1. Redux Interview Questions

1. **What is Redux?**
   ☞ A predictable state container for managing application state globally.
2. **What are the core principles of Redux?**
   ☞ Single source of truth, state is read-only, and changes via pure functions (reducers).
3. **What is an action in Redux?**
   ☞ A plain JS object with a type and optional payload — describes what to do.
4. **What is a reducer?**
   ☞ A pure function that returns the next state based on current state and action.
5. **What is the Redux store?**
   ☞ A centralized container that holds your app's entire state.
6. **How do you update state in Redux?**
   ☞ By dispatching actions using dispatch({ type: "ACTION_TYPE" }).
7. **What is middleware in Redux?**
   ☞ Functions like redux-thunk to handle async logic before reaching the reducer.
8. **How do you connect Redux to a React component?**
   ☞ Using useSelector() to read state, and useDispatch() to send actions.
9. **What is redux-thunk?**
   ☞ A middleware that lets you write async logic in action creators.
10. **Difference between Redux and Context API?**
    ☞ Redux is more scalable with tools/middleware; Context is simpler for small data.

2. React Router Interview Questions

11. **What is React Router?**
    ☞ A library to handle navigation and routing in React single-page applications.
12. **What is <BrowserRouter> used for?**
    ☞ It enables routing using the HTML5 History API.

13. **What is the use of <Routes> and <Route>?**
    ☞ <Routes> replaces <Switch> in React Router v6 to match and render <Route>s.
14. **How do you navigate programmatically?**
    ☞ Using useNavigate() hook from react-router-dom.
15. **What are dynamic routes?**
    ☞ Routes with parameters like /user/:id.

3. React Hooks Interview Questions

16. **What are Hooks in React?**
    ☞ Special functions that let functional components use state and lifecycle features.
17. **What is useState()?**
    ☞ A hook for adding state to functional components.
18. **What is useEffect() used for?**
    ☞ Runs side-effects like fetching data, updating DOM, etc.
19. **Difference between useEffect and useLayoutEffect?**
    ☞ useLayoutEffect runs before painting; useEffect runs after.
20. **What is useRef()?**
    ☞ Returns a mutable ref object — useful for DOM access or persisting values.
21. **What is useMemo() vs useCallback()?**
    ☞ useMemo memoizes values, useCallback memoizes functions.

4. JSX Interview Questions

22. **What is JSX?**
    ☞ A syntax extension that allows writing HTML inside JavaScript.
23. **Can browsers understand JSX directly?**
    ☞ No, JSX is transpiled to JS using Babel.
24. **How do you write inline styles in JSX?**
    ☞ As objects: style={{ color: 'red' }}.
25. **Can you return multiple elements from a component?**
    ☞ Yes, using fragments <> </>.

5. State Management Interview Questions

26. **What is state in React?**
    ☞ An object that holds dynamic data and controls component behavior.
27. **How do you manage local state?**
    ☞ Using useState() hook.
28. **How do you share state between components?**
    ☞ Lift state up to the nearest common ancestor.
29. **When should you use useReducer?**
    ☞ For complex state transitions or multiple related state values.
30. **What are common state management tools?**
    ☞ Redux, Context API, Zustand, MobX, Recoil.

6. Context API Interview Questions

31. **What is the Context API in React?**
    ☞ A way to pass data globally without prop drilling.
32. **How do you create and use context?**
    ☞ createContext(), then use Provider and useContext().
33. **What are good use cases for Context API?**
    ☞ Theme, user auth, language, global settings.
34. **Context vs Redux — which to choose?**
    ☞ Use Context for small-scale sharing, Redux for complex apps.
35. **How do you avoid unnecessary re-renders with Context?**
    ☞ Split context into smaller providers or use memoization.

7. Nested Routes Interview Questions

36. **What are nested routes?**
    ☞ Routes inside other routes — helps build layouts with child components.
37. **How to define nested routes in React Router v6?**
    ☞ Define child <Route>s inside parent <Route> using children or <Outlet>.
38. **What is <Outlet /> used for?**
    ☞ A placeholder in the parent component to render nested routes.
39. **How do you structure nested routes with layout?**
    ☞ Use a layout component that wraps the <Outlet /> and shared UI.
40. **Can nested routes access parent route params?**
    ☞ Yes, they can access params using useParams().

Bonus: Common Mixed Questions

41. **What is prop drilling and how to solve it?**
    ☞ Passing props deeply — solved by Context API or Redux.
42. **What is lazy loading in React?**
    ☞ Load components only when needed using React.lazy() + Suspense.
43. **What is the role of keys in React lists?**
    ☞ Helps identify which list items changed/removed/added.
44. **What is a controlled vs uncontrolled component?**
    ☞ Controlled: managed by React state; Uncontrolled: managed by DOM refs.
45. **How do you optimize React performance?**
    ☞ Memoization (React.memo, useMemo), lazy loading, avoiding prop drilling.
46. **What is an error boundary?**
    ☞ A class component that catches JS errors and shows fallback UI.
47. **What is hydration in SSR?**
    ☞ React attaches event listeners to static HTML rendered from the server.
48. **What is React StrictMode?**
    ☞ A wrapper to catch potential problems in development.
49. **Can hooks be used in class components?**
    ☞ ❌ No, hooks only work in functional components.
50. **What is React's default rendering behavior?**
    ☞ Re-renders component when state or props change.

React Architecture & Performance

51. What is reconciliation in React?
☞ It's the diffing algorithm React uses to update the virtual DOM efficiently.

52. How does React batch updates?
☞ React groups multiple state updates into a single render for performance (especially inside event handlers).

53. What is Concurrent Mode?
☞ An experimental mode that allows React to interrupt rendering and work on multiple tasks simultaneously.

54. What is a Fiber tree in React?
☞ A data structure used internally to track components and rendering progress.

55. How can you prevent unnecessary re-renders?
☞ Using React.memo, useMemo, useCallback, and keeping components pure.

Hooks – Advanced Use

56. What is useImperativeHandle() used for?
☞ Customizes what a ref exposes to the parent when using forwardRef.

57. What are custom hooks and why use them?
☞ Reusable hook functions (starting with use) to share logic across components.

58. Can you call hooks conditionally?
❌ No — Hooks must be called unconditionally and in the same order on every render.

59. What is hook dependency array and how does it work?
☞ It tells React when to re-run the hook — only if listed dependencies change.

60. How does useEffect cleanup work?
☞ By returning a function inside useEffect, you clean up resources (like timers, subscriptions) when component unmounts or before re-running.

Routing & SSR/CSR/ISR

61. What is the difference between CSR, SSR, SSG, and ISR?

Type	Description
CSR	React loads on client after JS loads (default SPA)
SSR	Server renders HTML on every request
SSG	HTML is pre-rendered at build time
ISR	Pages are statically built and revalidated in background

62. What is hydration in React SSR?
☞ Process where React attaches event handlers to static HTML rendered by the server.

63. What is code-splitting?
☞ Splitting JS bundles using React.lazy() to improve load time.

64. What is dynamic import in React?
☞ Loading modules/components only when needed using import().

65. How does Suspense work?
☞ It wraps lazy-loaded components and shows a fallback UI (like a spinner) until they load.

React Patterns & Architecture

66. What is a render prop?
☞ A function passed as a prop that returns JSX, giving control over rendering.

67. What are Higher Order Components (HOCs)?
☞ Functions that take a component and return an enhanced version of it.

68. What is the container vs presentational component pattern?
☞ Containers handle logic; presentational ones handle UI.

69. What is the provider pattern?
☞ A way to share global data via React Context (<Provider> at top level).

70. What are compound components?
☞ Grouped components that communicate implicitly via context (e.g., <Tabs>, <TabPanel>).

Edge Cases, Debugging & Tools

71. What is React Profiler?
☞ A tool to measure render time and identify performance bottlenecks.

72. How do you debug a memory leak in React?
☞ Use cleanup in useEffect, check for lingering refs or subscriptions.

73. What causes stale closures in hooks?
☞ When a callback refers to outdated state or props due to closure scope.

74. What is the difference between strict equality and referential equality in React?
☞ React uses **referential equality**, so even if values are same, new object/function triggers re-render.

75. What is tree-shaking and how does React support it?
☞ Eliminates unused code in bundles — React supports it via ES module exports.