Absolutely! Here are 100 essential questions and answers for a frontend development interview, covering HTML, CSS, JavaScript, Frameworks (React/Vue/Angular concepts), Performance, and General Best Practices.

💻 HTML & CSS (Q1-Q35)

HTML Questions

Q1. What is the HTML <!DOCTYPE> declaration for?

A: It is a required preamble that is used to tell the browser which version of HTML the document is written in. For HTML5, it's simply <!DOCTYPE html>.

Note: It's not an HTML tag; it's an instruction to the web browser about the document type.

Q2. What is semantic HTML? Give an example.

A: Semantic HTML is the use of HTML markup to reinforce the meaning and structure of the information, rather than just dictating its presentation.

Example: Using <header>, <nav>, <main>, <article>, <section>, and <footer> instead of just <div> elements.

Note: This is crucial for accessibility (screen readers) and SEO (search engine optimization).

Q3. What is the difference between block, inline, and inline-block elements?

A:

- **Block:** Starts on a new line, takes up the full width available, and allows setting height and width. (e.g., <div>, <p>)
- **Inline:** Does not start on a new line, only takes up as much width as necessary, and ignores attempts to set height and width. (e.g., <span>, <a>)
- **Inline-Block:** Does not start on a new line (like inline) but allows setting height and width (like block). (e.g., <img>)

Q4. What are data attributes in HTML5 used for?

A: They allow you to store extra information/data on standard HTML elements without any hacky workarounds like non-standard attributes. They are accessed in JavaScript via element.dataset.

Example: <div data-user-id="12345">...</div>

Q5. Explain the difference between <canvas> and <svg>.

A:

- **Canvas:** Used for **drawing graphics via JavaScript**. It's a single raster image (pixels). Great for games, complex graphs, and real-time rendering.
- **SVG (Scalable Vector Graphics):** Used for **describing 2D graphics in XML**. Each drawn shape is an object in the DOM. Great for logos, icons, and complex designs that need to scale without loss of quality.

Q6. What is the purpose of the alt attribute on <img> tags?

A: The alt (alternative text) attribute provides a text description of the image for:

1. Users whose browsers fail to load the image.
2. Screen readers for visually impaired users (accessibility).
3. Search engines for understanding the image content (SEO).

Q7. What is the difference between defer and async attributes on a <script> tag?

A: Both are for external scripts to prevent blocking HTML parsing, but:

- **async:** The script is **downloaded asynchronously** and **executed immediately** as soon as it's available (which might interrupt HTML parsing). Order is not guaranteed.
- **defer:** The script is **downloaded asynchronously** but **executed only after** the HTML parsing is complete, and **before** the DOMContentLoaded event. Execution order is guaranteed.

Q8. How is the HTML DOM (Document Object Model) related to the browser?

A: The browser parses the HTML document and creates a tree-like representation of the document structure, which is the DOM. JavaScript can interact with this tree structure to modify the content, style, and structure of the web page.

CSS Questions

Q9. Explain the CSS Box Model.

A: The CSS Box Model describes how HTML elements are modeled as rectangular boxes for layout purposes. It consists of four parts, from inside out: Content, Padding, Border, and Margin.ShutterstockExplore

Note: The standard box model calculates the element's width/height based on **content** only. The box-sizing: border-box model is often preferred as it includes **padding and border** in the element's total width/height.

Q10. What is CSS Specificity and how is it calculated?

A: Specificity is the algorithm used by browsers to decide which CSS rule's property values are the most relevant and, therefore, get applied to an element. It's calculated based on a hierarchy:

1. **Inline Style** (Highest)
2. **IDs**
3. **Classes**, **Attributes**, and **Pseudo-classes**
4. Elements and Pseudo-elements (Lowest)

   Note: The formula is often represented as (a, b, c, d) where a is inline, b is IDs, c is classes/attributes/pseudo-classes, and d is elements/pseudo-elements. The rule with the higher number wins.

Q11. What is the difference between visibility: hidden; and display: none;?

A:

- **display: none;:** The element is completely removed from the document flow. It **does not take up any space** on the page.
- **visibility: hidden;:** The element is hidden, but it **still occupies its original space** in the document flow.

Q12. Describe the difference between rem and em units.

A:

- **em:** Relative to the **font-size of its parent element**. Can lead to compounding issues (where nesting increases font size unintentionally).
- **rem (root em):** Relative to the **font-size of the root HTML element** (<html>). This provides a consistent, global scaling factor and is generally preferred for font sizing in scalable, maintainable CSS.

Q13. Explain z-index and the concept of Stacking Context.

A:

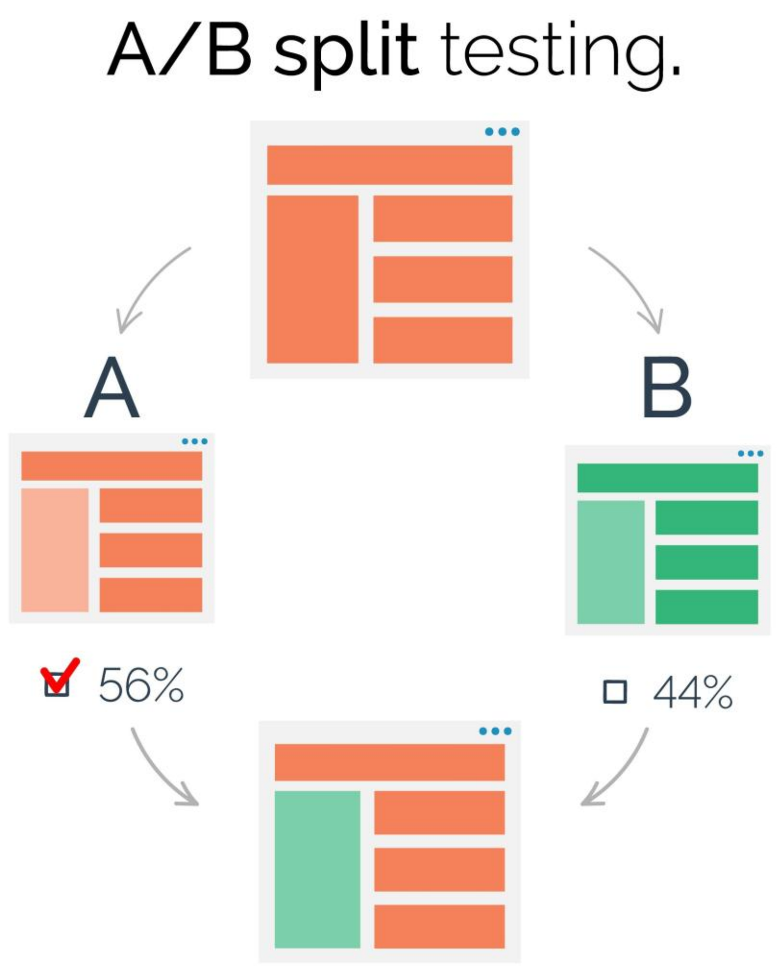
- **z-index:** A CSS property that specifies the **stack order** of an element that is **positioned** (i.e., position: absolute, relative, fixed, or sticky). A higher z-index means the element is closer to the user.
- **Stacking Context:** A three-dimensional conceptualization of HTML elements along an imaginary Z-axis. A new stacking context is created by any element with a position value

other than static AND a z-index value other than auto, or by properties like opacity < 1, transform, filter, etc. z-index values only compare elements within the **same stacking context**.

Q14. How do Flexbox and CSS Grid differ?

A:

- **Flexbox (One-dimensional):** A layout model designed for arranging items in a single line (either a row or a column). Great for components, navigation bars, and distributing space.
- **CSS Grid (Two-dimensional):** A layout model designed for arranging items in two dimensions (both rows and columns). Great for main page layouts and complex overlapping designs.



Shutterstock

Q15. How can you center a <div> element both horizontally and vertically?

A: The most modern and robust way is using Flexbox on the parent container:

```
CSS
.parent {
  display: flex;
  justify-content: center; /* Centers horizontally */
  align-items: center;    /* Centers vertically */
  height: 100vh;          /* Example height */
}
```

Q16. What are CSS Media Queries?

A: Media queries are a CSS technique that allows you to apply different styles based on the characteristics of the device being used to view the webpage, such as screen width, height, resolution, or orientation. They are the cornerstone of Responsive Web Design.

Example: @media (max-width: 600px) { ... }

Q17. What are CSS Preprocessors? Give examples.

A: Preprocessors like Sass, Less, and Stylus are languages that extend CSS, compiling their own syntax into regular CSS. They add programming capabilities to CSS, such as:

- Variables
- Nesting
- Mixins (reusable blocks of styles)
- Functions and logic

🚀 **JavaScript Fundamentals (Q36-Q75)**

Core JavaScript & ES6

Q18. What is Hoisting in JavaScript?

A: Hoisting is a JavaScript mechanism where variable and function declarations are moved to the top of their containing scope during the compilation phase, before code execution.

Note: Only the declaration is hoisted, not the initialization. let and const declarations are also hoisted, but they are subject to the Temporal Dead Zone (TDZ), meaning you cannot access them until their declaration line is executed.

Q19. What is the difference between var, let, and const?

A:

Feature	var	let	const
Scope	Function-scoped	Block-scoped	Block-scoped

Hoisting	Hoisted and initialized to undefined	Hoisted, but not initialized (TDZ)	Hoisted, but not initialized (TDZ)
Re-declaration	Allowed	Not Allowed	Not Allowed
Re-assignment	Allowed	Allowed	Not Allowed (must be initialized)

Q20. What is a Closure?

A: A closure is the combination of a function bundled together (enclosed) with references to its surrounding state (lexical environment). In simple terms, a closure gives you access to an outer function’s scope from an inner function, even after the outer function has finished executing.

Use Case: Data privacy/encapsulation, memoization, and creating function factories.

Q21. Explain the Event Loop.

A: The Event Loop is a mechanism that allows JavaScript (which is single-threaded) to perform non-blocking I/O operations by offloading them to the system kernel (or other threads) and managing their return using the Call Stack, Heap, Task Queue, and Microtask Queue.

Note: It continuously checks if the Call Stack is empty. If it is, it moves waiting functions (tasks/microtasks) from the queue to the stack.

Q22. What are Promises and why are they used?

A: A Promise is an object representing the eventual completion (or failure) of an asynchronous operation and its resulting value. They are used to manage asynchronous operations more easily than traditional callbacks, preventing "callback hell."

States: Pending, Fulfilled (resolved), or Rejected.

Q23. What is the difference between == and ===?

A:

- **== (Abstract Equality):** Compares for equality **after performing type coercion** (trying to convert types to match). (e.g., 1 == '1' is true)
- === (Strict Equality): Compares for equality without performing type coercion. Both the value and the type must be the same. (e.g., 1 === '1' is false)

  Note: Always prefer === to avoid unexpected type coercion bugs.

Q24. What are Arrow Functions and how do they differ from normal functions?

A: Arrow functions are a concise syntax for writing function expressions. Their main difference is how they handle the this keyword:

- **Arrow Functions:** Do **not** have their own this context. They inherit this from their **surrounding (lexical) scope**.
- **Normal Functions:** Have their own dynamic this context, which is determined by how the function is called.

Q25. Explain Destructuring in ES6.

A: Destructuring is a JavaScript expression that makes it possible to unpack values from arrays or properties from objects into distinct variables.

Example (Object): const { name, age } = user;

Example (Array): const [first, second] = myArray;

Q26. What is the Spread Operator (...) and the Rest Parameter (...)?

A:

- **Spread Operator:** Used to **expand** an iterable (like an array) into its individual elements.
  - **Use Cases:** Copying arrays/objects, merging arrays/objects, passing elements as function arguments.
- **Rest Parameter:** Used in a function's parameter list to **collect** all remaining arguments into a new array.
  - **Use Case:** Defining functions that accept a variable number of arguments.

DOM & API

Q27. What is Event Bubbling and Event Capturing?

A: These are the two ways event propagation works in the DOM.

- **Bubbling (Default):** The event first triggers on the **innermost** element and then "bubbles up" to its parent elements in the DOM tree.
- **Capturing:** The event first triggers on the **outermost** element (the window) and then moves "down" the DOM tree to the target element.

Q28. What is Event Delegation?

A: A technique where you attach a single event listener to a parent element instead of attaching multiple listeners to all its child elements. The parent listener handles events for all its descendants by leveraging event bubbling.

Benefit: Improves performance and simplifies code when managing many similar elements.

Q29. How do you make an AJAX request?

A: The modern approach is to use the Fetch API.

```
JavaScript
fetch('url')
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error(error));
```

An older but still valid way is using the XMLHttpRequest object.

Q30. What is the difference between localStorage, sessionStorage, and Cookies?

A:

Feature	**localStorage**	**sessionStorage**	**Cookies**
Duration	Persistent (until manually deleted)	Cleared when the browser tab/window is closed	Set by expiration date (can be persistent or session-based)
Storage Limit	5MB - 10MB	5MB - 10MB	4KB (sent with every HTTP request)
Accessibility	Client-side (JS)	Client-side (JS)	Client-side and Server-side (via HTTP header)

🖼️ Architecture & Frameworks (Q76-Q100)

General & Performance

Q31. How do you optimize a website for maximum performance?

A: Key strategies include:

- **Minimize HTTP requests:** Use CSS Sprites, combine/minify files.
- **Optimize assets:** Compress images, use modern formats (WebP).
- **Leverage caching:** Use proper HTTP caching headers and Service Workers.
- **Minimize critical rendering path:** Use **lazy loading** for images/components below the fold, and **defer/async** for non-critical JavaScript.
- **Use a CDN (Content Delivery Network).**

Q32. What is the Critical Rendering Path?

A: The sequence of steps the browser follows to convert HTML, CSS, and JavaScript into pixels on the screen. Optimizing this path involves prioritizing the content and styles needed for the First Contentful Paint (FCP).

Steps: Constructing the DOM (from HTML) and CSSOM (from CSS), combining them to form the Render Tree, performing Layout, and finally Painting.

Q33. Explain Server-Side Rendering (SSR) vs. Client-Side Rendering (CSR).

A:

- **CSR:** The browser receives a minimal HTML file (mostly empty), downloads JavaScript, and then the JavaScript renders the content in the browser.
  - **Pros:** Good for interactivity after initial load.
  - **Cons:** Slower initial load time, poor SEO.
- **SSR:** The server renders the full HTML content before sending it to the client. JavaScript takes over afterward (a process called **Hydration**).
  - **Pros:** Better initial load time, excellent SEO.
  - **Cons:** Slower first byte time, higher server load.

Q34. What is Cross-Origin Resource Sharing (CORS)?

A: A security mechanism implemented by web browsers that allows a web application running at one origin (domain, protocol, and port) to access selected resources from a different origin. It is implemented by the server using specific HTTP headers (like Access-Control-Allow-Origin).

Q35. What is the purpose of Web Components?

A: A set of web platform APIs that allow you to create new, reusable, custom HTML tags with encapsulated functionality and styling. They are framework-agnostic.

Key Technologies: Custom Elements, Shadow DOM (for encapsulation), and HTML Templates.

Framework/Library Concepts (e.g., React, Vue, Angular)

Q36. Explain the concept of a Virtual DOM (VDOM).

A: The VDOM is an in-memory representation of the real browser DOM. When an application's state changes, the framework (like React) first updates the VDOM. It then compares the new VDOM to the previous one (a process called "diffing"). Finally, it batches and applies only the necessary changes to the real DOM, optimizing performance by minimizing expensive DOM manipulation.

Q37. What are Components and why are they fundamental to modern frontend frameworks?

A: Components are independent, reusable, and isolated pieces of UI. They are the building blocks of an application, encapsulating their own logic, layout, and appearance.

Benefit: Modularity, reusability, and easier maintenance.

Q38. What is State and Props?

A:

- **State:** Data that is managed **within** a component, and can change over time. When the state changes, the component usually re-renders.
- Props (Properties): Data passed from a parent component to a child component. They are generally considered immutable (read-only) in the child component.

  Note: Data flow is generally unidirectional (parent to child) for simplicity and predictability.

Q39. What is Unidirectional Data Flow?

A: Data in the application flows in a single direction (e.g., Parent to Child via Props). This makes the application predictable, easier to understand, and simplifies debugging, as you can trace where changes originate.

Q40. What is TypeScript and what are its benefits?

A: TypeScript is an open-source language developed by Microsoft that is a superset of JavaScript that compiles to plain JavaScript.

Benefits: Adds static typing (finds errors before runtime), provides better tooling (autocompletion, refactoring), and improves code maintainability for large-scale applications.

Soft Skills & Tools

Q41. What is the difference between git merge and git rebase?

A:

- **git merge:** Combines the histories of two branches by creating a **new merge commit**. It preserves the full history, including the merge point.
- **git rebase:** Moves or combines a sequence of commits to a new base commit. It effectively **rewrites history** to create a cleaner, linear commit history, avoiding the extra merge commit.

Q42. How do you ensure your code is accessible (A11y)?

A: By following best practices like:

- Using **semantic HTML** (Q2).
- Providing proper **alt attributes** for images (Q6).
- Using **ARIA attributes** (aria-label, role, etc.) to convey information to screen readers for non-standard elements.
- Ensuring adequate **color contrast** and **keyboard navigation**.

Q43. What is the role of a module bundler like Webpack or Vite?

A: They are tools that take all of an application's individual assets (JavaScript, CSS, images, etc.) and package them up for production deployment.

Key Functions: Dependency resolution, code splitting, minification, transpilation (e.g., converting ES6+ to ES5), and asset optimization.

Q44. What is a Design System and why is it valuable?

A: A comprehensive set of standards, documentation, and reusable UI components that help teams build a consistent user interface across all products.

Value: Faster development, greater consistency, and a shared vocabulary between designers and developers.

Q45. How do you approach debugging a complex frontend issue?

A:

1. **Reproduce:** Consistently replicate the bug.
2. **Isolate:** Use binary search or comment out sections to isolate the faulty code/component.
3. **Inspect:** Use browser DevTools (console logs, breakpoints, network tab, performance profiler) to observe the state, call stack, and network activity.
4. **Hypothesize & Test:** Formulate a theory for the cause and test the fix.

BONUS Questions (Q46-Q50)

Q46. What is the concept of memoization?

A: A programming technique used to speed up computer programs by storing the results of expensive function calls and returning the cached result when the same inputs occur again.

Q47. What is requestAnimationFrame() and why is it better for animation than setTimeout/setInterval?

A: It tells the browser that you want to perform an animation and requests that the browser calls a specified function to update an animation before the browser's next repaint.

Benefit: It ensures the animation runs at the ideal 60 frames per second (or the screen's refresh rate) and stops running when the tab is inactive, saving battery and CPU.

Q48. What are Custom Events in JavaScript?

A: Events that you, the developer, can create and trigger on an element, allowing decoupled communication between different parts of your application without tight dependencies. They are created using new CustomEvent('name', { detail: data }) and triggered using element.dispatchEvent(event).

Q44. Explain the principle of Progressive Enhancement vs. Graceful Degradation.

A:

- **Progressive Enhancement:** Start with a baseline, accessible, and functional experience for the widest range of browsers/devices, and then add complex features only for modern browsers/capable devices.
- Graceful Degradation: Start by building the full, most advanced version of the application and then degrade features for older browsers, ensuring it remains somewhat functional.

  Note: Progressive Enhancement is generally the preferred, more modern philosophy, focusing on user needs first.

Q50. What is Throttling and Debouncing?

A: Techniques used to limit the rate at which a function executes, especially on high-frequency events (like window resize, scroll, or search box input).

- **Debouncing:** Ensures a function is called only **after** a specified delay since the last call. (e.g., calling an API *after* a user stops typing for 300ms).
- **Throttling:** Ensures a function is called at **most once** in a specified time period. (e.g., updating scroll position every 100ms, regardless of how fast the user scrolls).