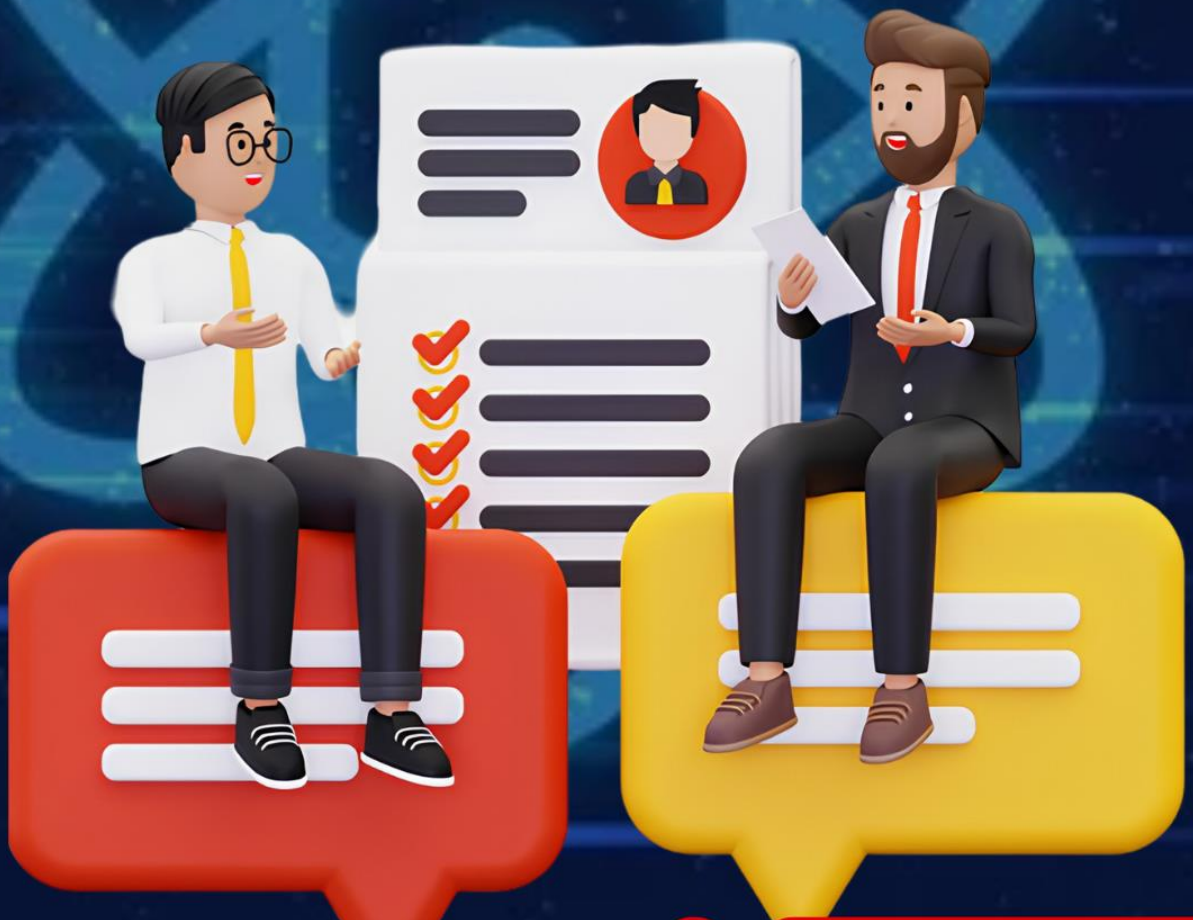
React JS

Interview Questions & Answers



a1training167@gmail.com

www.a1training.in

8368979712

6380486914

C-167,Omicron 1,6%Abadi,Greater Noida / Earthcon Sanakriti, Noida Extension

React Interview Q/A

1. React Basics

1. **What is React?**
    - **Answer:** React is a JavaScript library for building user interfaces, primarily single-page applications, by enabling developers to create reusable UI components.
2. **Explain the concept of Virtual DOM in React.**
    - **Answer:** The Virtual DOM is a lightweight representation of the actual DOM. React uses it to track changes in the DOM by comparing it with a virtual copy, enabling efficient updates and re-rendering only the components that have changed.
3. **What is JSX?**
    - **Answer:** JSX is a syntax extension for JavaScript, similar to HTML, used in React to describe the UI structure. JSX enables developers to write HTML-like syntax within JavaScript code.
4. **Why do we use keys in React?**
    - **Answer:** Keys help React identify which items have changed, been added, or removed in a list. Using keys enhances the performance and helps maintain the correct order of elements.
5. **What is the purpose of using React.Fragment?**
    - **Answer:** React.Fragment is used to group multiple elements without adding extra nodes to the DOM. It helps avoid unnecessary div wrappers.

---

2. React Components

6. **What is a React component?**
    - **Answer:** A React component is a reusable piece of code that returns a React element (JSX) to define a part of the user interface. Components can be functional or class-based.
7. **Explain the difference between functional and class components.**
    - **Answer:** Functional components are simpler and use functions to define components, while class components use ES6 classes and require render() to return JSX. Functional components support hooks, while class components do not.

8. **What is props in React?**
    - **Answer:** Props (short for properties) are a way to pass data from a parent component to a child component, allowing dynamic content rendering.
9. **How do you create a component in React?**
    - **Answer:** A functional component is created as a JavaScript function that returns JSX, while a class component extends React.Component and includes a render() method returning JSX.
10. **What are pure components?**
    - **Answer:** Pure components are components that do not re-render if their props or state remain unchanged. React uses shouldComponentUpdate() internally to implement this.

---

3. State and Lifecycle

11. **What is state in React?**
    - **Answer:** State is a JavaScript object that holds data which may change over time and affects the component's rendering. It is managed within the component and updated with setState().
12. **Explain the component lifecycle in React.**
    - **Answer:** React lifecycle methods can be divided into mounting (component is created), updating (component state or props change), and unmounting (component is removed from the DOM).
13. **How does setState work in React?**
    - **Answer:** setState is used to update the state object and triggers a re-render of the component. It is asynchronous and can accept a function or object to update the state.
14. **What are lifecycle methods in class components?**
    - **Answer:** Lifecycle methods include componentDidMount, componentDidUpdate, and componentWillUnmount, which allow handling actions at different stages of the component's existence.
15. **What is the purpose of componentDidMount?**
    - **Answer:** componentDidMount is used to execute code after a component is inserted into the DOM, such as making API calls or setting up subscriptions.

---

4. React Hooks

16. **What are hooks in React?**
    - **Answer:** Hooks are functions that let you use state and other React features in functional components, introduced in React 16.8.
17. **What is useState in React?**
    - **Answer:** useState is a hook that allows adding state to functional components. It returns a state variable and a function to update that variable.
18. **Explain the useEffect hook.**
    - **Answer:** useEffect is a hook for handling side effects in functional components, such as data fetching, subscriptions, or DOM manipulation, and runs after the component renders.
19. **What does useContext do?**
    - **Answer:** useContext is a hook that allows consuming a context value directly in functional components, avoiding the need to wrap components with Consumer components.
20. **How does the useReducer hook work?**
    - **Answer:** useReducer is a hook similar to useState but better suited for complex state logic. It takes a reducer function and an initial state and returns the current state and dispatch function.

---

5. Advanced Concepts

21. **What is HOC (Higher-Order Component) in React?**
    - **Answer:** A Higher-Order Component is a function that takes a component and returns a new component, enhancing it with additional functionality.
22. **Explain Context API in React.**
    - **Answer:** The Context API allows data sharing across the component tree without passing props at every level, ideal for global data like themes or user authentication.
23. **What is React Router?**
    - **Answer:** React Router is a library for handling client-side routing, allowing navigation between different views without reloading the page.
24. **What is Redux in React?**
    - **Answer:** Redux is a state management library used to manage the global state of an application. It helps in handling complex state management across components.

25. **Explain the concept of Redux middleware.**
    - **Answer:** Redux middleware allows intercepting actions before they reach the reducer, enabling side effects like async operations (e.g., API calls) using libraries like Thunk or Saga.

---

6. Performance Optimization

26. **How can you optimize performance in a React application?**
    - **Answer:** Techniques include using React.memo for component memoization, splitting code with dynamic imports, lazy loading, and avoiding unnecessary re-renders.
27. **What is React.memo?**
    - **Answer:** React.memo is a higher-order component that prevents a component from re-rendering if its props haven't changed.
28. **What are lazy loading and React.lazy?**
    - **Answer:** Lazy loading defers loading a component until it's needed, and React.lazy allows components to be loaded dynamically to improve performance.
29. **What is code splitting in React?**
    - **Answer:** Code splitting is a technique to break down a large app into smaller chunks, loading only the necessary code for each part, often using React.lazy and Suspense.
30. **What is React Suspense?**
    - **Answer:** Suspense allows components to "wait" for something (e.g., a lazy-loaded component) before rendering, showing a fallback (e.g., loading spinner) while waiting.

7. React Advanced Patterns

31. **What is render props in React?**
    - **Answer:** Render props is a pattern for sharing code between components by passing a function as a prop, which allows dynamic rendering based on shared logic.
32. **How do portals work in React?**
    - **Answer:** Portals enable rendering a component outside the parent DOM hierarchy, often used for modals or overlays, by specifying a target DOM node.

33. **What is the significance of error boundaries in React?**
    - **Answer:** Error boundaries catch JavaScript errors within components and prevent them from breaking the entire application, displaying a fallback UI instead.
34. **What is the difference between controlled and uncontrolled components?**
    - **Answer:** Controlled components have state managed by React, whereas uncontrolled components maintain their state in the DOM, typically using refs.
35. **What are forward refs in React?**
    - **Answer:** Forward refs enable passing refs through a component to access a child component's DOM node, useful in higher-order components or wrapper elements.
36. **How does useRef work in React?**
    - **Answer:** useRef creates a mutable reference object that persists across renders, commonly used to access DOM elements or store a value that doesn't trigger a re-render.
37. **What is useCallback in React?**
    - **Answer:** useCallback is a hook that memoizes a function so it isn't re-created on every render, useful for optimizing child components that depend on function props.
38. **Explain useMemo and its use cases.**
    - **Answer:** useMemo memoizes a value to avoid recalculating it on every render, used for expensive calculations or objects that only change when dependencies change.
39. **What is the useImperativeHandle hook?**
    - **Answer:** useImperativeHandle customizes the instance value exposed to parent components when using forwardRef, often for imperative actions on child components.
40. **Explain prop drilling and how to avoid it.**
    - **Answer:** Prop drilling occurs when props are passed through multiple components to reach a deeply nested component. It can be avoided with the Context API or state management libraries.

---

8. State Management

41. **How does Redux work in React?**

- **Answer:** Redux manages global state by creating a single source of truth, where the store holds state, and reducers handle actions to update the state immutably.

42. **What is an action in Redux?**
    - **Answer:** Actions are plain JavaScript objects that describe events and contain a type and optional payload, which reducers use to update the state.

43. **Explain reducers in Redux.**
    - **Answer:** Reducers are pure functions that determine how the application state changes in response to an action, taking the current state and an action as arguments.

44. **What is Redux Thunk?**
    - **Answer:** Redux Thunk is a middleware that allows action creators to return functions (typically for async operations) instead of plain action objects.

45. **How does Redux Toolkit simplify Redux?**
    - **Answer:** Redux Toolkit provides simplified APIs and best practices for configuring Redux, such as createSlice, which automatically generates actions and reducers.

46. **What is the purpose of selectors in Redux?**
    - **Answer:** Selectors retrieve specific data from the Redux state, making code more modular and optimizing performance by avoiding unnecessary re-renders.

47. **Explain the use of Recoil in React.**
    - **Answer:** Recoil is a state management library for managing global and shared state, offering atomic units of state and a simple API designed for React.

48. **What is MobX and how does it differ from Redux?**
    - **Answer:** MobX is a reactive state management library that uses observables to automatically track dependencies, differing from Redux's immutable and strict update approach.

49. **What is the role of the Provider component in Redux?**
    - **Answer:** The Provider component wraps the app and makes the Redux store available to all components, enabling them to connect to the store and access state.

50. **What is context-based state management, and how does it differ from Redux?**

- **Answer:** Context-based state management uses React's Context API to manage state across components without needing an external library, ideal for simpler use cases.

---

9. Testing React Applications

51. **How do you test React components?**
    - **Answer:** React components can be tested using libraries like Jest for unit tests and React Testing Library for DOM-based testing, ensuring components render and function as expected.
52. **What is Jest in React?**
    - **Answer:** Jest is a JavaScript testing framework commonly used with React to test components, functions, and integration, offering features like mocking and snapshot testing.
53. **What is snapshot testing in React?**
    - **Answer:** Snapshot testing captures a rendered component's structure and compares it against future renders, alerting of unexpected changes to the component's UI.
54. **How does React Testing Library differ from Enzyme?**
    - **Answer:** React Testing Library focuses on testing component behavior from a user perspective, while Enzyme provides more flexibility for shallow and deep rendering components.
55. **What is shallow rendering in Enzyme?**
    - **Answer:** Shallow rendering only renders a component's first level, not its child components, useful for isolating and testing a component without its dependencies.
56. **How do you test hooks in React?**
    - **Answer:** Hooks can be tested using custom render functions in React Testing Library or by separating logic into pure functions for easier unit testing.
57. **What is act() in React Testing Library?**
    - **Answer:** act() is used to wrap code that triggers changes, ensuring that updates and re-renders are completed before assertions, preventing state update warnings.
58. **How do you test async functions in React?**
    - **Answer:** Async functions are tested by using async and await to handle promises, and by mocking network requests with libraries like Jest and MSW.

59. **What is mocking in testing, and how do you mock APIs?**
    - **Answer:** Mocking simulates dependencies or external services. APIs can be mocked with libraries like Jest, Sinon, or MSW to test functions independently of external systems.
60. **How do you test user interactions with React Testing Library?**
    - **Answer:** Use functions like fireEvent or userEvent to simulate clicks, form submissions, and other user interactions, and assert the expected changes in the component.

---

10. React Styling

61. **What are CSS Modules in React?**
    - **Answer:** CSS Modules allow scoping CSS to specific components, preventing styles from affecting others by generating unique class names.
62. **Explain styled-components in React.**
    - **Answer:** styled-components is a library for styling React components using tagged template literals, enabling CSS within JavaScript and supporting dynamic styling.
63. **What is Emotion in React?**
    - **Answer:** Emotion is a CSS-in-JS library that enables styling React components with both styled-components syntax and css-prop, allowing for scoped and dynamic styles.
64. **How does inline styling work in React?**
    - **Answer:** Inline styling is done by passing a JavaScript object with camelCased property names to the style prop, suitable for dynamic or unique styling needs.
65. **What are the benefits of CSS-in-JS libraries?**
    - **Answer:** CSS-in-JS libraries provide scoped, dynamic, and conditionally applied styles, improved maintainability, and automatic CSS extraction for optimized performance.
66. **How do you implement theming in React?**
    - **Answer:** Theming can be implemented by defining a theme object and passing it to a ThemeProvider (such as from styled-components), making styles globally available.
67. **What are global styles in React?**

- **Answer:** Global styles apply to the entire application, typically included in index.css, App.css, or using a global style component with CSS-in-JS libraries.

68. **Explain CSS BEM and its use in React.**
    - **Answer:** BEM (Block, Element, Modifier) is a CSS naming convention that helps in writing modular, reusable, and understandable CSS, useful for managing large codebases.
69. **What is Tailwind CSS and how can it be used in React?**
    - **Answer:** Tailwind CSS is a utility-first CSS framework offering pre-designed classes for styling, which can be used in React by applying classes directly to JSX elements.
70. **How does SCSS differ from regular CSS, and how can you use it in React?**
    - **Answer:** SCSS offers enhanced features like variables, nesting, and mixins. It's added to React by installing a preprocessor like node-sass and renaming .css files to .scss.

11. React Best Practices

71. **What is prop validation in React, and how do you implement it?**
    - **Answer:** Prop validation ensures that components receive correct types and required props using PropTypes. It catches errors by specifying the type and required properties for each prop.
72. **How do you handle forms in React?**
    - **Answer:** Forms can be managed with controlled components (using state) or libraries like Formik or React Hook Form, which simplify form handling and validation.
73. **What are some performance optimization tips in React?**
    - **Answer:** Use techniques like memoization with React.memo, lazy loading, avoiding anonymous functions in render, using useCallback and useMemo, and leveraging efficient list rendering.
74. **What are the key accessibility best practices in React?**
    - **Answer:** Accessibility best practices include using semantic HTML elements, aria-* attributes, keyboard navigability, focus management, and testing with tools like Axe.
75. **Explain the concept of immutability in React.**
    - **Answer:** Immutability means not modifying existing data directly but instead creating a copy with changes. It ensures efficient re-rendering by allowing React to detect state changes.

76. **How do you structure a large React application?**
    - **Answer:** A large React application can be structured with feature-based folders, using atomic design principles or the “ducks” pattern for Redux files, and keeping reusable components in a shared folder.
77. **What is code-splitting, and why is it important?**
    - **Answer:** Code-splitting breaks down the application into smaller bundles, loaded only when needed, reducing initial load time and improving performance.
78. **How do you handle errors gracefully in React?**
    - **Answer:** Errors can be managed with error boundaries, try-catch blocks, and fallback UI for API failures. Logging errors and displaying user-friendly messages improve user experience.
79. **Why should you avoid using indexes as keys in React lists?**
    - **Answer:** Indexes as keys can cause unexpected behavior when the list order changes, as React may not recognize the correct elements to update, affecting component re-renders.
80. **How do you prevent excessive re-renders in React?**
    - **Answer:** Use React.memo, useCallback, useMemo, and avoid unnecessary state updates or prop changes that can trigger re-renders.

---

12. React Libraries and Ecosystem

81. **What is Formik, and why would you use it?**
    - **Answer:** Formik is a library for handling form state, validation, and submission in React, simplifying complex form management with validation schemas and hooks.
82. **What is React Hook Form, and how does it differ from Formik?**
    - **Answer:** React Hook Form is a lightweight library that manages form state and validation with hooks, offering better performance than Formik by minimizing re-renders.
83. **What is the purpose of Next.js in React development?**
    - **Answer:** Next.js is a React framework for server-side rendering (SSR) and static site generation (SSG), providing SEO benefits and improving load times for better user experience.
84. **What is Gatsby, and when would you use it?**

- **Answer:** Gatsby is a React-based framework optimized for building static websites. It's useful for content-heavy sites, offering fast performance through pre-rendered static files.

85. **What is React Query, and why is it useful?**
    - **Answer:** React Query is a data-fetching library for managing server state, handling caching, synchronization, and updates, making API data management easier and reducing boilerplate.
86. **Explain the purpose of Storybook in React.**
    - **Answer:** Storybook is a tool for building and testing UI components in isolation, allowing developers to document, test, and showcase reusable components outside of the main app.
87. **What is React Spring, and how is it used?**
    - **Answer:** React Spring is an animation library that uses physics-based animations for creating fluid transitions, enabling animations through hooks and declarative configurations.
88. **How does React Helmet work?**
    - **Answer:** React Helmet manages the document head, allowing developers to dynamically set meta tags, titles, and other head properties, improving SEO for single-page applications.
89. **What is Axios, and why use it in React?**
    - **Answer:** Axios is a popular HTTP client for making API requests, offering simpler syntax than fetch, with built-in support for features like interceptors and request cancellation.
90. **What is Firebase, and how can it be integrated with React?**
    - **Answer:** Firebase is a platform offering backend services like real-time databases, authentication, and hosting. It can be integrated to provide serverless functionality within React apps.

---

13. React with TypeScript

91. **How do you add TypeScript to a React project?**
    - **Answer:** Add TypeScript by installing typescript and the @types/react packages, renaming files from .js to .tsx, and configuring tsconfig.json.
92. **How do you type props in a functional component with TypeScript?**
    - **Answer:** Define an interface for props and use it in the component's function signature: const Component: React.FC<PropsType> = (props) => {...}.

93. **What is the useReducer typing strategy in TypeScript?**
    - **Answer:** Define the state and action types, then pass these types to useReducer: useReducer<ReducerType<StateType, ActionType>>(reducer, initialState).

94. **How do you type custom hooks with TypeScript?**
    - **Answer:** Define the return type of the hook using TypeScript types or interfaces and annotate the hook function's return value.

95. **How do you type useRef in TypeScript?**
    - **Answer:** Define the ref's type as useRef<HTMLElement | null>(null), allowing the ref to reference an HTML element or remain null before assignment.

---

14. React Developer Tools

96. **What is React DevTools?**
    - **Answer:** React DevTools is a browser extension that allows developers to inspect and debug the component tree, view state, props, and hooks, and track performance.

97. **How can you analyze component performance in React?**
    - **Answer:** Use React DevTools' "Profiler" tab to identify components that re-render frequently and optimize them, which helps in detecting performance bottlenecks.

98. **What is the purpose of the "Profiler" in React DevTools?**
    - **Answer:** The Profiler records the rendering performance of components, showing how long components take to render and highlighting potential areas for optimization.

---

15. General React Questions

99. **How does React compare with other libraries like Angular or Vue?**
    - **Answer:** React is focused on the UI layer, using a component-based architecture and virtual DOM. Angular is a complete framework with two-way binding and services, while Vue is more similar to React but offers a simpler learning curve.

100. **What are some of the biggest changes in recent versions of React?** - **Answer:** Recent updates include Concurrent Mode, Suspense for data fetching, improved performance optimizations, new hooks, and a more powerful Context API.