List interview question for React JS for 3+ years exp. developer

Finished

Here's a list of React JS interview questions suitable for a developer with 3+ years of experience, covering various crucial concepts and advanced topics:

Core Concepts & Fundamentals:

1. **Explain the Virtual DOM and how React uses it for performance optimization.**
2. **What is reconciliation in React, and how does it contribute to efficient UI updates?**
3. **Differentiate between controlled and uncontrolled components in React. When would you use each?**
4. **Explain the purpose of the `key` prop in React lists and the consequences of using array indices as keys.**
5. **What are React Fragments, and why are they useful?**
6. **How does React support single-page applications (SPAs)?**

Hooks:

7. **Explain React Hooks and why they were introduced.**
8. **When would you use `useMemo` and `useCallback`? Provide examples of their benefits.**
9. **Describe the `useEffect` hook. Discuss "effects without cleanup" and "effects with cleanup" with examples.**
10. **What is the `useContext` hook, and how does it help with prop drilling?**
11. **Explain the `useReducer` hook and scenarios where it might be preferred over `useState`.**
12. **What is the `useRef` hook, and when should it be used?**

Advanced Concepts & Patterns:

13. **What are Higher-Order Components (HOCs) and Render Props? Discuss their use cases and differences.**
14. **Explain prop drilling in React and potential solutions to mitigate it.**
15. **Discuss different approaches to state management in large React applications (e.g., Context API, Redux, Zustand).**
16. **How do you optimize the performance of a React application? Mention techniques like lazy loading, code splitting, and memoization.**
17. **Describe error boundaries in React and how to implement them.**
18. **Explain server-side rendering (SSR) and client-side rendering (CSR) in the context of React. What are the advantages and disadvantages of each?**

19. How do you handle asynchronous operations (e.g., API calls) in React components?
20. Discuss the importance of accessibility in React applications and how you ensure it.

JavaScript Fundamentals (Relevant to React):

21. Explain closures and scope in JavaScript.
22. Describe the JavaScript event loop and its role in asynchronous operations.
23. Discuss common array methods and their practical applications in React.

Sources

AURA 1.0 Fast
Finished In: 10s

more questions

Finished

Here are more React JS interview questions for a developer with 3+ years of experience, focusing on deeper understanding, best practices, testing, and the broader ecosystem:

Performance & Optimization (Deep Dive):

1. Explain the concept of memoization in React. How do `React.memo`, `useMemo`, and `useCallback` contribute to performance, and what are their trade-offs?
2. How would you identify and debug performance bottlenecks in a React application? What tools or techniques would you use?
3. Discuss strategies for optimizing large lists or tables in React (e.g., virtualization/windowing).
4. When would you consider using `shouldComponentUpdate` (for class components) or `React.memo` (for functional components), and what are their differences?
5. Explain code splitting and lazy loading in React. How do you implement them, and what are their benefits?

Testing:

6. What is your approach to testing React components? Discuss different types of tests (unit, integration, end-to-end) and the tools you'd use (e.g., Jest, React Testing Library, Cypress).
7. How do you test a component that interacts with an external API?
8. Explain how to test custom hooks effectively.
9. What are the advantages of React Testing Library over Enzyme?

State Management (Advanced):

10. When would you choose Redux over the Context API for state management, and vice versa?
11. Describe the typical Redux flow (actions, reducers, store, mapStateToProps, mapDispatchToProps, thunks/sagas).
12. How do you handle side effects in Redux (e.g., with Redux Thunk or Redux Saga)?
13. Discuss alternative state management libraries (e.g., Zustand, Jotai, Recoil) and when you might consider using them.

Architecture & Best Practices:

14. How do you structure a large-scale React application for maintainability and scalability? (e.g., folder structure, component organization).
15. Discuss the importance of component reusability and how you design components with that in mind.
16. What are common anti-patterns in React development, and how do you avoid them?
17. Explain the concept of "separation of concerns" in a React application.
18. How do you manage forms in React, especially complex forms with validation?

Error Handling & Debugging:

19. Beyond error boundaries, what other strategies do you employ for robust error handling in a React application?
20. Describe your debugging process for a complex bug in a React application. What tools do you use?

Ecosystem & Tooling:

21. Discuss your experience with build tools like Webpack or Vite in a React context.
22. What is your familiarity with TypeScript in React projects? What benefits does it bring?
23. Have you worked with any React UI libraries (e.g., Material-UI, Ant Design, Chakra UI)? Discuss your experience.
24. Explain the role of a bundler (like Webpack) in a React project.
25. What are your thoughts on "CSS-in-JS" solutions (e.g., Styled Components, Emotion) versus traditional CSS/Sass modules?

React 18 and Beyond:

26. What are the new features introduced in React 18 that you are excited about or have worked with? (e.g., automatic batching, `startTransition`, `useDeferredValue`, concurrent rendering).
27. Explain the concept of concurrent rendering in React.
28. How do `startTransition` and `useDeferredValue` help improve user experience in React 18?

more questions

Finished

Okay, let's dive even deeper with more React JS interview questions for a 3+ years experienced developer, focusing on advanced scenarios, theoretical understanding, and practical problem-solving.

Advanced Hooks & Context Scenarios:

1. **When might you create a custom hook? Provide an example of a common problem a custom hook could solve.**
2. **Explain the "stale closure" problem with `useEffect` and how to avoid it.**
3. **How would you optimize a `useContext` consumer to prevent unnecessary re-renders when only a small part of the context value changes?**
4. **Describe a scenario where `useImperativeHandle` would be useful. What are its potential downsides?**
5. **How does `useLayoutEffect` differ from `useEffect`? When would you choose one over the other?**

Rendering Behavior & React Internals:

6. **Explain the rules of Hooks. Why are they important, and what happens if you break them?**
7. **Describe how React handles events. Is it directly attaching event listeners to DOM elements?**
8. **What is the difference between `setState` in a class component and the setter function returned by `useState` in a functional component, especially regarding asynchronous updates?**
9. **Discuss the concept of "fiber architecture" in React. How does it enable concurrent rendering?**
10. **Explain how `ReactDOM.createPortal` works and its primary use cases.**

Design Patterns & Architecture:

11. **Compare and contrast the Compound Components pattern with the Render Props pattern. When would you use each?**
12. **What is the "presentational and container components" pattern? Is it still relevant with Hooks?**
13. **How do you approach designing a flexible and extensible component API?**
14. **Discuss the challenges and solutions for managing global state without a dedicated library (e.g., Redux).**
15. **How do you handle feature flags or A/B testing in a React application?**

Accessibility & Internationalization:

16. **What are ARIA attributes, and how do you use them to improve accessibility in React?**

17. How do you ensure keyboard navigation and focus management are robust in your React applications?
18. Discuss strategies for internationalization (i18n) in a React application, including handling translations, date/time formatting, and right-to-left (RTL) languages.

Security & Best Practices:

19. What are common security vulnerabilities in React applications (e.g., XSS) and how do you mitigate them?
20. How do you handle sensitive data (e.g., API keys) in a React application?
21. Discuss the importance of static analysis (e.g., ESLint) and code formatters (e.g., Prettier) in a React project.

Problem Solving & System Design:

22. Imagine you need to build a drag-and-drop interface. How would you approach this in React, considering performance and user experience?
23. How would you implement an undo/redo functionality for a complex form or editor in React?
24. You're tasked with rebuilding a legacy jQuery application in React. What are the key considerations and challenges you'd anticipate?
25. Describe a challenging bug you encountered in a React application and how you debugged and resolved it.