React Interview Q/As

Section 1: React Basics

Q1. What is React?
Answer : React is an open-source JavaScript library for building reusable UI components and single-page apps. It uses a virtual DOM and a component-based structure to efficiently update UIs.
💡 Tip: Always mention React = SPA + reusable components + virtual DOM.

Q2. What is JSX?
Answer : JSX is a syntax extension that looks like HTML inside JavaScript. It makes UI code easier to write/read, and is compiled into React.createElement calls under the hood.
💡 Tip: JSX is not required, but makes components more readable.

Q3. What is Virtual DOM and why is it faster?
Answer : Virtual DOM is an in-memory copy of the real DOM. React updates it first, compares (diffs) with the old version, then updates only changed parts in the real DOM.
💡 Tip: Always give example → "Virtual DOM reduces expensive DOM operations."

Q4. What are React components?
Answer : Components are reusable building blocks of React. They can be **functional** (use hooks) or **class-based** (older style with lifecycle methods). Functional components are preferred today.
💡 Tip: Say → "Modern React = Functional Components + Hooks."

Q5. Difference between props and state?
Answer : Props are read-only inputs passed from parent to child. State is internal and mutable, used to manage a component's own data.

Changing state triggers re-renders.
💡 Tip: Props = external, State = internal.

Section 2: Core Concepts

Q6. What are controlled vs uncontrolled components?
Answer : Controlled components have their form values managed by React state. Uncontrolled components rely on the DOM and are accessed via refs.
💡 Tip: Controlled = predictable, Uncontrolled = quick/simple.

Q7. Why use key in React lists?
Answer : Keys help React identify list items uniquely and efficiently update the DOM. Without keys, React may mis-update items.
💡 Tip: Avoid array index as keys in dynamic lists.

Q8. What are React Fragments?
Answer : Fragments let you return multiple elements without adding extra DOM nodes. Shorthand: <>...</>.
💡 Tip: Use fragments to avoid unnecessary <div> wrappers.

Q9. What is prop drilling and how to avoid it?
Answer : Prop drilling is passing props through many layers unnecessarily. It can be avoided with Context API or state management tools like Redux.
💡 Tip: Always mention Context API as the go-to fix.

Q10. What is reconciliation?
Answer : Reconciliation is React’s process of comparing virtual DOM trees and applying minimal updates to the real DOM.
💡 Tip: Keys improve reconciliation efficiency.

Section 3: React Hooks

Q11. What is useState hook?
Answer : useState allows functional components to use state. It returns a state variable and a function to update it.
💡 Tip: Mention functional update pattern for state based on previous value.

Q12. What is useEffect hook?
Answer : useEffect handles side effects like fetching data, subscriptions, or timers. Dependency array controls when it runs.
💡 Tip: Empty array = run once (on mount).

Q13. What is useContext hook?
Answer : useContext allows direct access to context values without prop drilling. It's used for global-ish data like themes or auth.
💡 Tip: Best for shared data → Theme/User/Auth.

Q14. What is useReducer hook?
Answer : useReducer is for complex state logic. It uses a reducer function and actions to update state, similar to Redux.
💡 Tip: Use in forms or complex component states.

Q15. Difference between useMemo and useCallback?
Answer : useMemo caches computed values, useCallback caches function references. Both optimize performance by preventing unnecessary recalculations/renders.
💡 Tip: useMemo = value, useCallback = function.

Section 4: Advanced React

Q16. What is useRef hook?
Answer : useRef provides a mutable object that persists across renders. Used for DOM access (e.g., focus an input) or storing values

without re-renders.
💡 Tip: Ref = mutable value, no re-render.

Q17. What are Higher-Order Components (HOCs)?
Answer : HOCs are functions that take a component and return a new one with extra functionality. Example: authentication, logging, or data fetching wrappers.
💡 Tip: Mention → "Hooks often replaced HOCs in modern React."

Q18. What are Pure Components / React.memo?
Answer : PureComponent or React.memo prevent re-renders if props/state haven't changed (shallow compare). They optimize performance.
💡 Tip: Use with care — only for heavy components.

Q19. What are the Rules of Hooks?
Answer : (1) Call hooks only at the top level of React functions, (2) Call hooks only in React components or custom hooks.
💡 Tip: Breaking rules causes React errors → memorize 2 rules.

Q20. Why should you not mutate state directly?
Answer : Direct mutation won't trigger re-render and can cause bugs. Always use setState or spread operators to update immutably.

💡 Tip: State must remain immutable.

Section 5: Routing & State Management

Q21. What is React Router?
Answer : React Router enables navigation in SPAs by mapping URLs to components without page reload. Uses <BrowserRouter>, <Route>, and <Link>.
💡 Tip: v6 uses <Routes> instead of <Switch>.

Q22. Difference between Context API vs Redux?
Answer : Context API shares data without prop drilling, simple for small apps. Redux is more structured with global store, actions, reducers → better for large apps.
💡 Tip: Context = small/global, Redux = complex/large.

Q23. What is conditional rendering in React?
Answer : Conditional rendering means showing components based on conditions (e.g., ternary operator, logical AND).
💡 Tip: Example → Show login vs logout button.

Q24. What is idempotency in APIs (React usage)?
Answer : Idempotent actions produce the same result no matter how many times they are repeated. In React, e.g., retrying an API request should not double-charge.
💡 Tip: Say → "Idempotency = safe retries."

Q25. What are controlled forms in React?
Answer : Controlled forms use state to manage form input values. State updates on every keystroke via onChange.
💡 Tip: Controlled forms enable validation easily.

Section 6: Performance & Deployment

Q26. How do you optimize React performance?
Answer : Use memoization (React.memo, useMemo, useCallback), code splitting with React.lazy, and list virtualization.
💡 Tip: Mention React DevTools Profiler for bottlenecks.

Q27. What is Server-Side Rendering (SSR)?
Answer : SSR renders components to HTML on the server, improving SEO and first load performance. Example: Next.js.
💡 Tip: SSR = better SEO + fast first paint.

Q28. What is lazy loading in React?
Answer : Lazy loading (code splitting) loads components only when needed using React.lazy and <Suspense>.
💡 Tip: Improves performance by reducing bundle size.

Q29. What are common mistakes React devs make?
Answer : Mutating state directly, using index as key, forgetting dependency arrays in useEffect, or overusing Context.
💡 Tip: Always show awareness of these pitfalls.

Q30. What are PropTypes in React?
Answer : PropTypes validate props at runtime, ensuring correct types. Example: PropTypes.string.isRequired.
💡 Tip: Today many use TypeScript instead of PropTypes.