REACT INTERVIEW QUESTIONS WITH ANSWERS

BASIC :

1. **What is React?**
React is a JavaScript library for building user interfaces, primarily used for single-page applications. It allows developers to create large web applications that can update and render efficiently in response to data changes.

2. **What are components in React?**
Components are the building blocks of a React application. They are reusable, independent pieces of code that represent parts of the user interface. Components can be either class-based or functional.

3. **What is JSX?**
JSX stands for JavaScript XML. It is a syntax extension for JavaScript that allows you to write HTML directly within JavaScript. React uses JSX to describe what the UI should look like.

4. **What is the virtual DOM?**
The virtual DOM is an in-memory representation of the real DOM. React uses it to efficiently update the DOM by comparing the virtual DOM with the previous version and only applying the necessary changes to the real DOM.

5. **Explain the difference between a class component and a functional component.**
Class components are ES6 classes that extend React.Component and have access to lifecycle methods and state. Functional components are simpler, defined as functions, and use hooks like useState and useEffect to manage state and side effects.

6. **What is the useState hook?**
The useState hook is a function that lets you add state to functional components. It returns an array with the current state value and a function to update it.

7. **What are props in React?**
Props (short for properties) are inputs to components. They are passed from parent components to child components and are used to pass data and event handlers.

8. **How do you pass data from parent to child components?**
Data is passed from parent to child components through props. The parent component defines an attribute on the child component and assigns it a value, which the child component can then access via this.props or the props parameter in a functional component.

9. **What is a state in React?**
State is an object managed within a component that holds data that can change over time. Unlike props, state is private and fully controlled by the component.

10. **How do you handle events in React?**
Event handling in React is similar to handling events in DOM elements, but with some syntactical differences. For example, React events use camelCase syntax, and you pass the event handler function without parentheses.

11. **What are React Fragments?**
React Fragments let you group multiple elements without adding an extra node to the DOM. You can use the <React.Fragment> element or the shorthand syntax <>...</>.

12. **What is the purpose of the key prop in lists?**
The key prop is used to identify which items in a list have changed, been added, or removed. Keys should be unique and stable to help React optimize rendering.

13. **What is prop drilling, and how can it be avoided?**
Prop drilling refers to the process of passing data from a parent component to deeply nested child components via props. It can be avoided using the Context API, which allows you to pass data without passing props through every level of the component tree.

14. **How do you conditionally render components in React?**
You can conditionally render components using JavaScript logical operators like CC, ternary operators, or if statements inside the component's render method.

INTERMEDIATE :

15. **What is the useEffect hook?**
The useEffect hook is used to perform side effects in functional components, such as data fetching, subscriptions, or manually updating the DOM. It runs after every render by default but can be controlled using dependencies.

16. **How do you fetch data from an API in React?**
You can fetch data from an API using the fetch function or a library like Axios within the useEffect hook to ensure the data is fetched after the component mounts.

17. **Explain the context API and its use cases.**
The Context API is used to manage global state that needs to be accessible by multiple components without prop drilling. It is often used for theme management, user authentication, and language settings.

18. **What is the useContext hook?**
The useContext hook allows you to access the value of a context directly in a functional component, making it easier to consume context values without needing a wrapper component.

19. **What are higher-order components (HOCs)?**
HOCs are functions that take a component and return a new

component with added functionality. They are used for code reuse, logic, and behaviors in React applications.

20. **What is the useRef hook, and how is it used?**
The useRef hook creates a mutable object that persists across renders. It is often used to access or manipulate DOM elements directly and to store values that don't trigger re-renders when changed.

21. **How does React handle forms?**
React handles forms using controlled components, where form elements like <input> are controlled by React state, making it easier to manage user input and form submission.

22. **Explain the concept of controlled and uncontrolled components.**
Controlled components are form elements that are controlled by React state. Uncontrolled components manage their own state internally and are accessed using refs.

23. **What is lazy loading, and how do you implement it in React?**
Lazy loading is a technique to defer the loading of non-essential resources until they are needed. In React, you can implement lazy loading using React.lazy() and Suspense to load components only when they are needed.

24. **What is the difference between useMemo and useCallback hooks?**
useMemo is used to memoize the result of a computation, avoiding recalculations unless dependencies change. useCallback is used to memoize a function, preventing it from being recreated on every render unless dependencies change.

25. **How do you optimize the performance of a React application?**
Performance can be optimized using techniques like code splitting, memoization (React.memo, useMemo, useCallback), avoiding unnecessary re-renders, using useEffect with dependencies, and using a proper state management strategy.

26. **What is server-side rendering (SSR) in React?**
SSR is a technique where the HTML content is generated on the server for each request, improving performance and SEO. Next.js is a popular framework that enables SSR with React.

27. **Explain React Router and its use cases.**
React Router is a library for routing in React applications. It allows you to create dynamic routes and manage navigation within single-page applications.

28. **What is the useReducer hook?**
The useReducer hook is an alternative to useState for managing

complex state logic. It accepts a reducer function and an initial state, returning the current state and a dispatch function.

29. **What are custom hooks in React?**
Custom hooks are user-defined functions that allow you to reuse logic across multiple components. They follow the same rules as React hooks and can encapsulate stateful logic.

30. **How do you manage global state in a React application?**
Global state can be managed using the Context API, Redux, or other state management libraries like MobX or Zustand, depending on the complexity and needs of the application.

ADVANCED :

31. **What is React Fiber?**
React Fiber is the new reconciliation engine in React 16 and above. It allows React to split rendering work into chunks, improving the responsiveness of applications by pausing and resuming rendering tasks.

32. **Explain the concept of reconciliation in React.**
Reconciliation is the process by which React updates the DOM by comparing the virtual DOM with the previous version and only applying the necessary changes. This helps improve performance by minimizing direct DOM manipulations.

33. **How does React handle error boundaries?**
Error boundaries are React components that catch JavaScript errors in their child component tree, log those errors, and display a fallback UI instead of crashing the entire application.

34. **What are React portals, and how do you use them?**
React portals provide a way to render children into a DOM node that exists outside the hierarchy of the parent component. They are useful for modals, tooltips, and other UI elements that need to be visually separate from the rest of the application.

35. **How does concurrent rendering work in React?**
Concurrent rendering is an experimental feature in React that allows React to interrupt rendering work to handle high-priority updates, making applications more responsive by allowing the UI to remain interactive while heavy computations are performed in the background.

36. **What is React Suspense, and how do you use it?**
React Suspense is a feature that lets you wait for some code to load and declaratively specify a loading state while waiting. It's often used with React.lazy to handle lazy-loaded components.

37. **Explain the difference between static and dynamic routing in React.**
Static routing defines routes at build time and does not change,

whereas dynamic routing allows routes to be generated dynamically based on application state or user input.

38. **What is the significance of the StrictMode in React?**
StrictMode is a tool for highlighting potential problems in an application. It activates additional checks and warnings for its descendants, helping to identify unsafe lifecycle methods, legacy API usage, and other issues.

39. **How do you implement authentication in a React application?**
Authentication can be implemented using strategies like JWT (JSON Web Tokens), OAuth, or integrating with third-party services like Firebase or Auth0. This involves managing user sessions, storing tokens, and protecting routes with higher-order components or hooks.

40. **Explain the concept of code splitting in React.**
Code splitting is a technique to split the code into smaller chunks, which are loaded on demand rather than all at once. It improves the load time of the application. React supports code splitting via dynamic import() and tools like Webpack.

41. **What are render props, and how do they differ from higher-order components?**
Render props are a pattern where a component's prop is a function that returns a React element. Unlike higher-order components

(HOCs), which wrap a component, render props allow you to reuse component logic by passing a function to the child component.

42. **How do you handle side effects in React with Redux?**
Side effects in Redux are handled using middleware like Redux Thunk or Redux Saga. Thunk allows you to write action creators that return a function instead of an action, enabling asynchronous operations. Saga uses generator functions to handle side effects in a more complex and declarative manner.

43. **Explain how React's context API differs from Redux.**
The Context API is simpler and is suitable for passing data through the component tree without prop drilling, but it is less powerful than Redux. Redux is a state management library that provides a more structured and scalable solution for managing global state, especially in large applications.

44. **What is the significance of React's Strict Mode, and how do you use it?**
StrictMode is a wrapper component that helps you identify potential problems in your application, such as unsafe lifecycle methods, deprecated APIs, and side effects that may not behave as expected. It doesn't render any UI but performs checks on the child components within it.

45. **How do you implement real-time features like WebSockets in React?**

Real-time features can be implemented using WebSockets by connecting to a WebSocket server in a React component (typically in useEffect for functional components). You listen for messages from the server and update the component state accordingly.

46. **What is the difference between shallow rendering and deep rendering in React testing?**
Shallow rendering tests a component by rendering only the component itself and not its children, making it faster and more focused on unit testing. Deep rendering, on the other hand, renders the component along with all of its child components, providing a more integrated test but at the cost of speed and complexity.

47. **Explain the concept of React reconciliation.**
Reconciliation is the process React uses to update the DOM when the state of a component changes. React compares the new virtual DOM with the old one and determines the minimal set of changes required to update the real DOM, which improves efficiency.

48. **How do you handle performance bottlenecks in large React applications?**
Performance bottlenecks can be handled by using techniques such as memoization (React.memo, useMemo, useCallback), optimizing the rendering of large lists (React.Virtualized), lazy loading components, reducing the number of re-renders, and profiling components to identify slow parts of the application.

49. **What are the potential downsides of using React's context API extensively?**
Using the Context API extensively can lead to performance issues due to re-renders triggered by context updates. It can also make the codebase harder to maintain and debug if not used carefully, especially in large applications.

50. **How do you implement a theme switcher in a React application?**
A theme switcher can be implemented using the Context API to store the current theme, and useState to toggle between themes. The context value is provided to the entire application, and components consume it to apply the corresponding theme styles.