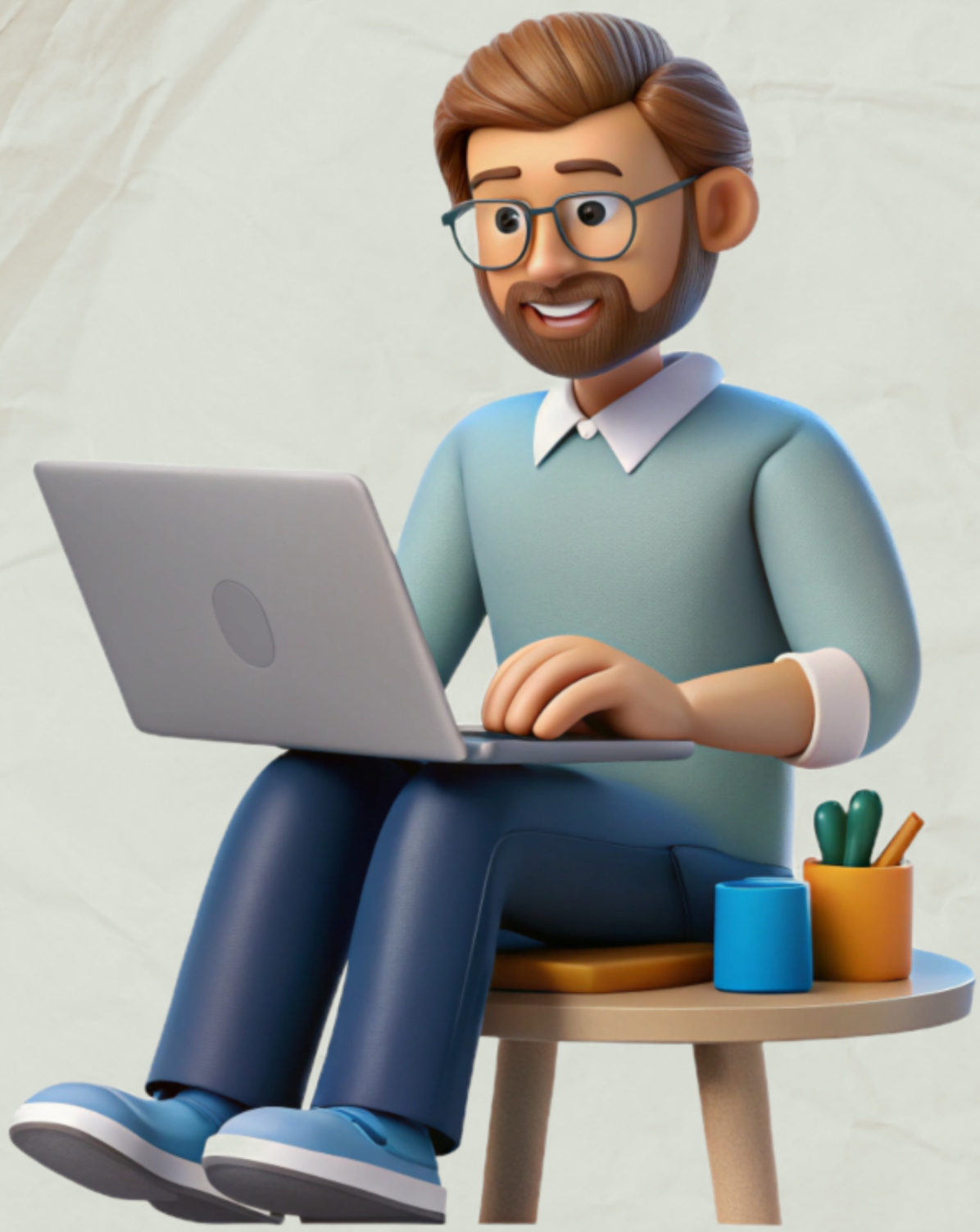
Hadil Ben Abdallah
@hadil-ben-abdallah

40 Web Developer Questions Recruiters Ask in 2025



Introduction

Web development interviews in 2025 are no joke. With the rapid evolution of technologies, frameworks, and best practices, recruiters are raising the bar. Whether you're a frontend magician, a backend pro, or a full-stack master, you need to be ready for questions that go deeper than "What's React?"

Here, I'll walk you through **40 handpicked interview questions** , not just with plain answers, but a little background to explain why each topic matters.

1 What is the Virtual DOM in React? ❓

React remains a dominant frontend library in 2025, and the **Virtual DOM** is at the heart of its performance magic. Interviewers love this question because it reveals how well you understand rendering and re-rendering.

Answer:

The Virtual DOM is a lightweight, in-memory representation of the real DOM. Instead of updating the actual DOM every time there's a change in state, React creates a Virtual DOM, updates it first, and then compares it with the previous version. This process is called **diffing**. Only the changes are then applied to the real DOM, making updates faster and more efficient.

2 What's the difference between SSR, CSR, and SSG? 🧠

This question is about **rendering strategies**, a hot topic with frameworks like Next.js and Astro pushing the boundaries of performance.

Answer:

Server-Side Rendering (SSR) generates the HTML for each request on the server, which is great for SEO and dynamic content. Client-Side Rendering (CSR) builds a mostly blank HTML page and then loads content using JavaScript in the browser, making it fast after the initial load. Static Site Generation (SSG) creates HTML pages during build time, which can be served instantly but isn't great for frequently updated data.

3

What is tree shaking in JavaScript bundlers like Webpack or Vite? 🌳

Tree shaking sounds like a gardening term, but it’s crucial for performance. Interviewers use this to test your knowledge of **build optimization**.

Answer:

Tree shaking is the process of removing unused or “dead” code during bundling. Modern JavaScript bundlers analyze which parts of your code are actually used, and exclude anything else from the final bundle. This reduces file size and speeds up your app.

Hadil Ben Abdallah
@hadil-ben-abdallah

4 What's new in React 19? 💥

React keeps evolving, and interviewers expect you to keep up , especially with **major versions** like React 19, which introduces game-changing improvements.

Answer:

React 19 introduced features like the **React Compiler**, which helps optimize components automatically by memoizing them. It also added **built-in async handling** using "use" and "actions", better **React Server Components**, and improved streaming for SSR. These features push React closer to native-like performance and developer ergonomics.

5

What's the difference between == and === in JavaScript? 🔗

This one may seem basic, but it often trips people up. It's a **core JavaScript concept**, and getting it wrong could mean unexpected bugs.

Answer:

The == operator compares values after type coercion, meaning it will try to convert one type to another if they're different. For example, 5' == 5 returns true. On the other hand, === checks both value and type, so '5' === 5 returns false. Always prefer === to avoid unpredictable behavior.

6

What is responsive design? 📱

User experience across devices matters more than ever in 2025. Interviewers want to know if you think mobile-first and design for accessibility and usability.

Answer:

Responsive design is about making web content look good on all screen sizes, phones, tablets, laptops, or desktops. It uses fluid grids, flexible images, and CSS media queries to adapt layouts dynamically. A well-crafted responsive UI ensures every user has a consistent experience.

Hadil Ben Abdallah
@hadil-ben-abdallah

7

What are Web Components and why are they useful? 🔒

This is about **future-proof, reusable UI building blocks**. Web Components are especially relevant in design systems and micro-frontend architecture.

Answer:

Web Components are a set of browser-native technologies that allow you to create custom, encapsulated HTML elements. They include Shadow DOM for style scoping, custom elements for creating your own tags, and HTML templates. They work across frameworks and are reusable and modular.

8

What is hydration in modern frameworks like Next.js? ⚡

Hydration is a buzzword in frameworks focused on performance and SEO. This question checks if you know how server-rendered pages become interactive.

Answer:

Hydration is the process where a server-rendered HTML page is turned into a fully interactive web page by attaching JavaScript event listeners in the browser. This makes SSR-powered pages feel dynamic and responsive after loading.

9

What is the role of unit testing in frontend development? 🧪

Testing is no longer optional. This question reveals if you're someone who ships confidently, or someone who just “hopes for the best.”

Answer:

Unit testing involves testing individual pieces (functions, components) of your frontend to ensure they work as expected. This helps catch bugs early, supports refactoring, and improves code quality and reliability.

Hadil Ben Abdallah
@hadil-ben-abdallah

10

What is the purpose of a CSS preprocessor like Sass? 🧵

They want to see if you write maintainable, scalable CSS. Preprocessors were a game-changer, even with Tailwind and CSS-in-JS today.

Answer:

Sass allows you to write cleaner and more structured CSS with variables, mixins, nesting, and functions. It helps avoid repetition and enables better organization of styles, especially in large codebases.

11 What is REST, and how does it differ from GraphQL? 🚦

This question checks whether you understand **API architecture** and can make smart decisions based on the project.

Answer:

- REST is an architectural style where endpoints represent resources and use standard HTTP methods like GET, POST, PUT, DELETE. Each request from the client to the server must contain all the information the server needs.
- GraphQL, on the other hand, is a query language that allows clients to **ask for exactly what they need**, no more, no less. It solves issues like over-fetching or under-fetching data. REST is simple and widely used, while GraphQL offers more flexibility, especially for frontend-heavy apps.

12

What are HTTP status codes, and why do they matter? 🔐

This is about communication, between your backend and the client. Interviewers love this question because misusing status codes leads to **bad DX (developer experience)** and debugging nightmares.

Answer:

HTTP status codes indicate the result of a client's request to the server. For example:

- "***200 OK***" means the request was successful.
- "***404 Not Found***" means the resource doesn't exist.
- "***500 Internal Server Error***" means something broke on the server.

13

What is middleware in backend frameworks like Express? 🛠️

Middleware might sound like fluff, but it’s the **pipeline** through which every request flows. Interviewers want to see if you understand how to extend and structure backend behavior.

Answer:

Middleware functions act like filters or processors for incoming HTTP requests before they reach your main route logic. They can handle authentication, logging, data validation, error handling, and more. For example, in Express.js, you can write middleware that checks if a user is authenticated before letting them access a route.

14

What is rate limiting, and how do you implement it? 🌪️

This is a security + performance combo question. Interviewers want to know you can **protect APIs from abuse**.

Answer:

Rate limiting restricts the number of requests a client can make in a given time window. This prevents misuse (e.g., bots, DDoS attacks) and helps maintain server health. Implementation can be done using libraries like “express-rate-limit” or tools like NGINX or Redis-based solutions for scalable enforcement.

Hadil Ben Abdallah
@hadil-ben-abdallah

15 How does authentication differ from authorization? 🙋‍♂️

You'll almost always get this one. It's essential to secure web applications, and it shows if you understand **access control** basics.

Answer:

Authentication is about verifying who someone is (e.g., via email/password, OAuth).

Authorization is about determining what that person is allowed to do (e.g., can they edit a post?). In short: **authenti**cation comes first, then **authori**zation.

16 What is JWT and when would you use it? 🧩

This one is often asked when talking about **stateless authentication**, especially in microservices or mobile apps.

Answer:

JWT (JSON Web Token) is a compact, URL-safe token format used to securely transmit information between parties. It’s commonly used for authentication. Once a user logs in, the server issues a signed token that the client sends with each request. The server can verify the token without storing session data, making JWT ideal for stateless APIs.

Hadil Ben Abdallah
@hadil-ben-abdallah

17 What is asynchronous programming, and why is it important in the backend? ⏱️

Backend performance relies heavily on handling multiple tasks without blocking. This question checks if you know how to write **non-blocking code**.

Answer:

Asynchronous programming allows tasks like database queries or API calls to run in the background without freezing the server. In JavaScript (Node.js), we use “async/await” or Promises; in Python (Django or FastAPI), there’s “async def”. It helps scale your app efficiently by keeping the event loop responsive.

18

What is a RESTful route? Give an example. 📂

Understanding RESTful conventions makes your code more readable and standardized. This is a favorite entry point for REST API questions.

Answer:

A RESTful route aligns with CRUD operations. For example, a “GET /api/posts” route fetches all posts, while “POST /api/posts” creates a new one. “GET /api/posts/1” fetches a post with ID 1, and “DELETE /api/posts/1” removes it. RESTful routes are intuitive and map closely to database actions.

19 What is input validation and why is it necessary? 🫧

This isn’t just a “good practice”, it’s your first line of defense against **injection attacks and corrupted data.**

Answer:

Input validation ensures that data sent from clients is correct and safe. For example, it checks that emails are in a valid format or that passwords meet security criteria. Without validation, attackers could send malicious data (like SQL injection), or your database could become inconsistent.

Hadil Ben Abdallah
@hadil-ben-abdallah

20 What is the MVC architecture? 🧱

Frameworks like Django, Rails, and Laravel use this classic structure. Interviewers want to see if you organize code thoughtfully.

Answer:

MVC stands for Model–View–Controller. The Model handles the data and business logic, the View renders the UI, and the Controller handles user input and interactions. This separation makes your code modular, easier to maintain, and more scalable.

21

What is the difference between SQL and NoSQL databases? 🧮

Knowing when to use each is crucial. This shows how well you understand **data modeling, consistency** and **scalability.**

Answer:

SQL databases (like PostgreSQL, MySQL) are relational and use structured schemas with tables. They’re great for complex queries and data integrity. NoSQL databases (like MongoDB, Redis) are non-relational, flexible, and scale horizontally, making them ideal for unstructured or fast-evolving data.

22 What is indexing in databases? 🔍

Performance is everything. Indexing is a key strategy to optimize database queries and make apps faster.

Answer:

An index is a data structure that improves the speed of data retrieval at the cost of extra storage and write speed. It’s like a table of contents, instead of scanning the whole table, the database can quickly find the row using the index.

Hadil Ben Abdallah
@hadil-ben-abdallah

23

What is the N+1 query problem and how do you avoid it? 🚧

This one tests your ability to optimize backend and database interactions. N+1 problems can destroy app performance.

Answer:

The N+1 query problem happens when you run one query to fetch records, and then N additional queries to fetch related data for each record. To avoid it, use techniques like **eager loading** (“JOIN, select_related”, or “populate” depending on your stack) to fetch all needed data in fewer queries.

24 What is database normalization? 📈

Normalization ensures your schema is clean and efficient. This is a favorite question for testing **database design skills**.

Answer:

Normalization is the process of organizing data to reduce redundancy and improve data integrity. It involves splitting large tables into smaller ones and defining relationships between them. Common forms are 1NF, 2NF, and 3NF, each reducing duplication and dependency.

25

How do you optimize a slow SQL query?

This question tests your **real-world problem-solving** when facing performance bottlenecks.

Answer:

Start by analyzing the query with “EXPLAIN” or a similar tool. Check for missing indexes, avoid “SELECT *”, reduce joins, and look at query logic. Also consider denormalizing data, caching, or restructuring queries to improve performance.

Hadil Ben Abdallah
@hadil-ben-abdallah

26

What is connection pooling and why is it important? 🧰

Scalability hinges on how well you manage resources like DB connections. Interviewers want to see if you understand resource reuse.

Answer:

Connection pooling allows multiple requests to share a small number of database connections. It reduces the overhead of repeatedly creating and closing connections and improves application scalability and performance.

27 How does CDN improve web performance? 🌎

Web devs should know how to make their apps feel faster across the globe.

Answer:

A CDN (Content Delivery Network) caches your static assets (JS, CSS, images) in servers around the world. Users fetch these assets from the closest edge server, reducing latency and improving load speed, especially for global audiences.

28

What is caching and what tools do you use? 🔄

Caching is essential for scalability. This question explores your ability to reduce load and speed up responses.

Answer:

Caching stores frequently accessed data temporarily to reduce server load and response time. Common tools include Redis, Memcached, and browser cache. You can cache API responses, DB queries, or even entire HTML pages.

Hadil Ben Abdallah
@hadil-ben-abdallah

29

What is lazy loading and how does it improve performance? 🚀

This is about optimizing UX by **loading only what’s necessary** when it's needed.

Answer:

Lazy loading defers loading of non-critical resources (like images, components) until they are needed. It reduces initial page load time and saves bandwidth, improving performance especially on slower networks.

30

What is the difference between vertical and horizontal scaling? ⚖️

Scaling decisions impact architecture and cost. This question tests how well you plan for **growth and load**.

Answer:

Vertical scaling adds more power (CPU, RAM) to a single machine. Horizontal scaling adds more machines to distribute load. Horizontal scaling is generally more fault-tolerant and elastic, often used in cloud environments.

31

How do you approach debugging a web application? 🧪

This is a must-answer for any developer. Debugging is a daily skill, show that you're methodical.

Answer:

I start by isolating the issue with browser dev tools, logs, or breakpoints. Then I replicate the bug and work backwards through the stack. I look for recent changes, validate inputs/outputs, and write test cases to prevent regressions.

Hadil Ben Abdallah
@hadil-ben-abdallah

32

Tell me about a time you improved performance in an app.

Scaling decisions impact architecture and cost. This question tests how well you plan for **growth and load**.

Answer:

Vertical scaling adds more power (CPU, RAM) to a single machine. Horizontal scaling adds more machines to distribute load. Horizontal scaling is generally more fault-tolerant and elastic, often used in cloud environments.

33

How do you handle negative feedback in a code review? 👂

This reveals emotional intelligence and growth mindset, qualities of a great team player.

Answer:

I view feedback as an opportunity to improve. I thank the reviewer, ask clarifying questions if needed, and update my code. I try to separate the critique from myself personally and keep a collaborative attitude.

34 What tools do you use for debugging? 🛠️

This one checks if you're equipped to solve problems effectively.

Answer:

I use browser dev tools, Postman for APIs, "console.log" or breakpoints for JS, and logging libraries like Winston or Django's logger. For frontend, React DevTools is invaluable; on the backend, I use debugging tools like "pdb" in Python or "node inspect".

Hadil Ben Abdallah
@hadil-ben-abdallah

35

How do you prioritize tasks when everything feels urgent? ⏳

Interviewers love to ask this in startups or fast-paced teams. They want to see how you think under pressure.

Answer:

I assess urgency vs. importance. I use the Eisenhower matrix or tools like Jira to break down tasks, estimate effort, and align with business priorities. I also communicate with my team or manager to confirm the best path forward.

36

What's your approach when facing a problem you've never seen before? 🤔

This reveals your mindset toward **unknowns** and **lifelong learning**.

Answer:

I break the problem down, research similar issues, and experiment with small tests. I search docs, forums, or ask peers. I try to learn as I go instead of just patching the issue, so I can solve it better next time.

37

How do you collaborate with other developers on a large codebase? 🤝

This question tests your ability to **work well in teams** and write code others can build on.

Answer:

I stick to team conventions, write clear commit messages, and document decisions. I use Git branches, PRs, and review others' code to stay in sync. Communication is key, I ask questions and give feedback respectfully.

Hadil Ben Abdallah
@hadil-ben-abdallah

38

How do you ensure code quality in your projects? 🔍

This is about professionalism, do you leave code better than you found it?

Answer:

I write tests (unit/integration), use linters (ESLint, Prettier), and automate checks in CI/CD pipelines. I also break code into small, manageable pieces, write meaningful names, and follow clean code practices.

39

How do you handle production incidents or bugs? 🚑

Can you stay calm under fire? This shows your maturity and process under pressure.

Answer:

I stay calm, assess the blast radius, and communicate with the team. I check logs, metrics, and recent changes. If needed, I roll back, patch, or apply hotfixes. Afterward, I conduct a postmortem to learn and prevent recurrence.

40

How do you stay up-to-date with web development trends? 📚

This reveals your curiosity and commitment to your craft.

Answer:

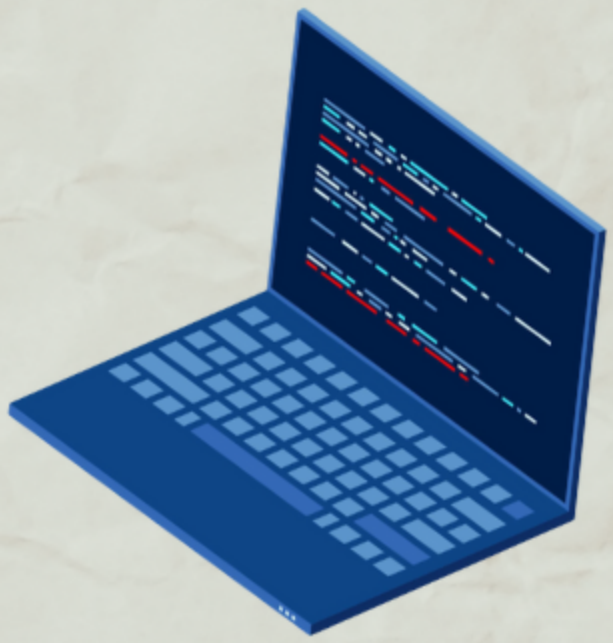
I follow thought leaders on Twitter, read blogs (CSS-Tricks, Smashing Mag, etc.), watch conference talks, and build side projects. I also use newsletters, GitHub trending, and communities like dev.to and Reddit.

Hadil Ben Abdallah
@hadil-ben-abdallah

🎉 Final Thoughts

Interviews in 2025 are about more than just syntax, they’re about systems thinking, architecture, collaboration, and adaptability. Whether it’s frontend interactivity, backend logic, database design, or performance tuning, these 40 questions will help you shine 💎

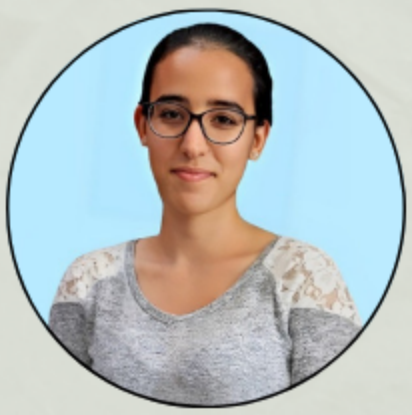
🔮 **Final Tip**: Don’t just memorize answers, understand the *why* behind them.



Found this helpful ?

Share to help others

Save to refer to later



Follow Hadil Ben Abdallah for more amazing content about programming