React Important Questions

1. What is React and How Does It Work?

React is a JavaScript library for building user interfaces using a component-based approach. It uses a Virtual DOM to update only the parts of the DOM that need changes, which makes it highly efficient.

2. What are Components in React?

Components are the building blocks of React applications. They represent parts of the UI, can be functional or class-based, and help in creating reusable, isolated code.

3. What is JSX?

JSX (JavaScript XML) is a syntax extension for JavaScript that looks like HTML. It makes it easier to write and visualize React elements and is transformed into JavaScript code during compilation.

4. What is the Virtual DOM?

The Virtual DOM is a lightweight copy of the actual DOM. React updates the Virtual DOM first and compares it with the real DOM to make only necessary updates, improving performance.

5. Explain Props and State.

- **Props** are read-only attributes passed from parent to child components, while **State** is a component's internal data storage that can change over time.

6. What is the Component Lifecycle?

The component lifecycle consists of phases like mounting, updating, and unmounting. Lifecycle methods allow developers to perform actions at specific points in a component's life.

7. What are the Common Lifecycle Methods?

- **componentDidMount**: Runs after a component mounts.
- **componentDidUpdate**: Runs after a component updates.
- **componentWillUnmount**: Runs before a component unmounts.

8. What is the useEffect Hook?

useEffect allows you to perform side effects (like data fetching) in functional components. It runs after the component renders and re-runs when dependencies change.

9. What is the useState Hook?

useState adds state to functional components. It returns an array with the current state value and a function to update that state.

10. What is Conditional Rendering?

Conditional rendering in React allows components to render differently based on certain conditions, using JavaScript's conditional statements like if, ternary operators, or &&.

11. What are Keys in React?

Keys are unique identifiers for elements in lists, helping React track changes and improve performance by reusing elements instead of re-rendering the entire list.

12. What is a Controlled Component?

A controlled component in React is one where form data is handled by the component's state, usually with value and onChange attributes.

13. What is an Uncontrolled Component?

An uncontrolled component allows the DOM to handle form data. Instead of using state, you use refs to access form data directly.

14. What is the Context API?

The Context API provides a way to pass data through the component tree without prop drilling, allowing components to access “global” data easily.

15. What is Prop Drilling?

Prop drilling is the process of passing props from a top-level component to deeply nested components, which can be inefficient and lead to cluttered code.

16. What is Fragment in React?

Fragments allow grouping of multiple elements without adding extra nodes to the DOM. You can use <React.Fragment> or simply <></>.

17. What is useRef?

useRef creates a reference to a DOM element or variable, which can be used for accessing or modifying elements directly without causing re-renders.

18. What is Memoization in React?

Memoization is an optimization technique that stores the result of a function call and returns the cached result when the same inputs occur again, improving performance.

19. What is React.memo?

React.memo is a higher-order component that prevents re-rendering of functional components if the props haven’t changed, enhancing performance.

20. What is the Purpose of useCallback?

useCallback is a hook that memoizes a function so it doesn't get recreated on each render unless its dependencies change. It optimizes performance when passing callbacks to child components.

21. What is useMemo?

useMemo memoizes the result of a function, recalculating it only when dependencies change. It helps optimize performance for expensive computations.

22. What is the Difference Between useCallback and useMemo?

- **useCallback** memoizes a function.
- **useMemo** memoizes the result of a function.

Both help prevent unnecessary re-renders, but they're used for different purposes.

23. Explain Error Boundaries.

Error boundaries are React components that catch JavaScript errors in their child component tree, log them, and display a fallback UI to prevent the app from crashing.

24. How do you Optimize a React Application?

- Use **React.memo** for component memoization.
- Use **lazy loading** and **code splitting** for large components.
- Implement **useCallback** and **useMemo** for expensive calculations.
- Avoid unnecessary re-renders by using **shouldComponentUpdate** or **React.PureComponent**.

25. What is Code Splitting?

Code splitting is a technique to load only the necessary parts of an application, often achieved with **React.lazy** and **Suspense**. It improves initial load times by splitting code into smaller chunks.

26. What is React.lazy and Suspense?

- **React.lazy** allows components to load lazily (only when needed).
- **Suspense** provides a fallback UI while a lazy-loaded component is being fetched.

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

An HOC is a function that takes a component and returns a new component, adding additional functionality. It's a way to reuse component logic across multiple components.

28. What are Portals in React?

Portals enable rendering of a component's children outside its parent DOM node, useful for creating modals and popups without affecting the parent structure.

29. What is the Strict Mode in React?

Strict Mode is a development tool that highlights potential problems in the application by performing additional checks and warnings for the children of <React.StrictMode>.

30. What is the Purpose of React's useReducer Hook?

useReducer is an alternative to useState, helpful for handling complex state logic in functional components. It takes a reducer function and an initial state, and returns the current state and a dispatch function for triggering updates.

31. What is Reconciliation in React?

Reconciliation is React's process of updating the DOM efficiently by comparing the new Virtual DOM with the previous one and applying only the necessary changes to the actual DOM. React uses an algorithm to detect changes, like additions, updates, or deletions, in components.

32. What is the Difference Between Functional and Class Components?

- **Class Components**: Require the class keyword and include lifecycle methods. State is managed with this.state, and functions need to be bound to the component.
- **Functional Components**: Simpler to write and rely on hooks (useState, useEffect) to manage state and lifecycle. They're preferred in modern React for their simplicity and performance.

33. What is Strict Mode in React?

Strict Mode is a wrapper component used to detect potential issues in the code. It helps in identifying unsafe lifecycles, deprecated APIs, and unexpected side effects. It doesn't render any visible UI, just performs checks in development mode.

34. How Does React Handle Events?

React uses a synthetic event system, which is a wrapper around native events, making them cross-browser compatible. Instead of directly attaching events to DOM elements, React attaches them to a single root element, improving performance.

35. What are Render Props?

Render props is a pattern where a component's children is a function that receives props and returns a React element. This technique is used for sharing logic between components by passing the function as a child. Example:

<DataProvider render={(data) => <DisplayData data={data} />} />

36. What is the Purpose of Forward Refs in React?

React.forwardRef allows a parent component to directly access the ref of a child component. This is useful for focusing on child inputs or accessing child elements that might otherwise be out of reach.

37. What is PropTypes and How is It Used?

PropTypes is a type-checking library in React used to enforce type and shape constraints on component props. It's especially useful for catching type-related bugs in development. Example:

```
MyComponent.propTypes = {
  name: PropTypes.string.isRequired,
  age: PropTypes.number,
};
```

38. How Does the Context API Compare to Redux?

Both provide ways to manage global state in an app:

- **Context API**: Lightweight and built into React, ideal for small to medium apps with simple state needs.
- **Redux**: A more powerful state management library that works with various middlewares, great for large apps with complex, interconnected states.

39. What is Shallow Rendering?

Shallow rendering is a testing method that renders a component without rendering its children. This allows developers to test component logic in isolation without depending on the behavior of child components.

40. What is PureComponent in React?

React.PureComponent is a class component that implements a shallow comparison for props and state to prevent unnecessary re-renders. It's useful for optimizing performance by avoiding re-renders when prop and state values haven't changed.

41. What are Custom Hooks?

Custom hooks are functions that allow you to reuse logic between components. They are created by combining built-in hooks (useState, useEffect, etc.) and custom logic. Custom hooks start with the prefix use (e.g., useFetch, useForm).

42. How Can You Avoid Prop Drilling?

To avoid prop drilling (passing props through many levels of components), you can:

- Use the **Context API** to manage global state.
- Use **Redux** for complex state management.
- Use **custom hooks** to encapsulate shared logic.

43. What is Lazy Loading in React?

Lazy loading delays loading of components until they're needed, improving the app's initial load time. React provides React.lazy and Suspense to handle lazy loading of components.

44. What is Suspense in React?

Suspense is a component for managing loading states while lazy-loading components or handling asynchronous tasks in concurrent mode. It shows a fallback UI until the data or component is ready.

45. How Do you Handle Forms in React?

Forms can be controlled or uncontrolled:

- **Controlled Forms**: Use state to manage form inputs.
- **Uncontrolled Forms**: Use refs to access input values directly from the DOM.

46. What is React's Error Boundary and How is It Used?

Error boundaries catch JavaScript errors in their child components and display a fallback UI instead of breaking the whole app. They're created by implementing componentDidCatch and getDerivedStateFromError methods in class components.

47. What is Hydration in React?

Hydration is the process of attaching event listeners to pre-rendered HTML in server-side rendering (SSR) apps. After the HTML is rendered on the server, React uses it as a starting point on the client to improve load speed.

48. What is Server-Side Rendering (SSR)?

SSR is a technique where the HTML of a page is generated on the server, making it ready for search engines and reducing load times. React can achieve SSR with frameworks like Next.js.

49. What is the Difference Between SSR and CSR (Client-Side Rendering)?

- **SSR**: Renders HTML on the server; suitable for SEO and quick initial loads.

- **CSR**: Renders HTML on the client; suitable for interactive, single-page applications.

50. Explain the Difference Between useEffect and componentDidMount?

- componentDidMount runs only once after the initial render in class components.
- useEffect can mimic componentDidMount by passing an empty dependency array [], but it's more flexible because it can run after any state or prop change.

51. How is Context Used with Hooks in Functional Components?

With useContext, functional components can access Context values directly, making it easy to consume global state without a class component.

const value = useContext(MyContext);

52. What is the Difference Between useLayoutEffect and useEffect?

- **useEffect**: Runs asynchronously after the render is committed to the screen.
- **useLayoutEffect**: Runs synchronously after layout but before the browser repaints, ideal for layout adjustments.

53. What is Code Splitting in React and Why is It Important?

Code splitting is a technique that breaks the app into smaller chunks, loading only what's needed. It improves performance, especially for large applications, by reducing the initial load size.

54. What is Context API's Provider and Consumer?

- **Provider**: Passes data to child components in the component tree.

- **Consumer**: Reads data from Context, allowing any child to access values set in the Provider.

55. What is useImperativeHandle?

useImperativeHandle customizes the instance value exposed to parent components using refs. It's typically used with forwardRef to control what gets accessed by a parent.

56. What is Suspense with Data Fetching?

In React's experimental concurrent mode, Suspense can be used with data fetching to pause rendering until data is ready, providing a smooth experience with loading states.

57. What are Compound Components?

Compound components are a pattern where related components work together to share state and behavior. They're used in cases like Tabs or Dropdowns where one main component (e.g., Tabs) controls its subcomponents (e.g., TabPanels).

58. What are Render Phases in React?

Render phases include the **render phase** (where React determines what changes to apply) and the **commit phase** (where changes are applied to the DOM). useLayoutEffect and useEffect are examples of hooks that operate in different phases.

59. What are Synthetic Events in React?

Synthetic events are React's cross-browser wrapper for native browser events. They normalize event properties, making them behave consistently across all browsers.

60. What is React Fiber?

React Fiber is a reimplementation of the React core algorithm that enables better control over rendering tasks, improving performance and allowing features like Suspense and concurrent rendering.