Here is a list of 50 rarely asked React.js interview questions that go beyond the usual interview queries. These questions cover advanced concepts and deeper understanding of React.js:

1. How do you optimize performance in a large-scale React application?
2. Explain the concept of **Reconciliation** in React.
3. What are **Controlled** vs **Uncontrolled** components in React? Can you give an example of when you'd use one over the other?
4. What is **React Fiber** and how does it improve React's rendering performance?
5. How can you create a custom hook for form validation in React?
6. Explain the **virtual DOM** and its benefits in React.
7. What are the differences between React.memo() and PureComponent?
8. How would you handle **error boundaries** in a large React app?
9. What is **React Suspense** and how can it be used to manage code splitting?
10. How would you implement **infinite scrolling** in a React component?
11. How can you prevent unnecessary re-renders in React functional components?
12. Can you explain the **React Context API** in depth? How does it compare to Redux?
13. What is the difference between useEffect with an empty dependency array and useLayoutEffect?
14. How does React’s **useReducer** hook work, and how is it different from useState?
15. What are **HOC** (Higher Order Components) and how are they used in React?
16. How would you implement **code splitting** in a React app?
17. What are **render props** and when should they be used in React?
18. Explain the concept of **JSX** and how it gets transformed into HTML.
19. What is the purpose of React.StrictMode and how does it help during development?
20. How do you manage **side effects** in React when using hooks?
21. Can you explain the concept of **Lazy Loading** in React and how it can be used to improve app performance?
22. What is the role of **keys** in lists in React and why are they important?
23. Explain the lifecycle of a **class-based component** in React, from mount to unmount.
24. What is **React Portals** and when would you use it?
25. How would you handle **state synchronization** between React components and external systems?
26. How can you use **React refs** for accessing DOM elements in functional components?
27. What is **React.memo** used for, and how does it optimize functional components?
28. How can you handle **asynchronous operations** in React (e.g., fetching data)?
29. Can you explain the **Component Composition** pattern and how it differs from inheritance?
30. What are the **pros and cons** of using Redux in a React application?
31. How does React’s **batching** mechanism work in terms of updating the state?
32. Can you explain how **React.lazy()** works with dynamic imports for code splitting?
33. How would you handle **local storage** and **session storage** in a React application?
34. What are the potential risks of overusing **useEffect**?
35. How would you implement a **debounced input field** using React hooks?
36. What are **Compound Components** and when would you use them?
37. How can you handle **dependency management** in a complex React app?
38. What is the difference between **shallow equality** and **deep equality** in React?
39. How can you handle **drag-and-drop** functionality in React?
40. What is **React’s Context API** and how does it improve state management over traditional prop drilling?
41. How would you implement **server-side rendering** (SSR) with React?
42. Can you explain the difference between **React's setState()** method and **useState()** hook?
43. What is the **shouldComponentUpdate** method, and how do you use it in React class components?
44. How do you handle **modals** or **popups** in React to ensure accessibility and manage state?
45. How would you manage **complex forms** in React with multiple fields and validations?
46. Explain the concept of **memoization** in React and when it's useful.
47. How do you implement **client-side routing** using React Router?
48. What is **React’s Refs forwarding** and how is it different from regular refs?
49. How do you test **React components** with **React Testing Library** and why is it preferred over Enzyme?
50. How would you handle **authentication** and **authorization** in a React app?

These questions dive deeper into understanding React's core concepts and performance optimizations, as well as its advanced features and integration patterns.