is done via a system of messages. They cannot directly access or manipulate the DOM.

JavaScript

// main.js  
const myWorker = new Worker('worker.js');  
  
myWorker.postMessage(); // Send data to the worker  
  
myWorker.onmessage = function(e) {  
 console.log('Result from worker:', e.data); // Receives data back from the worker  
}  
  
// worker.js  
onmessage = function(e) {  
 const result = e.data \* e.data;  
 postMessage(result); // Send the result back to the main thread  
}

### 81. What are WebSockets?

The WebSocket API is a technology that enables opening a two-way, interactive communication session between a user's browser and a server.10 With this API, you can send messages to a server and receive event-driven responses without having to poll the server for a reply.

Unlike traditional HTTP, which follows a request-response model, WebSockets provide a persistent, full-duplex connection. This means both the client and the server can send data to each other at any time, making it ideal for real-time applications such as:

* Live chat applications
* Multiplayer online games
* Real-time data feeds (e.g., stock tickers, sports scores)
* Collaborative editing tools

### 82. What is the Application Cache and Manifest file?

The Application Cache (AppCache) was an HTML5 feature designed to allow web applications to be used offline.3 It worked by specifying a

**manifest file** (a text file with a .appcache extension) in the <html> tag's manifest attribute. This file listed the resources (HTML, CSS, JS, images) that the browser should cache locally.

However, the Application Cache was found to have significant design flaws and was difficult to work with. It has since been **deprecated** and removed from web standards. The modern and recommended technology for creating offline-capable web applications is **Service Workers**, which are part of the Progressive Web App (PWA) technology suite and offer much more power and flexibility.

### 83. What is MathML?

MathML stands for **Mathematical Markup Language**. It is an XML-based language used to describe mathematical notation and structure for display on web pages.10

With HTML5, MathML can be embedded directly into HTML documents, allowing browsers to render complex mathematical formulas and equations semantically. This is a significant improvement over using images to display equations, as MathML is accessible to screen readers and can be styled with CSS.

HTML

<p>The quadratic formula is:</p>  
<math>  
 <mi>x</mi>  
 <mo>=</mo>  
 <mfrac>  
 <mrow>  
 <mo form="prefix">−</mo>  
 <mi>b</mi>  
 <mo>±</mo>  
 <msqrt>  
 <msup>  
 <mi>b</mi>  
 <mn>2</mn>  
 </msup>  
 <mo>−</mo>  
 <mn>4</mn>  
 <mi>a</mi>  
 <mi>c</mi>  
 </msqrt>  
 </mrow>  
 <mrow>  
 <mn>2</mn>  
 <mi>a</mi>  
 </mrow>  
 </mfrac>  
</math>

### 84. What is the purpose of the time element and its datetime attribute?

The <time> element is a semantic tag used to represent a specific period in time.3 This can be a date, a time, or a duration.

While the content between the <time> tags is human-readable, the element's primary power comes from the datetime attribute. This attribute provides the same information in a machine-readable format, which is useful for search engines, calendars, and other automated services to parse and use the data.

The datetime attribute must be in a valid format, such as YYYY-MM-DD for a date or YYYY-MM-DDThh:mm:ss for a date and time.4

HTML

<p>The conference will be held on <time datetime="2025-10-26">October 26th, 2025</time>.</p>  
<p>The event starts at <time datetime="2025-10-26T19:00-05:00">7:00 PM Central Time</time>.</p>

### 85. How do you handle events in HTML?

Events are actions or occurrences that happen in the browser, such as a user clicking a button, a page finishing loading, or an input field being changed. HTML allows you to respond to these events by using event handler attributes that execute a piece of JavaScript code when the event occurs.15

Event handler attributes are named with an "on" prefix, such as onclick, onmouseover, onkeydown, and onload.

While this method works, the modern and more flexible approach is to use JavaScript's addEventListener() method to attach event listeners to DOM elements. This separates the HTML structure from the JavaScript behavior, which is a core principle of modern web development.

HTML

<button onclick="alert('You clicked the button!')">Click Me</button>  
  
<button id="myButton">Click Me Too</button>  
<script>  
 const button = document.getElementById('myButton');  
 button.addEventListener('click', () => {  
 alert('You clicked the second button!');  
 });  
</script>

## Section VII: Web Accessibility (A11y)

This section is dedicated to the crucial topic of web accessibility (often abbreviated as A11y), covering the principles and techniques required to build inclusive websites that are usable by people with disabilities. A developer who understands accessibility demonstrates a commitment to professional, ethical, and user-centric development.

### 86. What is Web Accessibility (A11y)?

Web Accessibility is the inclusive practice of ensuring there are no barriers that prevent interaction with, or access to, websites on the World Wide Web by people with physical disabilities, situational disabilities, and socio-economic restrictions on bandwidth and speed.7

When sites are correctly designed, developed, and edited, all users have equal access to information and functionality. This means creating content that can be navigated and understood by people who may be using assistive technologies like screen readers, screen magnifiers, or voice recognition software, as well as those who can only use a keyboard for navigation.

### 87. How does semantic HTML improve accessibility?

Semantic HTML is the foundation of an accessible website.11 As discussed previously, assistive technologies like screen readers do not see the visual layout of a page; they interpret the DOM. Semantic elements provide a machine-readable structure that these technologies can use to convey meaning and provide navigational shortcuts.

* **Headings (<h1>-<h6>)** create a document outline.
* **Landmark elements (<nav>, <main>, <header>, <footer>)** define the major regions of a page, allowing users to jump directly to the content they need.
* **List elements (<ul>, <ol>)** inform the user that they are about to hear a list of related items and how many items are in the list.
* **Form elements (<label>, <fieldset>)** create clear associations between text and input controls.

Using a <div> for everything forces a screen reader user to navigate the page linearly, without any context or shortcuts, making the experience frustrating and inefficient.

### 88. What is the role of the alt attribute for accessibility?

The alt attribute on an <img> tag provides a text alternative for the image.2 This is arguably one of the most critical and fundamental aspects of web accessibility.

For a user with a visual impairment using a screen reader, the alt text is read aloud, describing the content and function of the image. Without it, the user has no way of knowing what information the image conveys. A well-written alt attribute provides an equivalent experience for non-visual users. If an image is purely decorative, it should have an empty alt attribute (alt="") to signal to screen readers that it can be safely ignored.

### 89. What is ARIA and when should you use it?

ARIA stands for **Accessible Rich Internet Applications**. It is a set of special accessibility attributes that can be added to any HTML markup to help make web content and web applications more accessible to people with disabilities.7

ARIA is particularly useful for making dynamic content and advanced user interface controls (widgets) accessible, especially those developed with JavaScript for which there is no native HTML equivalent (e.g., a custom slider, a tabbed interface, or an autocomplete dropdown).

ARIA provides three main features:

1. **Roles:** Define what an element is or does (e.g., role="navigation", role="button", role="alert").
2. **Properties:** Define characteristics of elements (e.g., aria-required, aria-labelledby).
3. **States:** Define the current conditions of elements (e.g., aria-disabled, aria-expanded).

### 90. What is the first rule of ARIA?

The first and most important rule of ARIA is: **If you can use a native HTML element or attribute with the semantics and behavior you require already built in, use it instead of repurposing an element and adding an ARIA role, state, or property to make it accessible.**

This principle reveals a crucial hierarchy of best practices. It is always better to use a native <button> element than to create a <div> and add role="button" and the necessary keyboard event listeners with JavaScript. The native element provides all the necessary semantics, keyboard focus behavior, and accessibility features for free, with less code and less room for error. ARIA should be seen as a patch for making non-semantic or custom-built components accessible, not as a replacement for proper semantic HTML. Overusing ARIA where native elements suffice is often a sign of a less experienced developer or a codebase that needs refactoring.

HTML

<div role="button" tabindex="0" onclick="doSomething()">Click me</div>  
  
<button onclick="doSomething()">Click me</button>

### 91. How do you ensure a website is keyboard navigable?

Keyboard navigability is essential for users with motor disabilities who cannot use a mouse, as well as for power users and screen reader users.11 The key principles are:

1. **Use Native Interactive Elements:** Use <a>, <button>, <input>, and other native interactive elements wherever possible. These elements are keyboard-focusable by default.
2. **Logical Tab Order:** Ensure the order in which elements receive focus when pressing the Tab key is logical and follows the visual flow of the page. This is usually achieved naturally by having a logical source order in the HTML.
3. **Visible Focus Indicator:** Never remove the browser's default focus outline (e.g., with outline: none;) without providing a clear and high-contrast alternative. Users need to be able to see which element currently has focus.
4. **Custom Controls:** If you create custom interactive components (e.g., from <div>s), you must make them focusable by adding tabindex="0" and add the necessary JavaScript event listeners to handle keyboard events (like Enter and Space for activation).

### 92. What is the purpose of the scope attribute in an HTML table?

As detailed in question 38, the scope attribute is a critical accessibility feature for tables.2 It is used on

<th> elements to explicitly define whether a header cell is a header for a column (scope="col") or a row (scope="row").

This programmatic association allows screen readers to correctly announce the relationship between a data cell and its corresponding headers, making complex data tables understandable to users with visual impairments.

## Section VIII: Performance, Optimization, & Best Practices

This concluding section addresses professional-level concerns that go beyond basic markup, focusing on how to write HTML that is performant, maintainable, and valid. These topics often separate junior candidates from more senior ones, as they demonstrate an understanding of the broader context in which HTML operates.

### 93. What are some ways to optimize an HTML page's load time?

Optimizing page load time is crucial for user experience and SEO. Several strategies involving HTML and related assets can be employed 7:

1. **Minimize HTTP Requests:** Combine multiple CSS and JS files into single files to reduce the number of requests the browser has to make.
2. **Minify HTML, CSS, and JavaScript:** Remove all unnecessary characters (whitespace, comments, etc.) from code files to reduce their size.
3. **Optimize Images:** Compress images to reduce their file size and use responsive image techniques (srcset, <picture>) to serve appropriately sized images.
4. **Use a Content Delivery Network (CDN):** Host static assets (images, CSS, JS) on a CDN to serve them from a location geographically closer to the user, reducing latency.
5. **Defer or Asynchronously Load JavaScript:** Use the defer and async attributes on <script> tags to prevent JavaScript from blocking the rendering of the page.
6. **Enable Browser Caching:** Configure server headers to tell the browser to cache static assets so they don't need to be re-downloaded on subsequent visits.
7. **Implement Lazy Loading:** Use the loading="lazy" attribute on images and iframes to defer the loading of off-screen content until the user scrolls near it.

### 94. What is the difference between <script>, <script async>, and <script defer>?

The async and defer attributes on the <script> tag provide powerful control over how and when external JavaScript files are loaded and executed, with significant performance implications. Their correct usage depends entirely on the script's dependencies.3

* **<script> (Default):** The browser pauses HTML parsing, fetches the script, executes it, and then resumes parsing. This is **render-blocking**.
* **<script async>:** The script is fetched in parallel with HTML parsing. As soon as the script is downloaded, HTML parsing is paused, and the script is executed. After execution, parsing resumes. Scripts may execute in any order, as soon as they are ready. This is useful for independent scripts, like analytics, that don't depend on the DOM or other scripts.
* **<script defer>:** The script is fetched in parallel with HTML parsing. The script is only executed *after* the HTML parsing is complete, just before the DOMContentLoaded event. defer scripts are guaranteed to execute in the order they appear in the HTML. This is the best choice for scripts that need to interact with the DOM.

The following table summarizes the behavior.13

| Attribute | HTML Parsing | Script Fetching | Script Execution |
| --- | --- | --- | --- |
| **<script> (Default)** | Paused during fetch & execution | Blocking | Immediately after fetch |
| **<script async>** | Continues during fetch; paused during execution | Parallel (non-blocking) | As soon as downloaded; order not guaranteed |
| **<script defer>** | Continues during fetch | Parallel (non-blocking) | After HTML parsing is complete; order guaranteed |

Choosing the wrong attribute can lead to race conditions and execution errors. An interviewer asking this question is testing a candidate's deep understanding of the browser's rendering pipeline and their ability to write non-blocking, performant code.

### 95. What is lazy loading and how is it implemented in HTML?

Lazy loading is a performance optimization technique where the loading of non-critical resources (typically images or iframes) is deferred until they are needed.7 Instead of loading all images on a page at once, lazy loading waits to load an image until the user scrolls and it is about to enter the viewport. This significantly improves initial page load time, saves bandwidth, and reduces memory usage.

The simplest way to implement lazy loading in modern browsers is to use the loading="lazy" attribute directly on <img> and <iframe> tags. This is a native browser feature that requires no JavaScript.

For older browsers that do not support this attribute, lazy loading can be implemented using JavaScript with the Intersection Observer API, which provides an efficient way to detect when an element enters the viewport.

HTML

<img src="heavy-image.jpg" loading="lazy" alt="An image that will be loaded on demand.">  
  
<iframe src="https://example.com" loading="lazy" title="An embedded page."></iframe>

### 96. Why is it important to validate your HTML code?

Validating HTML code means checking it against the official standards and specifications set by the World Wide Web Consortium (W3C).6 This is an important quality assurance step for several reasons:

1. **Cross-Browser Compatibility:** Valid code is more likely to be rendered consistently across different browsers and devices, as it adheres to the standards they are all built to support.
2. **Debugging:** Validation tools can catch syntax errors, such as unclosed tags or improper nesting, which can be difficult to spot manually and can cause strange rendering issues.
3. **Future-Proofing:** Code that conforms to standards is more likely to work correctly with future versions of browsers.
4. **Maintainability:** Clean, valid code is easier for other developers to read, understand, and maintain.

The most common tool for this is the official W3C Markup Validation Service.

### 97. What is progressive rendering?

Progressive rendering is a broad term for techniques used to render content for display as quickly as possible to improve perceived load time, rather than waiting for the entire page to load before rendering anything.14 The goal is to get meaningful content in front of the user as fast as possible.

HTML-related strategies that contribute to progressive rendering include:

* **Placing CSS in the <head>:** This allows the browser to start building the render tree early.
* **Placing JavaScript at the end of the <body> (or using defer):** This prevents scripts from blocking the initial rendering of the page content.
* **Optimizing the Critical Rendering Path:** Ensuring that the HTML, CSS, and JavaScript needed to render the "above-the-fold" content (the part of the page visible without scrolling) are loaded and processed first.

### 98. What is the difference between display: none and visibility: hidden?

These are two CSS properties used to hide elements, but they do so in fundamentally different ways, which is a common interview question that bridges HTML and CSS knowledge.5

* **visibility: hidden:**
  + The element is made invisible, but it still **occupies its space** in the document layout. It's as if the element is transparent.
  + Child elements can be made visible again by setting visibility: visible on them.
* **display: none:**
  + The element is completely **removed** from the document flow. It takes up no space, and the layout reflows as if the element never existed.
  + The element and all of its descendants are hidden and cannot be made visible by changing the visibility property on a child.

The HTML hidden attribute is generally equivalent to applying display: none; via CSS.

### 99. What are some best practices for writing clean HTML?

Writing clean, professional HTML is about more than just getting the syntax right. It involves adhering to a set of conventions and principles that make the code maintainable, accessible, and performant.7

* **Use Semantic HTML:** Choose elements that accurately describe their content.
* **Maintain a Logical Structure:** Use a proper heading hierarchy and ensure the source order makes sense.
* **Validate Your Code:** Regularly check your HTML for errors.
* **Separate Content from Presentation:** Use HTML for structure and CSS for styling. Avoid inline styles.
* **Write Readable Code:** Use consistent indentation and formatting to make the code easy to read.
* **Prioritize Accessibility:** Always include alt text for images, use labels for forms, and ensure keyboard navigability.
* **Comment Your Code:** Add comments to explain complex or non-obvious parts of the markup.
* **Optimize for Performance:** Follow best practices for image optimization and script loading.

### 100. What is the difference between HTML4 and HTML5?

HTML5 is a major evolution of the HTML standard, introducing a vast number of new features, semantic elements, and APIs that are foundational to the modern web.3 Key differences include:

* **Semantic Elements:** HTML5 introduced landmark elements like <header>, <footer>, <article>, and <section>, providing a richer, more descriptive structure.
* **Multimedia Support:** Native <audio> and <video> elements were added, eliminating the reliance on third-party plugins like Flash.
* **Graphics:** The <canvas> element for pixel-based drawing and native support for <svg> were introduced.
* **Forms:** New input types (email, date, range, etc.) and validation attributes (required, pattern) were added to enhance forms.
* **APIs:** A suite of powerful JavaScript APIs were introduced, including Web Storage, Geolocation, Web Workers, and WebSockets, enabling more complex, application-like experiences in the browser.
* **Simpler Syntax:** The <!DOCTYPE> declaration was simplified, and other syntax rules were made more consistent and developer-friendly.

In essence, while HTML4 was primarily for structuring documents, HTML5 transformed the browser into a full-fledged application platform.

## Conclusion

The landscape of HTML has matured into a sophisticated ecosystem where structure is inextricably linked to meaning, accessibility, and performance. The questions and detailed explanations provided in this guide illustrate a clear trajectory in web development: a move away from purely presentational markup toward a holistic approach that considers the diverse needs of users, search engines, and developers.

A mastery of modern HTML is demonstrated not by the ability to recall an exhaustive list of tags, but by the ability to articulate the principles behind their use. This includes understanding the profound impact of the <!DOCTYPE> declaration on browser rendering, the critical role of semantic markup as an API for assistive technologies, the performance implications of resource loading strategies, and the ethical and practical necessity of building accessible web experiences. As the web continues to evolve, these foundational principles will remain the bedrock of high-quality, professional web development. Developers who internalize this deeper understanding will be best positioned to build the robust, inclusive, and performant web applications of the future.

#### Works cited

1. Devinterview-io/html5-interview-questions: HTML5 ... - GitHub, accessed on August 25, 2025, <https://github.com/Devinterview-io/html5-interview-questions>
2. Top HTML5 and HTML Interview Questions and Answers (2024) - AlmaBetter, accessed on August 25, 2025, <https://www.almabetter.com/bytes/articles/html-interview-questions>
3. 50 Html Interview Question And Answers - GitHub, accessed on August 25, 2025, <https://github.com/zahidrahimoon/50-Html-Interview-QnA>
4. Top 50+ HTML Interview Questions and Answers [2025] - LambdaTest, accessed on August 25, 2025, <https://www.lambdatest.com/learning-hub/html-interview-questions>
5. Top 90 HTML Interview Questions for Freshers & Experts - Simplilearn.com, accessed on August 25, 2025, <https://www.simplilearn.com/html-interview-questions-and-answers-article>
6. HTML Developer Interview Questions - Braintrust, accessed on August 25, 2025, <https://www.usebraintrust.com/hire/interview-questions/html-developers>
7. Top 40 HTML Interview Questions and Answers to Know in 2025!, accessed on August 25, 2025, <https://www.upgrad.com/blog/html-interview-questions-answers/>
8. 40 HTML Interview Questions - MentorCruise, accessed on August 25, 2025, <https://mentorcruise.com/questions/html/>
9. 100+ Most Useful Github Repositories Every Programmer Needs - DEV Community, accessed on August 25, 2025, <https://dev.to/johongirr/100-most-useful-github-repositories-every-programmer-needs-32cg>
10. 100 HTML Interview Questions and Answers in 2025 - Turing, accessed on August 25, 2025, <https://www.turing.com/interview-questions/html>
11. The 25 Most Common HTML Developers Interview Questions - Final Round AI, accessed on August 25, 2025, <https://www.finalroundai.com/blog/html-developer-interview-questions>
12. Top 10 CSS and HTML questions to ask interviewee? [closed] - Stack Overflow, accessed on August 25, 2025, <https://stackoverflow.com/questions/1960699/top-10-css-and-html-questions-to-ask-interviewee>
13. Prodjar/html-interview-questions: 100+ HTML5 Interview ... - GitHub, accessed on August 25, 2025, <https://github.com/Prodjar/html-interview-questions>
14. Awesome-JavaScript-Interviews/HTML/Collection-of-HTML ... - GitHub, accessed on August 25, 2025, <https://github.com/rohan-paul/Awesome-JavaScript-Interviews/blob/master/HTML/Collection-of-HTML-Interview-Questions.md>
15. Top HTML and HTML5 Interview Questions (2025) - InterviewBit, accessed on August 25, 2025, <https://www.interviewbit.com/html-interview-questions/>
16. 60 HTML interview questions to hire top developers - Adaface, accessed on August 25, 2025, <https://www.adaface.com/blog/html-interview-questions/>
17. Top HTML CSS JAVASCRIPT Interview Questions & Answers - H2K Infosys, accessed on August 25, 2025, <https://www.h2kinfosys.com/blog/top-html-css-javascript-interview-questions-answers/>
18. Saran-pariyar ... - GitHub, accessed on August 25, 2025, <https://github.com/Saran-pariyar/100_Days_Of_Frontend_Interview_Questions>
19. VINAYAK9669/99-HTML-INTTERVIEW-QUESTIONS: This ... - GitHub, accessed on August 25, 2025, <https://github.com/VINAYAK9669/99-HTML-INTTERVIEW-QUESTIONS>
20. 8 HTML Interview Questions Every Hiring Manager Should Ask, accessed on August 25, 2025, <https://www.interviewzen.com/blog/top-8-must-ask-html-interview-questions/>
21. MOST ASKED HTML Interview Questions (2025) | Intellipaat - YouTube, accessed on August 25, 2025, <https://www.youtube.com/watch?v=FhTvNXk4LEQ>
22. html - Most semantic way to markup a conversation (or interview)? - Stack Overflow, accessed on August 25, 2025, <https://stackoverflow.com/questions/8798685/most-semantic-way-to-markup-a-conversation-or-interview>
23. What html and css questions can I expect in an interview? : r/webdev - Reddit, accessed on August 25, 2025, <https://www.reddit.com/r/webdev/comments/tnngm9/what_html_and_css_questions_can_i_expect_in_an/>
24. Html and css interview questions. : r/Frontend - Reddit, accessed on August 25, 2025, <https://www.reddit.com/r/Frontend/comments/1db3oa7/html_and_css_interview_questions/>
25. 25+ HTML, CSS, and JavaScript Interview Questions - CoderPad, accessed on August 25, 2025, <https://coderpad.io/interview-questions/html-css-js-interview-questions/>