# REACT INTERVIEW QUESTION BY LEARN CODING

**Basic Questions**

1. **What is React?**
    - React is a JavaScript library developed by Facebook for building user interfaces. It uses a component-based architecture and a virtual DOM for efficient updates.
2. **What are components in React?**
    - Components are reusable, independent pieces of code in React that define how a UI looks and behaves. They can be class-based or functional.
3. **What is JSX?**
    - JSX stands for JavaScript XML. It's a syntax extension that allows writing HTML within JavaScript code, making React components more readable.
4. **What is the difference between state and props?**
    - **State:** Internal to a component, used to store data that changes over time.
    - **Props:** External to a component, passed by a parent component to make it dynamic.

5. **What is the Virtual DOM?**
    - The Virtual DOM is a lightweight copy of the real DOM. React uses it to batch updates and apply changes efficiently, improving performance.

**Intermediate Questions**

6. **What are React hooks?**
    - Hooks are functions introduced in React 16.8 that allow using state and other React features in functional components. Examples include useState, useEffect, and useContext.
7. **Explain the use of useEffect.**
    - useEffect is used to perform side effects in functional components, such as fetching data, subscribing to events, or manually changing the DOM. It runs after the render cycle.
8. **What is the difference between controlled and uncontrolled components?**
    - **Controlled Components:** Managed by React state (e.g., <input value={stateValue} onChange={handleChange} />).
    - **Uncontrolled Components:** Managed by the DOM (e.g., <input ref={inputRef} />).
9. **How does React handle forms?**

- Forms in React are controlled using state or uncontrolled with refs. React provides controlled form inputs for a better way to track and validate user input.

10. **What is the Context API?**
    - The Context API allows sharing state globally across components without passing props down manually at every level.

**Advanced Questions**

11. **What is code splitting in React?**
    - Code splitting is a technique to break a large application into smaller chunks to improve performance. React supports it through dynamic imports (React.lazy and Suspense).
12. **What is the purpose of keys in React?**
    - Keys help React identify and manage changes to items in a list efficiently. They should be unique and stable for each item.
13. **How does React's reconciliation process work?**
    - React compares the Virtual DOM to the real DOM and calculates the minimal set of changes required to update the UI efficiently.
14. **What is Prop Drilling, and how can it be avoided?**
    - **Prop Drilling:** Passing props through multiple components unnecessarily.
    - **Solution:** Use the Context API or state management libraries like Redux.
15. **Explain the difference between useMemo and useCallback.**
    - **useMemo:** Memoizes the return value of a function to avoid recalculations.
    - **useCallback:** Memoizes a function reference to avoid re-creating it.

**Scenario-Based Questions**

16. **How would you optimize a slow React application?**
    - Use React.memo for functional components.
    - Avoid unnecessary re-renders with shouldComponentUpdate or React.PureComponent.
    - Use code splitting with React.lazy and Suspense.
    - Optimize the state by lifting it to the minimal required level.
17. **What happens when you call setState in React?**
    - React schedules a re-render of the component and updates the Virtual DOM. It batches multiple setState calls for better performance.
18. **How would you handle errors in React?**

- Use Error Boundaries (componentDidCatch and getDerivedStateFromError) to catch and display errors without crashing the app.

19. **What are Higher-Order Components (HOCs)?**
    - HOCs are functions that take a component and return a new component with additional features. They are used for reusing logic.
20. **How do you handle asynchronous operations in React?**
    - Use useEffect for side effects and asynchronous operations. For complex state updates, libraries like Redux Thunk or Redux Saga can be used.

**Questions About Tools and Ecosystem**

21. **What is Redux, and how does it work with React?**
    - Redux is a state management library. It centralizes the application state in a single store and uses actions and reducers to manage updates.
22. **What is the difference between Redux and Context API?**
    - Context API is built into React and simpler for smaller apps. Redux is more powerful, providing middleware, dev tools, and better scalability for large applications.
23. **What is React Router?**
    - React Router is a library for routing in React apps. It enables navigation between different views or components.
24. **How do you test a React application?**
    - Use tools like Jest and React Testing Library for unit and integration tests. Snapshot testing is common for components.
25. **What is the significance of Webpack in React?**
    - Webpack is a module bundler used to bundle assets like JavaScript, CSS, and images. It optimizes the app for production.

**Advanced and Unique React Interview Questions**

26. **What is React Fiber, and how is it different from the older reconciliation algorithm?**
    - React Fiber is the new reconciliation algorithm introduced in React 16. It breaks rendering work into units and spreads it across multiple frames, enabling React to pause and resume work, making it faster and more efficient.
27. **What are render props in React?**
    - Render props is a pattern where a function is passed as a prop to a component to determine its output. It allows for sharing logic between components.
28. **Explain the significance of React.StrictMode.**

- React.StrictMode is a development-only tool that highlights potential issues like deprecated methods or side effects in your React application.

29. **What is shallow rendering in React testing?**
    - Shallow rendering renders only the top-level component without its child components. This is useful for unit testing a single component in isolation.

30. **What is the difference between useEffect and useLayoutEffect?**
    - useEffect runs after the DOM has been painted, while useLayoutEffect runs synchronously after the DOM mutations but before the browser paints. The latter is used when DOM measurements or mutations are necessary.

31. **How does React handle updates in lists with keys?**
    - React uses keys to identify which list items have changed, been added, or removed. This helps optimize re-rendering by only updating the necessary parts of the DOM.

32. **What are synthetic events in React?**
    - Synthetic events are React's cross-browser wrapper around the browser's native events. They ensure consistent behavior across different browsers.

33. **What is React Portal, and why is it used?**
    - React Portals allow rendering children into a DOM node outside the parent hierarchy. It’s commonly used for modals, tooltips, and overlays.

34. **What is the difference between React.PureComponent and React.Component?**
    - React.PureComponent performs a shallow comparison of props and state to prevent unnecessary re-renders, whereas React.Component does not.

35. **How do you implement lazy loading in React?**
    - Use React.lazy and Suspense for dynamic imports. Example:
    - const LazyComponent = React.lazy(() => import('./LazyComponent'));
    - <Suspense fallback={<div>Loading...</div>}>
    -   <LazyComponent />
    - </Suspense>

36. **What is the difference between context and Redux in state management?**
    - Context API is simple and built into React, suitable for small-scale state sharing. Redux is more structured and suited for complex applications requiring middleware, debugging tools, and scalability.

37. **What are some common performance bottlenecks in React, and how do you fix them?**

- Bottlenecks: Unnecessary re-renders, large bundles, and excessive API calls. Fixes: Use memoization (React.memo, useMemo), lazy loading, and debouncing/throttling API calls.

38. **What is the difference between useReducer and useState?**
    - useState is simpler for local state management, while useReducer is better for complex state logic or when state depends on previous states.
39. **What are error boundaries, and how do you use them?**
    - Error boundaries are React components that catch JavaScript errors in their child component tree and render a fallback UI instead of crashing the app. Implemented using componentDidCatch and getDerivedStateFromError.
40. **What is hydration in React?**
    - Hydration is the process of attaching React event listeners to the existing HTML markup on the server during server-side rendering (SSR).
41. **How would you manage side effects in React?**
    - Side effects are managed using useEffect for functional components. For advanced use cases, libraries like Redux Thunk, Redux Saga, or React Query can handle side effects more robustly.
42. **What are concurrent features in React?**
    - React's concurrent features (introduced in React 18) allow rendering to be interrupted and resumed, improving user experience during heavy computations or rendering.
43. **What is the difference between React.createRef and useRef?**
    - React.createRef is used in class components, while useRef is for functional components. Both provide a way to access and interact with DOM elements or store mutable values.
44. **What are controlled fragments in React?**
    - Fragments (<React.Fragment> or <>) allow grouping multiple elements without adding extra nodes to the DOM.
45. **Explain the role of React.forwardRef.**
    - React.forwardRef allows refs to be passed to child components. It's useful for accessing DOM nodes or child component instances.
46. **What is server-side rendering (SSR) in React?**
    - SSR is the process of rendering React components on the server and sending fully rendered HTML to the client. Libraries like Next.js make SSR implementation easier.
47. **What are custom hooks in React?**
    - Custom hooks are user-defined functions that allow reusing stateful logic across components. They follow the naming convention useSomething.

48. **What is reconciliation, and how does React perform it?**
   - Reconciliation is React's process of updating the DOM by comparing the Virtual DOM with the previous Virtual DOM and applying minimal updates.

49. **How do you handle memory leaks in React applications?**
   - Clean up subscriptions and event listeners in the useEffect cleanup function, avoid excessive DOM manipulation, and use tools like React Profiler to identify issues.

50. **What are React fibers, and how do they enable concurrent rendering?**
   - Fibers are units of work in React's rendering process. They enable breaking rendering work into chunks, allowing React to pause and resume rendering tasks.