Here's a comprehensive list of 200 basic to intermediate-level React.js interview questions that cover a broad range of React concepts, from foundational to practical knowledge.

**Basic React Questions**

1. What is React?
2. What is JSX in React?
3. Explain the virtual DOM in React.
4. What are components in React?
5. What is the difference between a class component and a functional component?
6. What is the role of render() in React?
7. What is the purpose of key in React lists?
8. How do you create a new React application?
9. What are props in React?
10. What are state and props in React, and what’s the difference between them?
11. What is the purpose of this in a class component?
12. How do you pass data from a parent component to a child component in React?
13. How do you update state in a React component?
14. What is the useState hook?
15. What is the difference between useState and this.setState()?
16. How do you handle events in React?
17. What is an event handler in React?
18. How can you bind event handlers in React class components?
19. What is the useEffect hook used for?
20. How does React handle form elements like input, textarea, and select?
21. How do you handle form submission in React?
22. How do you conditionally render a component in React?
23. What is the difference between == and === in React (JavaScript)?
24. How do you prevent a form from refreshing the page on submission?
25. What is React's default behavior for form elements when they are not controlled?
26. What is the componentDidMount() lifecycle method?
27. What is componentWillUnmount() in React used for?
28. What are componentDidUpdate() and shouldComponentUpdate()?
29. What is the role of super() in class components?
30. Can you explain React's render() method?
31. What is a functional component in React?
32. What is a class component in React?
33. How do you pass default values to props in React?
34. What is **PropTypes** in React?
35. What is React's **Children API**?
36. What is the purpose of the key prop in lists of React components?
37. What is the dangerouslySetInnerHTML property used for in React?
38. How do you handle a dynamic className in React?
39. What is **React.Fragment** and how is it useful?
40. What is the purpose of **React.memo()**?
41. How can you optimize performance in a React app?
42. What are **stateless** and **stateful** components?
43. Can you explain the concept of **component composition** in React?
44. What is a **controlled component** in React?
45. What is an **uncontrolled component** in React?
46. How does **React Hooks** simplify function components?
47. What is the **useRef** hook used for?
48. What is **useEffect** hook and how does it differ from componentDidMount and componentWillUnmount?
49. What is **React Router** and how is it used in single-page applications?
50. What is the difference between **React Router v5** and **React Router v6**?
51. What is the useContext hook used for?
52. What are **Higher-Order Components (HOCs)**?
53. How do you create a **Higher-Order Component (HOC)**?
54. What is the purpose of **React.StrictMode**?
55. What is **React.lazy()** used for?
56. How do you use **React Suspense** for lazy loading?
57. What are **React portals**?
58. What is **Context API** in React?
59. How do you implement **global state management** with React's Context API?
60. What is **Redux** in React and why would you use it?
61. What is the difference between **Redux** and **React Context API**?
62. What are actions, reducers, and stores in **Redux**?
63. How do you integrate **Redux** with React?
64. What are **dispatch** and **mapDispatchToProps** in Redux?
65. What is **connect()** in Redux?
66. How does **React** update the UI when state changes?
67. How does **React’s Virtual DOM** work internally?
68. What are **React keys** and why are they important in lists?
69. What are **refs** in React and how do you use them?
70. How do you access DOM elements with **React refs**?
71. What is the difference between **React.Component** and **React.PureComponent**?
72. How do you optimize **list rendering** in React?
73. What is **memoization** and how can it be used in React?
74. What is **shallow rendering** in React testing?
75. How do you test React components?
76. How do you create **unit tests** for React components?
77. What is the **React Testing Library**?
78. How do you test **hooks** in React?
79. How do you simulate **user interactions** in React tests?
80. What are **form validation** techniques in React?
81. What are **React event handlers** and how do they differ from regular JavaScript event handlers?
82. What is **react-bootstrap**?
83. How do you handle **custom events** in React?
84. What is **state lifting** in React?
85. How do you handle **asynchronous operations** in React (e.g., fetching data from APIs)?
86. What are **side effects** in React, and how do you handle them?
87. How can you implement **dynamic routing** in React?
88. How can you implement **nested routes** in React Router?
89. What is **React DevTools** and how does it help in debugging React applications?
90. What are **React performance optimization techniques**?
91. How does **React's reconciliation algorithm** work internally?
92. What are **Pure Components** and how do they improve performance in React?
93. What is **componentDidCatch** and how does it relate to error boundaries in React?
94. What is **React’s state batching** and how does it improve performance?
95. How do you implement **local storage** in React?
96. How do you implement **session storage** in React?
97. How do you implement **cookies** in React?
98. What is **code splitting** in React and how do you implement it?
99. What is **React's Fiber architecture**?
100. What is **JSX** and how is it different from **HTML**?

**Intermediate React Questions**

1. How does **React’s Context API** differ from Redux for state management?
2. How do you handle **deeply nested state** in React?
3. What is **useReducer** and when would you use it instead of useState?
4. How do you use **React Router v6** features like **useRoutes**?
5. How does **React's event delegation** work?
6. What is **React's synthetic event system**?
7. How do you handle **dynamic forms** in React?
8. How can you **conditionally render** a component in React based on its state?
9. How can you **implement pagination** in React?
10. How does **React** handle **performance optimization** with **shouldComponentUpdate**?
11. How do you prevent **unnecessary re-renders** in React functional components?
12. What are **render props** in React, and how do they work?
13. What is the purpose of **useMemo** and **useCallback** hooks in React?
14. How do you implement **context-based theming** in React?
15. How would you implement **infinite scroll** in React?
16. What is **lazy loading** in React, and how does it work?
17. How do you implement **user authentication** and **authorization** in a React app?
18. How do you implement **role-based access control (RBAC)** in React?
19. What is **React's Error Boundaries**, and how do you use them to catch rendering errors?
20. What is **React’s StrictMode**, and how does it help you debug your application?
21. How do you manage **cross-component communication** in React?
22. What are **props.children** in React?
23. How does **React.memo** improve performance in functional components?
24. How does **React's Hooks API** improve the use of state and lifecycle in functional components?
25. How do you manage **global state** without Redux in React?
26. How do you handle **asynchronous state updates** in React?
27. What is the **React life cycle** for a class-based component from creation to destruction?
28. What is **React.Fragment**, and how do you use it?
29. What is **reconciliation** in React?
30. How do you implement **real-time functionality** (e.g., chat or notifications) in React?
31. What is the **React Suspense** API and how does it enable **lazy loading** and asynchronous rendering?
32. How do you **authenticate with third-party providers** (OAuth) in React?
33. How can you implement **drag-and-drop functionality** in React?
34. How do you integrate **third-party libraries** in React apps?
35. How can you create a **global state** with **React Context API** and **useReducer**?
36. How would you implement a **multi-step form** in React?
37. What is **useLayoutEffect** and how does it differ from **useEffect**?
38. How do you create a **custom hook** for handling forms in React?
39. How can you implement **global error handling** in React using Error Boundaries?
40. What are **Dynamic Imports** in React and how do they impact app performance?
41. How does **React’s Context API** handle component re-renders in a deeply nested component tree?
42. How do you pass **functions as props** in React?
43. How do you implement **localization** and **internationalization (i18n)** in React?
44. What is **useCallback** and how does it optimize performance in React components?
45. What is **memoization**, and why is it important in React?
46. How do you pass **defaultProps** in a functional component in React?
47. How does **Redux DevTools** work with React apps?
48. How do you handle **side effects** (e.g., fetching data) in React with hooks?
49. How do you handle **authentication tokens** in a React app?
50. What are **hooks** for managing the lifecycle methods in functional components?