Interview Questions

Here's an extensive list of 300 beginner-level React.js interview questions. This comprehensive set will cover various aspects of React, including its concepts, components, hooks, performance optimization, state management, and more.

General React Concepts

1. What is React?
2. Why is React popular for web development?
3. What are the main features of React?
4. What is a component in React?
5. What are the types of components in React?
6. What is the difference between functional and class components?
7. What is the component lifecycle in React?
8. What are props in React?
9. How do you pass props to a component?
10. What is state in React?
11. How is state different from props?
12. How do you manage state in a functional component?
13. Can you explain the concept of lifting state up?
14. What is JSX?
15. Why do we use JSX in React?
16. What are keys in React?
17. Why are keys important in lists?
18. What is the virtual DOM?
19. How does React update the virtual DOM?
20. What is the purpose of ReactDOM.render()?

Component Structure and Composition

21. How do you create a React component?
22. What is the difference between props.children and props?
23. How can you reuse a component?
24. What is a higher-order component (HOC)?
25. What are render props in React?
26. How can you compose components in React?
27. What are default props in React?
28. How do you set default props for a component?
29. What are stateless functional components?

30. How do you create a stateless component?

React Hooks

31. What are React hooks?
32. How do you use the useState hook?
33. What is the useEffect hook used for?
34. Can you explain the rules of hooks?
35. What is the useContext hook?
36. What is the useRef hook?
37. How can you create a custom hook in React?
38. What does the useReducer hook do?
39. How can you manage form state using hooks?
40. How does useMemo help with performance?

Event Handling

41. How do you handle events in React?
42. What is the difference between synthetic events and native events?
43. How do you pass arguments to an event handler?
44. What is event delegation in React?
45. How can you prevent default behavior of an event?

Conditional Rendering and Lists

46. How do you implement conditional rendering?
47. What is the difference between using && and the ternary operator for conditional rendering?
48. How do you render a list of items in React?
49. What is the significance of the key prop when rendering lists?
50. How do you update an array in state?

Styling in React

51. How can you style a React component?
52. What are CSS modules?
53. How do styled-components work?
54. How do you conditionally apply classes in React?
55. What is the difference between inline styles and CSS stylesheets?

Forms and Inputs

56. How do you handle form submissions in React?
57. What is the difference between controlled and uncontrolled components?

58. How do you implement validation for a form?
59. What is the purpose of the onChange event in input elements?
60. How can you reset a form in React?

Performance Optimization

61. What are some ways to optimize performance in React?
62. What is code splitting?
63. How can you implement lazy loading in React?
64. What is React.memo?
65. How can you avoid unnecessary re-renders in React?

Routing

66. What is React Router?
67. How do you implement routing in a React application?
68. What are the different types of routing in React Router?
69. How do you pass props to a route in React Router?
70. What is a route guard?

State Management

71. What is Redux?
72. How does Redux work with React?
73. What are the core principles of Redux?
74. How do you connect a React component to Redux?
75. What is the purpose of middleware in Redux?

Context API

76. What is the Context API?
77. How do you create a context in React?
78. How do you consume context in a component?
79. What are the benefits of using the Context API?
80. How do you provide context to a component tree?

Testing

81. How do you test a React component?
82. What is Jest?
83. What is React Testing Library?
84. How can you simulate events in testing?
85. What are snapshot tests?

Interview Questions

Common Problems and Debugging

86. What are some common issues you might face in a React app?
87. How do you handle errors in a React application?
88. What are Error Boundaries?
89. How can you log errors in React?
90. What tools can you use to debug a React application?

Advanced Concepts

91. What is server-side rendering (SSR)?
92. What is static site generation (SSG)?
93. How does hydration work in React?
94. What is the reconciliation algorithm?
95. How do you implement dynamic imports in React?

Miscellaneous

96. What are fragments in React?
97. How can you use React with TypeScript?
98. What is the significance of React.StrictMode?
99. How do you handle side effects in React?
100. What are render props?

Hooks Deep Dive

101. What are the performance implications of using useEffect?
102. How can you avoid infinite loops in useEffect?
103. What is the purpose of the cleanup function in useEffect?
104. How do you share state between components using hooks?
105. Can you nest hooks within a component?

Performance Optimization Continued

106. What is the use of React.lazy?
107. How does useCallback improve performance?
108. How can you profile a React application for performance?
109. What is the use of React.Suspense?
110. How do you manage large component trees efficiently?

Component Communication

111. How do parent and child components communicate in React?
112. What are callback functions, and how are they used in React?

113. How can you manage sibling component communication?
114. What are the benefits of using prop drilling?
115. When would you choose to use context over props?

Component State Management

116. What is local component state?
117. How do you manage state for complex forms in React?
118. How do you update nested state in React?
119. What is the purpose of setState in class components?
120. How can you reset state in a functional component?

Styling Techniques

121. What are the pros and cons of CSS-in-JS?
122. How can you implement responsive design in React?
123. How do you use CSS preprocessors like SASS or LESS in React?
124. What are global styles, and how can you implement them in React?
125. How do you manage themes in a React application?

Working with APIs

126. How do you fetch data from an API in React?
127. What is the purpose of async/await in data fetching?
128. How do you handle loading states while fetching data?
129. What are some common patterns for error handling in API calls?
130. How do you implement pagination with API data?

Managing Dependencies

131. How do you manage dependencies in a React project?
132. What is the purpose of package.json?
133. How do you update dependencies in a React project?
134. How can you add external libraries to a React application?
135. What are peer dependencies in React?

Accessibility

136. What are ARIA roles, and why are they important in React?
137. How do you ensure accessibility in your React applications?
138. What are some best practices for keyboard navigation in React?
139. How do you implement focus management in React?
140. What is the purpose of the alt attribute for images?

Interview Questions

Best Practices

141. What are some best practices for structuring a React application?
142. How do you ensure code readability in React components?
143. What are some common anti-patterns in React?
144. How can you ensure that your components are reusable?
145. What are the benefits of using TypeScript with React?

Community and Ecosystem

146. What are some popular libraries in the React ecosystem?
147. How do you stay updated with the latest developments in React?
148. What are some common React developer tools?
149. What is the purpose of create-react-app?
150. What are some popular UI libraries for React?

Project Structure

151. How would you organize a large React application?
152. What is the significance of the src folder in a React project?
153. How do you manage assets in a React application?
154. What is the role of index.js in a React app?
155. How do you separate concerns in a React application?

Handling Asynchronous Operations

156. How do you handle multiple asynchronous operations in React?
157. What are promises, and how do they work in React?
158. How can you implement retries for failed API requests?
159. How do you cancel a network request in React?
160. What is the significance of using Promise.all()?

User Experience

161. How do you enhance user experience in React applications?
162. What is the purpose of loading spinners or skeleton screens?
163. How do you implement tooltips in React?
164. How do you handle user notifications in a React app?
165. What are some common user interface patterns in React?

Working with Data

166. What is the purpose of fetch() in data retrieval?
167. How do you handle date and time in React applications?

168. What are some libraries for managing dates in JavaScript?
169. How can you implement sorting functionality in a React app?
170. What are some common data visualization libraries for React?

Routing Deep Dive

171. How do you handle nested routes in React Router?
172. What are route parameters, and how are they used?
173. How do you implement redirects in React Router?
174. What is a 404 page, and how do you handle it in React?
175. How can you manage authentication in React Router?

Application State Management

176. What is the difference between local and global state?
177. How do you manage global state with the Context API?
178. What are some common patterns for state management in React?
179. How can you implement optimistic UI updates in React?
180. What are the trade-offs between Redux and Context API?

Error Handling

181. How do you implement error boundaries in React?
182. What is the purpose of error boundaries?
183. How do you log errors in a React application?
184. What are some strategies for error recovery in React?
185. How do you handle form validation errors in React?

Miscellaneous Concepts

186. What are the differences between React 16 and React 17?
187. How do you implement hot reloading in a React application?
188. What is the purpose of StrictMode in React?
189. How do you use the useLayoutEffect hook?
190. What is the difference between useLayoutEffect and useEffect?

Miscellaneous Continued

191. How can you debounce input in React?
192. What is the purpose of the useImperativeHandle hook?
193. How do you handle dynamic imports in React?
194. How can you create a loading component in React?
195. What are the benefits of using CSS transitions in React?

Interview Questions

React Ecosystem

196. What is the purpose of Next.js?
197. How do you implement routing in Next.js?
198. What are the benefits of using Gatsby with React?
199. What is React Native, and how does it differ from React?
200. What are some popular backends for React applications?

Advanced Patterns

201. How do you implement the Compound Component pattern in React?
202. What is the Provider pattern in React?
203. How can you create a dynamic form with React?
204. What are controlled vs. uncontrolled components?
205. How do you implement a Modal component in React?

Miscellaneous Techniques

206. How can you use an infinite scroll in React?
207. What is a carousel, and how can you implement it in React?
208. How do you handle file uploads in React?
209. What is the purpose of React Query?
210. How do you manage real-time data in React?

Responsive Design

211. How do you implement responsive layouts in React?
212. What are media queries, and how do you use them in React?
213. How do you implement mobile-first design in React?
214. What is the significance of viewport units in CSS?
215. How can you manage breakpoints in a React application?

Testing Deep Dive

216. How do you test asynchronous code in React?
217. What is mock testing, and how is it used in React?
218. How do you test Redux-connected components?
219. How do you handle snapshot testing for dynamic content?
220. What are the benefits of testing libraries in React?

Internationalization

221. How do you implement internationalization in React?
222. What is the purpose of libraries like react-intl?

223. How do you handle pluralization in translations?
224. What are some best practices for managing translations?
225. How do you switch languages in a React application?

Working with TypeScript

226. How do you set up TypeScript in a React project?
227. What are the benefits of using TypeScript with React?
228. How do you define prop types with TypeScript?
229. What is the purpose of interfaces in TypeScript?
230. How do you manage state with TypeScript in React?

Version Control

231. How do you use Git with a React project?
232. What is the purpose of .gitignore?
233. How can you manage branches in a Git repository?
234. What are some best practices for commit messages?
235. How do you handle merge conflicts in Git?

DevOps

236. What are some common deployment methods for React applications?
237. How do you use Docker with React?
238. What is continuous integration, and how can it be implemented with React?
239. How do you monitor a React application in production?
240. What is the purpose of a CDN in serving React apps?

Security

241. How do you handle XSS attacks in React applications?
242. What is CSRF, and how can you prevent it?
243. How do you secure sensitive information in a React app?
244. What are some common security best practices for React?
245. How do you handle user authentication securely?

Real-World Applications

246. How do you implement a search functionality in React?
247. What are some patterns for handling user preferences?
248. How do you implement a rating component in React?
249. What are some considerations for building a chat application?
250. How do you implement a notifications system in React?

Interview Questions

Community and Resources

251. What are some popular React blogs or websites?
252. How do you participate in the React community?
253. What are some React conferences or meetups you know of?
254. How do you use Stack Overflow for React-related queries?
255. What are some GitHub repositories you follow for React?

Final Thoughts

256. What are your favorite React tools or libraries?
257. How do you keep learning about React and its ecosystem?
258. What challenges have you faced while working with React?
259. What is your favorite project you've built with React?
260. How do you approach debugging in React applications?

Expanding Knowledge

261. How do you handle custom fonts in React?
262. What are the differences between React 17 and 18?
263. How do you implement a breadcrumb navigation in React?
264. What are some considerations for building a single-page application?
265. How do you use environment variables in a React project?

DevTools and Environment

266. What is React DevTools?
267. How do you use Chrome DevTools to debug React applications?
268. What is the purpose of source maps in React?
269. How can you set up a testing environment for React?
270. What are some common build tools used with React?

Mobile Development

271. How does React Native differ from React?
272. What are some challenges in mobile development with React?
273. How do you implement native modules in React Native?
274. What are the benefits of using React Native for mobile apps?
275. How do you manage device-specific features in React Native?

Final Set of Questions

276. What are some strategies for working with legacy React code?
277. How do you implement lazy loading for images in React?

278. What is the purpose of React.forwardRef?
279. How do you implement authentication in a React application?
280. How do you optimize images in a React application?

Wrapping Up

281. What are some common misconceptions about React?
282. How do you manage time zones in a React application?
283. What is the significance of accessibility testing in React?
284. How do you create a custom theme provider in React?
285. What are some tools for performance profiling in React?

Advanced Concepts

286. How do you handle nested contexts in React?
287. What is the purpose of useDebugValue?
288. How do you integrate third-party libraries into a React app?
289. How can you use service workers in a React application?
290. What are some patterns for managing complex component trees?

Miscellaneous and Real-World Scenarios

291. How do you handle long lists of data in React?
292. What is the difference between client-side and server-side rendering?
293. How do you create a sticky header in React?
294. What are some ways to enhance performance for slow connections?
295. How do you implement a dashboard in React?

Industry Insights

296. What are the current trends in React development?
297. How do you evaluate the quality of a React library?
298. What are some key metrics for measuring React application performance?
299. How do you choose the right tools for a React project?
300. What are the future developments you anticipate in the React ecosystem?

This extensive list of questions covers a wide range of topics related to React.js, allowing you to assess a candidate's foundational knowledge, practical skills, and understanding of best practices in React development.

Interview Questions

Here are some coding interview questions related to React.js that test a candidate's practical skills, understanding of concepts, and problem-solving abilities. Each question includes a brief description of what the interviewer might expect or be looking for in the answer.

React Coding Questions

1. **Build a Simple Counter Component**
    - Create a counter component that increments and decrements a number when the respective buttons are clicked.
2. **Todo List Application**
    - Build a simple todo list app that allows users to add, remove, and mark items as complete.
3. **Conditional Rendering**
    - Create a component that displays a message based on the user's logged-in status (logged in or logged out).
4. **Fetching Data from an API**
    - Implement a component that fetches data from a public API (like JSONPlaceholder) and displays the results.
5. **Form Handling**
    - Build a form component that captures user input (name, email) and displays it below the form when submitted.
6. **Dynamic List Rendering**
    - Create a component that accepts an array of objects as props and renders a list of items based on that array.
7. **Implement a Toggle Component**
    - Create a toggle switch component that switches between "On" and "Off" states when clicked.
8. **Search Filter**
    - Build a search component that filters a list of items based on user input.
9. **Custom Hook**
    - Create a custom hook that tracks the window width and logs it to the console when it changes.
10. **Context API Example**
    - Build a simple theme toggler using React Context API that switches between light and dark modes.
11. **Error Boundary**
    - Implement an error boundary component that catches JavaScript errors in its child components and displays a fallback UI.
12. **Debounce Input Field**
    - Create an input field that only updates its value after the user has stopped typing for a specific duration.
13. **Infinite Scroll**

- Build a component that implements infinite scrolling by fetching new data when the user scrolls to the bottom of the page.

14. **Modal Component**
    - Create a reusable modal component that can display different content and can be opened and closed via props.
15. **Drag and Drop List**
    - Implement a drag-and-drop list where items can be rearranged using the HTML5 drag-and-drop API.
16. **Controlled vs. Uncontrolled Components**
    - Write a controlled component for a text input and compare it with an uncontrolled component.
17. **React Router Implementation**
    - Create a simple app with at least two pages using React Router and implement navigation between them.
18. **Performance Optimization**
    - Write a component that uses React.memo to prevent unnecessary re-renders when props don't change.
19. **Using Refs**
    - Implement a component that focuses an input field when a button is clicked using the useRef hook.
20. **Implement a Rating Component**
    - Create a star rating component where users can select a rating from 1 to 5 stars.

Advanced Coding Questions

21. **State Management with Redux**
    - Build a small application using Redux to manage the state of a counter with increment and decrement actions.
22. **React Hook Form**
    - Create a form using React Hook Form that validates input fields and displays error messages.
23. **Memoization with useMemo**
    - Write a component that calculates and memoizes a value based on props to prevent recalculation on re-renders.
24. **Build a Calendar Component**
    - Create a simple calendar component that displays the current month and allows navigation between months.
25. **Build a Carousel Component**
    - Implement a simple image carousel with next and previous buttons to cycle through images.

Tips for Candidates

Interview Questions

- **Understand the Requirements**: Make sure to clarify the requirements and constraints of each question before diving into coding.
- **Think Aloud**: Communicate your thought process clearly while coding, as it helps interviewers understand your approach.
- **Write Clean Code**: Focus on writing clean, readable, and maintainable code. Consider using functional components and hooks.
- **Test Your Code**: If time allows, test your components to ensure they behave as expected.

These coding questions should help gauge a candidate's practical knowledge of React and their ability to apply concepts in real-world scenarios.