React Detailed Guide: Concepts, Best Practices & Interview Q&A;

Why choose React?

React offers a component-based architecture, Virtual DOM for efficient rendering, and a huge ecosystem. It is declarative, making UI predictable and easier to debug. Best Practice: Keep components small and reusable. Interview Q: Why is React preferred over Angular? A: React is lightweight, flexible, and focuses on UI rendering, while Angular is a full-fledged framework.

Virtual DOM and its benefits

Virtual DOM is an in-memory representation of the real DOM. React updates the Virtual DOM first, then efficiently updates the real DOM using diffing. Benefit: Performance optimization. Interview Q: How does Virtual DOM improve performance? A: It reduces costly direct DOM manipulations.

JSX and its advantages

JSX allows writing HTML-like syntax in JavaScript, making code more readable and maintainable. Best Practice: Use JSX for clarity and leverage linting tools. Interview Q: Is JSX mandatory? A: No, but it simplifies UI development.

Reconciliation

React compares the new Virtual DOM with the previous one and updates only changed elements. This process is called reconciliation. Interview Q: What algorithm does React use for reconciliation? A: React uses a diffing algorithm optimized for tree structures.

State vs Props

State is mutable and local to a component; Props are immutable and passed from parent to child. Best Practice: Lift state up when multiple components need it. Interview Q: Can props be changed? A: No, props are read-only.

Hooks: useState & useEffect

useState manages local state; useEffect handles side effects like API calls. Best Practice: Always clean up subscriptions in useEffect. Interview Q: When does useEffect run? A: After render, optionally based on dependencies.

useMemo vs useCallback

useMemo memoizes values; useCallback memoizes functions. Best Practice: Use them for expensive computations or stable callbacks. Interview Q: When should you avoid them? A: Avoid overuse as they add complexity.

Code Splitting

Break code into chunks using React.lazy and Suspense for better performance. Interview Q: Why is code splitting important? A: It reduces initial load time.

Accessibility in React

Ensure semantic HTML, ARIA roles, and keyboard navigation. Best Practice: Test with screen readers. Interview Q: How do you make a button accessible? A: Use tag and proper ARIA attributes.

Unidirectional Data Flow

Data flows from parent to child, making state predictable. Interview Q: Why is this important? A: It simplifies debugging and state management.

Pure Components vs HOC

Pure Components prevent unnecessary re-renders; HOCs wrap components to add functionality. Interview Q: Give an example of HOC. A: withAuth(Component).

Security: CSRF & XSRF

Use CSRF tokens, validate requests, and avoid storing sensitive data in local storage. Interview Q: How do you prevent XSS in React? A: Use dangerouslySetInnerHTML cautiously and sanitize input.

useEffect Use Cases

Fetching data, subscriptions, DOM updates. Best Practice: Always clean up effects. Interview Q: How do you prevent infinite loops in useEffect? A: Use dependency arrays correctly.

React Optimization Techniques

Memoization, virtualization, code splitting, avoiding inline functions. Interview Q: How do you optimize large lists? A: Use react-window or react-virtualized.

Lazy Loading

Load components only when needed using React.lazy. Interview Q: How does lazy loading improve performance? A: It reduces initial bundle size.

Class vs Functional Components

Class uses lifecycle methods; Functional uses hooks. Interview Q: Which is preferred today? A: Functional components with hooks.

SEO in React

Use SSR or static generation, add meta tags dynamically. Interview Q: Why is SEO challenging in React? A: Because it's client-side rendered by default.

React Router

Handles client-side routing for SPAs. Interview Q: How do you implement dynamic routes? A: Use .

Context API

Provides global state without prop drilling. Interview Q: When to use Context vs Redux? A: Use Context for small apps, Redux for complex state.

Server-Side Rendering (SSR)

Renders React on server for better SEO and performance. Interview Q: Which library supports SSR? A: Next.js.

Error Boundaries

Catch runtime errors and prevent app crashes. Interview Q: Can hooks be used in error boundaries? A: No, only class components can be error boundaries.