**REACT IMPORTANT CONCEPTS**

1. What is React JS?  
React JS is a JavaScript library for building user interfaces, particularly for single-page applications. It allows developers to create reusable UI components and manage the state of an application efficiently.

2. What are the features of React?

- \*\*Component-Based Architecture\*\*: Encourages reusability.

- \*\*Virtual DOM\*\*: Efficiently updates and renders components.

- \*\*JSX\*\*: Allows HTML-like syntax in JavaScript.

- \*\*Unidirectional Data Flow\*\*: Makes it easier to understand data changes.

- \*\*Lifecycle Methods\*\*: Provides control over component lifecycle.

3. What is JSX?

JSX (JavaScript XML) is a syntax extension that allows mixing HTML with JavaScript. It makes it easier to create React elements and components. JSX needs to be transpiled to JavaScript.

4. What are components in React?

Components are reusable, self-contained pieces of UI that define how a certain part of the user interface should appear. They can be class components or functional components.

5. Difference between class and functional components?

- \*\*Class Components\*\*: Can hold state and lifecycle methods; defined using ES6 classes.

- \*\*Functional Components\*\*: Simpler, typically stateless; can use hooks for state and side effects.

6. What are props in React?

Props (short for properties) are read-only attributes passed to components to configure them. They allow data to be passed from parent to child components.

7. What is state in React?

State is a built-in object that allows components to manage their internal data. Unlike props, state can change over time and causes the component to re-render when it does.

8. How does state differ from props?

- \*\*Props\*\*: Immutable and passed from parent to child.

- \*\*State\*\*: Mutable and managed within the component.

9. What is the virtual DOM?

The virtual DOM is a lightweight representation of the actual DOM. React uses it to optimize updates and rendering, minimizing direct manipulation of the real DOM.

10. How does the virtual DOM work?

When a component's state or props change, React creates a new virtual DOM tree. It then compares this tree with the previous one (diffing) and updates only the parts of the real DOM that have changed.

11. What is a React Hook?

React Hooks are functions that let you use state and other React features in functional components. They allow you to manage state and side effects without using class components.

12. What are the commonly used hooks?

- `useState`

- `useEffect`

- `useContext`

- `useReducer`

- `useMemo`

- `useCallback`

13. Explain the useState hook.

`useState` is a hook that allows you to add state to functional components. It returns an array with the current state and a function to update it.

14. Explain the useEffect hook.

`useEffect` is a hook that allows you to perform side effects in functional components, such as data fetching or DOM manipulation. It runs after every render and can be configured to run conditionally.

15. What is the purpose of useReducer?

`useReducer` is a hook that manages complex state logic in functional components. It's similar to `useState` but is more suitable for managing state transitions and actions.

16. What are higher-order components (HOCs)?

HOCs are functions that take a component and return a new component, enhancing it with additional props or functionality. They are used for code reuse and cross-cutting concerns.

17. What is the purpose of useContext?

`useContext` allows you to access context directly in functional components, making it easier to share values across the component tree without passing props manually at every level.

18. How does the context API work?

The Context API provides a way to share values between components without passing props explicitly through the component tree. It consists of `React.createContext`, a Provider, and a Consumer.

19. Explain React lifecycle methods.

Lifecycle methods are special methods in class components that allow you to hook into different phases of a component's lifecycle, such as mounting, updating, and unmounting.

20. What is the purpose of componentDidMount?

`componentDidMount` is called after a component is mounted. It’s often used for initiating data fetching or setting up subscriptions.

21. What is the purpose of componentDidUpdate?

`componentDidUpdate` is called after updates to a component. It can be used to perform operations based on changes to props or state.

22. What is the purpose of componentWillUnmount?

`componentWillUnmount` is called just before a component is unmounted. It’s used for cleanup tasks like cancelling network requests or removing event listeners.

23. What is React Router?

React Router is a library that enables dynamic routing in React applications, allowing navigation between different components or pages.

24. How do you handle routing in React?

You can handle routing using `BrowserRouter` and `Route` components from React Router to define routes and link them to specific components.

25. What is lazy loading in React?

Lazy loading is a technique to load components only when they are needed, which improves the performance of applications by reducing initial load time.

26. What is code splitting?

Code splitting is the practice of splitting your code into smaller chunks that can be loaded on demand, optimizing load time and resource usage.

27. How do you optimize performance in React?

- Use `React.memo` to prevent unnecessary re-renders.

- Use `useMemo` and `useCallback` to memoize values and functions.

- Optimize component structure and avoid deep nesting.

28. What are React fragments?

React fragments allow you to group multiple elements without adding extra nodes to the DOM. They are useful for returning multiple children from a component.

29. How do you handle events in React?

You handle events in React using camelCase syntax for event names and passing functions as event handlers.

30. What is controlled vs uncontrolled components?

- \*\*Controlled Components\*\*: Form inputs where React controls the state.

- \*\*Uncontrolled Components\*\*: Form inputs that maintain their own state, typically accessed via refs.

31. How do you lift state up in React?

You lift state up by moving state and state management to the nearest common ancestor of the components that need to share the state.

32. What is the purpose of React keys?

Keys help React identify which items have changed, are added, or are removed in lists. They improve performance by optimizing the reconciliation process.

33. How do you handle forms in React?

Forms are handled using controlled components by managing their state with `useState` and updating the state on input changes.

34. What is Redux?

Redux is a state management library for JavaScript applications that helps manage application state in a predictable way.

35. How do you use Redux with React?

You use Redux with React by creating a store, defining reducers, and connecting components using the `connect` function or `useSelector` and `useDispatch` hooks.

36. Explain the Redux workflow.

The Redux workflow involves dispatching actions, which are processed by reducers that update the state in the store, and then the updated state is reflected in the UI.

37. What is the role of reducers in Redux?

Reducers are pure functions that take the current state and an action and return a new state. They determine how the application’s state changes in response to actions.

38. What are actions in Redux?

Actions are plain JavaScript objects that describe a change in the application. They have a `type` property and can carry additional data (payload).

39. What is the purpose of the Redux store?

The Redux store holds the entire state of the application, allowing components to access and update state via actions and reducers.

40. How do you connect React with Redux?

You connect React with Redux using the `Provider` component to pass the store down the component tree and using `connect` or hooks like `useSelector` and `useDispatch` to interact with the store.

41. What is middleware in Redux?

Middleware in Redux allows you to extend Redux's capabilities by intercepting actions dispatched to the store, enabling features like asynchronous actions or logging.

42. Explain the concept of immutability.

Immutability means that data cannot be changed once it is created. In Redux, maintaining immutability allows for easier tracking of state changes and optimizing rendering.

43. What is the difference between React and Angular?

React is a library focused on building UI components, while Angular is a full-fledged framework that provides a comprehensive solution for building applications, including routing, forms, and state management.

44. How do you manage side effects in React?

Side effects are managed using the `useEffect` hook in functional components, which can handle operations like data fetching or subscriptions.

45. What is the purpose of useMemo?

`useMemo` is a hook that memoizes the result of a computation, preventing expensive calculations on every render unless the dependencies change.

46. What is the purpose of useCallback?

`useCallback` is a hook that memoizes a function, returning the same function reference unless its dependencies change, optimizing performance by preventing unnecessary re-renders.

47. How do you test React components?

React components can be tested using libraries like Jest and React Testing Library, which provide tools for rendering components and simulating user interactions.

48. What is Jest?

Jest is a JavaScript testing framework designed for simplicity and support for testing React applications, offering features like snapshot testing and mocks.

49. What is Enzyme?

En

zyme is a testing utility for React that makes it easier to test components' output and behavior, allowing shallow rendering and full DOM rendering.

50. How do you handle error boundaries in React?

Error boundaries are components that catch JavaScript errors in their child component tree. You implement them by defining `componentDidCatch` and using the `static getDerivedStateFromError` method to display a fallback UI.

These answers provide a solid foundation for discussing React concepts in an interview setting. Good luck!