React Developer Interview – Quick Reference Cheat Sheet

■ Core React Concepts

Q: What’s the difference between functional and class components?
A: Functional components use hooks and are simpler; class components use lifecycle methods. Functional components are preferred.

Q: How do props differ from state?
A: Props are external and immutable; state is internal and mutable.

Q: How do you pass data from a child to a parent?
A: By passing a callback function from parent to child and invoking it with data.

■■ Hooks

Q: What’s the dependency array used for in useEffect?
A: It defines when to re-run the effect; an empty array means run once.

Q: Difference between useCallback and useMemo?
A: useCallback memoizes a function; useMemo memoizes a computed value.

Q: Can hooks be conditional?
A: No. Hooks must be called in the same order every render.

■ Performance Optimization

Q: How do you optimize re-renders in lists?
A: Use keys correctly and memoize components.

Q: What’s React.lazy() and Suspense used for?
A: For code-splitting and lazy loading components with fallback UI.

■ Routing & State Management

Q: How do you handle dynamic routes?
A: Use useParams() to access route parameters.

Q: Context vs Redux vs Zustand?
A: Context for simple global data, Redux for predictable state, Zustand for lightweight state with hooks.

■ Practical React

Q: Controlled vs uncontrolled components?
A: Controlled components store form data in state; uncontrolled use refs.

Q: How do you cancel a fetch request on unmount?
A: Use AbortController.

■ Next.js

Q: Difference between server and client components?
A: Server components render on the server and have no client JS; client ones can use hooks.

Q: What’s ISR (Incremental Static Regeneration)?
A: Updates static pages after deployment with fresh data.

■ JavaScript Fundamentals

Q: Why is immutability important in React?
A: Because React relies on shallow comparisons to detect state changes.

Q: How to update deeply nested state safely?
A: Use spread operators or Immer.

■ Advanced & Behavioral

Q: How would you structure a large-scale React app?

A: Feature-based folders separating UI, logic, and state layers.

Q: How do you diagnose a slow React app?

A: Profile re-renders, check memoization, reduce bundle size.