

HTML Interview Questions and Answers

Intern to senior level

HTML and browser interview questions and answers: semantic HTML, script loading with async and defer, browser storage, how the browser renders a page and Core Web Vitals.

5 questions Read online: <https://www.freedevtemplates.com/interview-questions/html>

1. What are semantic HTML elements and why do they matter?

HTML Easy Intern, Junior

Semantic elements describe their meaning in the name, such as header, nav, main, article and footer, so browsers, assistive technology and search engines can understand the page structure.

A `<div>` or `<span>` says nothing about the content. `<header>`, `<nav>`, `<main>`, `<article>`, `<section>`, `<aside>` and `<footer>` tell the browser what that region is for. Screen readers use them as landmarks, and search engines use them as a hint about what is primary content.

Use one `<main>` per page, put the site chrome in `<header>` and `<footer>`, and wrap a self-contained story or card in `<article>`. Keep using `<div>` when you only need a styling hook. Pair landmarks with a clear heading order (h1 then h2) rather than skipping levels.

Key points

- Semantic tags describe meaning, not just layout
 - They help accessibility, SEO and maintainability
 - div and span are still correct when there is no meaning to express
-

2. What is the difference between cookies, localStorage and sessionStorage?

HTML Easy Junior, Mid-level

Cookies are sent to the server with every matching request and can expire; localStorage persists until cleared and stays in the browser; sessionStorage lasts for the current tab only.

Use cookies for authentication when the server must see the value (prefer HttpOnly, Secure and SameSite). Use localStorage for non-sensitive UI preferences that should survive a restart. Use sessionStorage for a wizard or draft that should die when the tab closes.

Both storage APIs are readable by JavaScript, so they are the wrong place for access tokens if you have XSS. Cookies set without HttpOnly have the same problem. All three are origin-scoped. localStorage has a typical quota of about 5 MB; cookies should stay tiny because they travel on every request.

3. What is the difference between async and defer on script tags?

HTML Medium Mid-level, Senior

Both download the script without blocking HTML parsing. defer waits until parsing finishes and runs scripts in document order; async runs a script as soon as it is downloaded, which can be before or after parsing and in any order.

A classic `<script src>` in the head blocks parsing until the file is downloaded and executed. defer is the right default for module-style application code that depends on the DOM and on other scripts. async is for independent tags such as analytics, where order does not matter.

`type="module"` scripts are deferred by default. async and defer have no effect on inline scripts. Put critical, tiny inline scripts only when you must paint something before the main bundle arrives.

4. How does the browser render a web page?

HTML Medium Mid-level, Senior

The browser parses HTML into a DOM and CSS into a CSSOM, combines them into a render tree, computes layout (reflow), paints pixels and composites layers onto the screen.

This path is called the critical rendering path. Scripts without async/defer pause HTML parsing. CSS is render-blocking: the browser waits for stylesheets before first paint so it does not flash unstyled content. After the first paint, changing geometry (width, height, top) causes reflow; changing only color or transform can often skip to paint or composite.

Performance work targets each stage: reduce render-blocking CSS, defer non-critical JS, avoid layout thrashing (reading layout then writing styles in a loop), and keep animations on transform and opacity so they run on the compositor thread. Senior interviews often follow this with Core Web Vitals: LCP, INP and CLS.

Key points

- DOM + CSSOM render tree layout paint composite
- CSS blocks first paint; JS without async/defer blocks parsing
- Animate transform and opacity to avoid layout

5. What are Core Web Vitals and how do you improve them?

HTML Hard Senior

Core Web Vitals are Google's field metrics for real-user experience: LCP (loading), INP (interactivity) and CLS (visual stability). They influence search ranking and are measured on real devices, not only Lighthouse.

LCP (Largest Contentful Paint) should happen within about 2.5 seconds: compress the hero image, preload it, and cut render-blocking CSS/JS. INP (Interaction to Next Paint) replaced FID: keep event handlers short, break up long tasks, and show immediate feedback. CLS (Cumulative Layout Shift) should stay below 0.1: always set width and height (or aspect-ratio) on images and embeds, and avoid inserting banners above existing content.

Lab tools (Lighthouse, DevTools Performance) are for debugging. Rankings use field data from the Chrome User Experience Report. Fix the slowest template first, measure with web-vitals or Search Console, and do not chase a 100 Lighthouse score at the cost of features users need.

These answers are written for real interview conversations. Prefer explaining a trade-off out loud over reciting a definition. Full explanations and related questions: <https://www.fredevtemplates.com/interview-questions/html>