For Freshers:

1. What is React?
React is a JavaScript library for building user interfaces, specifically for single-page applications. It allows developers to create reusable UI components.

2. What are components in React?
Components are the building blocks of a React application. They are reusable pieces of UI that can manage their own state.

3. What is JSX?
JSX stands for JavaScript XML. It is a syntax extension for JavaScript, which looks similar to HTML, used with React to describe what the UI should look like.

4. How do you create a React component?
A React component can be created using either a JavaScript function or a class that returns JSX.

5. What is the difference between state and props?
`state` is a local data storage that is specific to a component and can change over time, while `props` are inputs to a component that are passed down from a parent component and cannot be modified.

6. What are React Hooks?
Hooks are functions that allow you to use state and other React features in functional components.

7. What is the use of useState in React?
`useState` is a Hook that allows you to add state to functional components.

8. What is the Virtual DOM?
The Virtual DOM is an in-memory representation of the real DOM elements generated by React components before they are rendered on the page. It allows React to efficiently update and render components.

9. What is the purpose of useEffect in React?

`useEffect` is a Hook that allows you to perform side effects in your components, such as data fetching, setting up a subscription, and manually changing the DOM.

10. How do you pass data between React components?

Data is passed between components using `props`.

11. What is prop drilling and how can it be avoided?

Prop drilling is the process of passing data through multiple nested components. It can be avoided using Context API or state management libraries like Redux.

12. What is React Router?

React Router is a standard library for routing in React. It allows you to navigate between different components in a React application.

13. What are Higher-Order Components (HOC)?

HOCs are functions that take a component and return a new component with added functionality.

14. What is a key in React and why is it important?

A key is a unique identifier used by React to track changes in lists of elements. It helps React to optimize re-rendering.

15. How can you optimize performance in a React application?

Performance can be optimized using techniques like lazy loading, code splitting, memoization, and avoiding unnecessary re-renders with `React.memo` or `PureComponent`.

16. What is the purpose of useRef in React?

`useRef` is a Hook that allows you to persist values between renders without causing a re-render.

17. How do you handle forms in React?

Forms in React are handled using controlled components, where the form data is managed by React state.

18. What is the difference between controlled and uncontrolled components?
Controlled components have their form data managed by React state, while uncontrolled components manage their own state using the DOM.

19. What is context in React and when would you use it?
Context provides a way to pass data through the component tree without having to pass props down manually at every level.

20. What is React.StrictMode?
`React.StrictMode` is a tool used for highlighting potential problems in an application. It activates additional checks and warnings.

21. What is React.Fragment?
`React.Fragment` allows you to group a list of children elements without adding extra nodes to the DOM.

22. What is a PureComponent in React?
`PureComponent` is a base class that optimizes performance by preventing unnecessary re-renders if the props and state have not changed.

23. How does React handle events?
React events are handled using camelCase syntax, and they work similar to DOM events, but with some syntactical differences.

24. What are React portals?
React portals allow you to render children into a DOM node that exists outside the hierarchy of the parent component.

25. How does reconciliation work in React?
Reconciliation is the process by which React updates the DOM with the results of rendering a component. It compares the previous and next virtual DOM and makes the minimal number of changes to the real DOM.

For 5+ Years Experienced:

26. What is code splitting and why is it important in React?
Code splitting is a technique used to split your code into smaller bundles, which can be loaded on demand, improving the performance of your React app.

27. How do you manage state in large-scale React applications?
State in large-scale applications can be managed using state management libraries like Redux, MobX, or by using React's Context API for smaller applications.

28. What are some common performance pitfalls in React, and how can they be avoided?
Common pitfalls include unnecessary re-renders, large bundle sizes, and inefficient rendering. These can be avoided by using memoization, code splitting, lazy loading, and optimizing component updates.

29. How do you handle asynchronous operations in React?
Asynchronous operations in React are handled using Promises, `async/await`, or external libraries like Axios for data fetching. React Query or SWR can also be used for data management.

30. What is server-side rendering (SSR) in React, and what are its benefits?
SSR is a technique where HTML is rendered on the server before being sent to the client, improving performance and SEO. Next.js is commonly used for SSR in React.

31. Explain the concept of React Fiber.
React Fiber is the new reconciliation algorithm in React 16 that enables the virtual DOM to be updated asynchronously. It improves the rendering process, making React more responsive to user interactions.

32. How do you implement authentication in a React application?
Authentication can be implemented using JWT (JSON Web Tokens), OAuth, or third-party services like Firebase. The user's authentication state can be managed using React context or state management libraries.

33. What are the limitations of React, and how do you work around them?

Limitations of React include the steep learning curve for beginners, the fast pace of changes in the ecosystem, and challenges with SEO (without SSR). Workarounds include using frameworks like Next.js, proper documentation, and staying updated with the React community.

34. How would you handle error boundaries in React?
Error boundaries are React components that catch JavaScript errors anywhere in their child component tree, log those errors, and display a fallback UI instead of crashing the whole component tree.

35. What is the role of keys in lists, and what happens if you don't use them?
Keys help React identify which items have changed, are added, or are removed. If you don't use keys, React will re-render the entire list instead of only the items that have changed, leading to performance issues.

36. How do you optimize the performance of a React application?
Optimization techniques include using `React.memo`, `useMemo`, `useCallback`, avoiding unnecessary re-renders, code splitting, lazy loading, and implementing proper state management.

37. What is the purpose of the useReducer Hook?
`useReducer` is a Hook that is used for state management in React. It is an alternative to `useState` and is useful for managing complex state logic or when the state depends on previous values.

38. How do you handle side effects in React applications?
Side effects are handled using the `useEffect` Hook, or by using middleware like Redux Thunk or Redux Saga for more complex side effects.

39. What are some strategies for testing React components?
Testing strategies include using tools like Jest and React Testing Library for unit and integration testing, and Enzyme for shallow rendering and testing component output.

40. What is React.lazy and how does it work?

`React.lazy` is a function that allows you to load components lazily through code splitting. It helps in reducing the initial load time by splitting the bundle into smaller chunks.

41. How would you implement routing in a React application?
Routing is implemented using React Router, which provides components like `BrowserRouter`, `Route`, and `Link` to manage navigation and routing within a React application.

42. What is the significance of defaultProps in React?
`defaultProps` is used to define default values for props in a component. It helps in ensuring that a component renders with the necessary props even if some are not passed.

43. Explain the concept of lifting state up in React.
Lifting state up involves moving the state to the closest common ancestor of components that need it. This is done to manage shared state between multiple components effectively.

44. How do you handle immutability in React state management?
Immutability is handled by avoiding direct mutations of state. Instead, new objects or arrays are created using methods like `Object.assign`, the spread operator, or immutable libraries.

45. What are some common security issues in React, and how can they be mitigated?
Common security issues include XSS (Cross-Site Scripting) and vulnerabilities due to improper handling of user inputs

. These can be mitigated by sanitizing inputs, using HTTPS, and avoiding the use of `dangerouslySetInnerHTML`.

46. What is the difference between useMemo and useCallback?
`useMemo` memoizes the result of a function, while `useCallback` memoizes the function itself. Both are used to optimize performance by preventing unnecessary recalculations or re-renders.

47. How do you implement Redux in a React application?

Redux is implemented by creating a store, defining reducers, and connecting components to the store using `connect` or `useSelector` and `useDispatch` Hooks.

48. What is the difference between class components and functional components?

Class components use `this` and lifecycle methods, while functional components use Hooks for state and side effects. Functional components are simpler and more concise.

49. What are some advantages of using TypeScript with React?

TypeScript provides type safety, better tooling, and improved developer experience. It helps in catching errors early and improves the maintainability of React applications.

50. How do you manage side effects and asynchronous operations in Redux?

Side effects and asynchronous operations in Redux are managed using middleware like Redux Thunk or Redux Saga, which handle async logic and side effects outside of the reducers.