Senior React.js Interview Questions & Answers

Q: What are the differences between functional and class components? Why prefer functional now?

A: Class components used lifecycle methods, while functional components use hooks like useEffect and useState. Functional components are simpler, easier to test, and are the modern recommended approach.

Q: How does React’s reconciliation (Virtual DOM) work?

A: React diffs the new virtual DOM with the old one. Matching elements are updated, and keys help track changes in lists. This makes updates efficient. Developers can optimize with React.memo, useMemo, and useCallback.

Q: How would you handle state management in a large-scale React app?

A: Use local state (useState/useReducer) for UI, Context API for small global state, and tools like Redux Toolkit or Zustand for complex state. For server state, React Query is a strong choice.

Q: How do you optimize React performance?

A: Use React.memo, useCallback, and useMemo to prevent unnecessary re-renders. Apply code splitting with React.lazy, virtualize long lists, debounce expensive operations, and profile with React DevTools.

Q: Explain custom hooks. Can you give an example?

A: Custom hooks extract reusable stateful logic. Example: a useWindowSize hook tracks window dimensions with useState and useEffect.

Q: What is the difference between useEffect and useLayoutEffect?

A: useEffect runs asynchronously after paint, good for data fetching. useLayoutEffect runs synchronously before paint, useful for DOM measurements, but can hurt performance if overused.

Q: How would you structure a large React project?

A: Use a feature-based folder structure, separating components, hooks, and services by feature. Use shared directories for reusable UI components. Add lazy loading for routes and enforce boundaries between modules.

Q: How do you ensure code quality in a React project?

A: Use TypeScript for type safety, ESLint + Prettier for linting/formatting, Jest + React Testing Library for unit tests, Cypress for integration tests, and Storybook for UI documentation. Apply CI/CD pipelines for automation.

Q: What’s the difference between SSR, SSG, and CSR in React?

A: CSR renders on the client side. SSR generates HTML on the server (Next.js), improving SEO. SSG pre-renders at build time, best for blogs/docs. ISR combines benefits by regenerating pages incrementally.

Q: How do you secure a React application?

A: Store tokens in HttpOnly cookies instead of localStorage, sanitize inputs to prevent XSS, rely on React’s escaping, use backend validation for authorization, and apply secure headers with Helmet.