100+ Most Asked React.js QnA

Made By:

Yadneyesh (Curious Coder)
CodWithCurious.com

Find More PDFs on Our Telegram Channel
@Curious_Coder

1. **What is React.js?**
   React.js is a JavaScript library used for building user interfaces. It allows developers to create reusable UI components and efficiently update the UI when the data changes.
2. **What are the key features of React.js?**
   React.js offers features like virtual DOM, component-based architecture, one-way data flow, JSX syntax, and a rich ecosystem of libraries and tools.
3. **What is JSX?**
   JSX (JavaScript XML) is an extension to JavaScript that allows you to write HTML-like syntax within JavaScript code. It is used to describe the structure and appearance of React components.
4. **What is the significance of the virtual DOM in React.js?**
   The virtual DOM is a lightweight copy of the actual DOM. React uses the virtual DOM to optimize and speed up the process of updating the real DOM by comparing the current virtual DOM with the previous one.
5. **What is the difference between a functional component and a class component in React.js?**
   Functional components are stateless and are typically written as plain JavaScript functions. They are simpler and easier to test. Class components, on the other hand, have a state, can use lifecycle methods, and are written as ES6 classes.
6. **What is the purpose of the constructor in a React component?**
   The constructor is used to initialize the state and bind event handlers in a class component. It is called before the component is mounted.
7. **What is state in React.js?**
   State is an object that holds data and determines how a component renders and behaves. It is private and fully controlled by the component itself.
8. **What is the difference between state and props in React.js?**
   State is managed within a component and can be changed, while props are passed to a component from its parent and cannot be modified directly by the component receiving them.
9. **What is a controlled component?**
   A controlled component is a component where the form data is handled by React components. The React component that renders the form also controls what happens in that form on subsequent user input.
10. **What are React lifecycle methods?**
    React lifecycle methods are special methods that are called at specific points in a

component's lifecycle. These methods include componentDidMount, componentDidUpdate, componentWillUnmount, and many others.

11. **What is the significance of the render() method in React.js?**
    The render() method in React.js is responsible for returning the JSX that represents the structure and appearance of a component. It gets called whenever the component updates.
12. **What is the purpose of keys in React lists?**
    Keys are used to give a unique identity to each element in a list of components. They help React efficiently update and re-render the components by identifying which items have changed, been added, or removed.
13. **What is the context in React.js?**
    Context provides a way to pass data through the component tree without having to pass props manually at every level. It is used for sharing data that can be considered "global" for a tree of React components.
14. **What are higher-order components (HOCs) in React.js?**
    Higher-order components are functions that take a component and return a new component with additional functionality. They are used for code reuse, abstraction, and sharing common logic between components.
15. **What are React hooks?**
    Hooks are functions that allow you to use state and other React features without writing a class. They were introduced in React 16.8 to provide a simpler and more readable way to write React components.
16. **What are the basic rules of hooks?**
    Hooks have some rules: They must be used only at the top level of a function component, hooks mustbe called in the same order every time the component renders, and they cannot be called conditionally.
17. **What is the useState hook used for?**
    The useState hook is used to add state to functional components. It returns a stateful value and a function to update that value. By calling the update function, the component can trigger a re-render with the updated state.
18. **What is the useEffect hook used for?**
    The useEffect hook is used to perform side effects in functional components. It allows you to run code after the component has rendered and handle actions such as data fetching, subscriptions, or manually updating the DOM.
19. **What is the difference between useCallback and useMemo hooks?**
    useCallback is used to memoize functions, preventing unnecessary re-creation of functions on re-renders. useMemo is used to memoize values, caching the result of expensive calculations and avoiding recomputation.
20. **What is React Router?**
    React Router is a popular library for handling routing in React applications. It allows you to define different routes, render components based on the current URL, and navigate between different views in a single-page application.
21. **What is Redux?**
    Redux is a predictable state container for JavaScript applications. It provides a centralized store to manage application state and uses actions and reducers to modify and update that state in a predictable manner.
22. **What is the purpose of actions in Redux?**
    Actions in Redux are plain JavaScript objects that describe an event or user

interaction. They are dispatched to the store and trigger the corresponding reducer to update the state based on the action's type and payload.

23. **What are reducers in Redux?**
    Reducers in Redux are pure functions that specify how the application's state changes in response to actions. They take the current state and an action as input and return a new state based on that action.
24. **What is the connect function in Redux?**
    The connect function is used to connect a React component to the Redux store. It provides the component with access to the store's state and actions, allowing it to subscribe to changes and dispatch actions.
25. **What is the purpose of middleware in Redux?**
    Middleware in Redux provides a way to intercept and modify actions before they reach the reducers. It can be used for handling asynchronous actions, logging, and other tasks that need to be done outside the normal action flow.
26. **What is Redux Thunk?**
    Redux Thunk is a middleware for Redux that allows you to write action creators that return functions instead of plain action objects. This enables handling of asynchronous actions, such as API calls, inside action creators.
27. **What is React Native?**
    React Native is a framework for building native mobile applications using React. It allows developers to write mobile apps using JavaScript and leverage the power of React to create reusable UI components.
28. **What is the difference between React and React Native?**
    React is a JavaScript library for building user interfaces, primarily for web applications, while React Native is a framework for building native mobile applications. React Native uses native components and APIs specific to each platform.
29. **What are React Native components?**
    React Native components are similar to React components but are built specifically for mobile app development. They include components for handling user input, displaying data, navigating between screens, and more.
30. **What is the purpose of StyleSheet in React Native?**
    StyleSheet is a built-in component in React Native that allows you to define styles for your components. It provides a way to write styles using JavaScript objects or create reusable style constants.
31. **What is the difference between state and props in React Native?**
    The difference between state and props in React Native is the same as in React.js. State is managed within a component and can be changed, while propsare passed to a component from its parent and cannot be modified directly by the component receiving them.
32. **What is the purpose of AsyncStorage in React Native?**
    AsyncStorage is a simple, asynchronous, persistent key-value storage system provided by React Native. It allows you to store data on the device's disk and retrieve it later, making it useful for caching data or storing user preferences.
33. **What is the purpose of the Expo framework in React Native?**
    Expo is a set of tools, libraries, and services built on top of React Native. It provides a simplified development workflow, pre-configured native modules, and access to device features, allowing developers to build and deploy React Native apps faster.

34. **What is the purpose of the shouldComponentUpdate() method?**
    The shouldComponentUpdate() method is a lifecycle method in React that determines whether a component should re-render or not. By implementing this method and returning false under certain conditions, you can optimize the performance of your application.
35. **What is the React DevTools?**
    React DevTools is a browser extension that allows you to inspect and debug React component hierarchies. It provides a set of tools for inspecting components, examining props and state, and profiling performance.
36. **What is the purpose of the key prop in React?**
    The key prop is used to give a unique identity to each element in a list of components. It helps React efficiently update and re-render the components by identifying which items have changed, been added, or removed.
37. **What are the different ways to style components in React?**
    There are multiple ways to style components in React, including inline styles, using CSS classes with className, CSS-in-JS libraries like styled-components, and using preprocessor-based solutions like Sass or Less.
38. **What is the purpose of React Fragments?**
    React Fragments allow you to group multiple elements without adding an extra node to the DOM. They are useful when you need to return multiple elements from a component's render method without introducing unnecessary wrapping elements.
39. **What are controlled components in React forms?**
    Controlled components are form elements whose values are controlled by React state. The state is updated whenever the user interacts with the form, and the input values are set explicitly using React state, allowing for more control and validation.
40. **What is the purpose of the children prop in React?**
    The children prop is a special prop that allows you to pass components, elements, or text as children to other components. It enables the composition of components and the creation of more flexible and reusable component APIs.
41. **What is the difference between shallow rendering and full rendering in React testing?**
    Shallow rendering, using tools like Enzyme's shallow(), only renders the component itself, without rendering its child components. Full rendering, using tools like Enzyme's mount(), renders the full component tree, including child components.
42. **What are React portals?**
    React portals allow you to render children components into a different DOM subtree outside of the parent component's hierarchy. They are useful for cases where you need to render a component outside of its parent's DOM hierarchy, such as modal dialogs or tooltips.
43. **What is the purpose of the React.memo() function?**
    React.memo() is a higher-order component that memoizes the rendering of a functional component, similar to the shouldComponentUpdate() lifecycle method for class components. It prevents unnecessary re-renders of the component if its props have not changed.
44. **What are the differences between a controlled component and an uncontrolled component?**
    A controlled component is a component where form data is handled by React state

and is fully controlled by React. An uncontrolled component, on the other hand, manages its own stateand stores form data internally without relying on React state.

45. **What is the purpose of error boundaries in React?**
    Error boundaries are React components that catch JavaScript errors during rendering, in lifecycle methods, and in constructors of their child component tree. They help to prevent the entire application from crashing and allow for graceful error handling.
46. **What is the React.StrictMode?**
    React.StrictMode is a component that helps highlight potential problems in an application. It enables additional checks and warnings in the development mode to help identify and address potential bugs and deprecated features.
47. **What is the purpose of the React.Fragment component?**
    React.Fragment is a built-in component in React that allows you to group multiple elements without adding an extra node to the DOM. It is useful when you need to return multiple elements from a component's render method without introducing unnecessary wrapping elements.
48. **What is the significance of the "key" attribute when rendering an array of components?**
    The "key" attribute is used to give a unique identity to each element in an array of components. It helps React efficiently update and re-render the components by identifying which items have changed, been added, or removed.
49. **What is the useReducer hook in React?**
    The useReducer hook is a built-in hook in React that allows you to manage state using a reducer function. It is an alternative to useState and is useful for managing more complex state logic or state transitions.
50. **What is the purpose of the useContext hook in React?**
    The useContext hook is used to consume values from a React context. It allows functional components to access context values without nesting multiple layers of components or using render props.
51. **What is the difference between React and ReactDOM?**
    React is the library for building user interfaces, while ReactDOM is the package responsible for rendering React components into the DOM. ReactDOM provides methods like render() and hydrate() for rendering components.
52. **What is the purpose of the React.Fragment component?**
    React.Fragment is a built-in component in React that allows you to group multiple elements without adding an extra node to the DOM. It is useful when you need to return multiple elements from a component's render method without introducing unnecessary wrapping elements.
53. **What is the purpose of React.PureComponent?**
    React.PureComponent is a base class for components that performs a shallow comparison of props and state to determine whether a re-render is necessary. It can improve performance by avoiding unnecessary re-renders.
54. **What is the difference between React.createElement() and JSX?**
    React.createElement() is a method that creates a React element programmatically, while JSX is a syntax extension that allows you to write HTML-like code within JavaScript. JSX is compiled to React.createElement() calls.
55. **What is the purpose of React.createRef()?**
    React.createRef() is a method used to create a ref object, which can be attached to

React elements. It provides a way to access and interact with the underlying DOM nodes or React components.

56. **What is the purpose of the forwardRef() function?**
 The forwardRef() function is used to forward a ref from a higher-order component (HOC) to a wrapped component. It allows the wrapped component to receive a ref directly and access the underlying DOM node or React component.
57. **What is the purpose of React.lazy() and Suspense in React?**
 React.lazy() is a function that allows you to lazily load a component, which means the component is loaded only when it is actually needed. Suspense is a component that enables displaying fallback content while the lazy-loaded component is loading.
58. **What is the purpose of the useImperativeHandle() hook?**
 The useImperativeHandle() hook allows a functional component toexpose specific functions or values to the parent component through a ref. It is useful when you want to provide a more imperative API for a component that is primarily written as a functional component.
59. **What is the purpose of the useLayoutEffect() hook?**
 The useLayoutEffect() hook is similar to useEffect(), but it runs synchronously after all DOM mutations. It is useful when you need to perform operations that require access to the DOM immediately after React has performed all updates.
60. **What is the purpose of the useDebugValue() hook?**
 The useDebugValue() hook is used to display a custom label for custom hooks in React DevTools. It helps to provide more meaningful names for custom hooks when debugging and inspecting the component hierarchy.
61. **What is the purpose of the memo() function in React?**
 The memo() function is a higher-order component that memoizes the rendering of a functional component. It prevents unnecessary re-renders of the component if its props have not changed, similar to React.PureComponent for class components.
62. **What is the purpose of the create-react-app tool?**
 create-react-app is a command-line tool that sets up a new React application with a preconfigured development environment. It provides a simplified setup process and allows developers to start building React applications quickly.
63. **What is the purpose of the React Developer Tools extension?**
 The React Developer Tools extension is a browser extension that helps developers inspect and debug React component hierarchies. It provides a set of tools for inspecting components, examining props and state, and profiling performance.
64. **What is the purpose of the shouldComponentUpdate() method in React class components?**
 The shouldComponentUpdate() method is a lifecycle method in React class components that determines whether a component should re-render or not. By implementing this method and returning false under certain conditions, you can optimize the performance of your application.
65. **What is the purpose of the componentWillUnmount() method in React class components?**
 The componentWillUnmount() method is a lifecycle method in React class components that is called just before a component is unmounted and removed from the DOM. It allows for performing cleanup tasks such as removing event listeners or cancelling subscriptions.

66. **What is the purpose of the componentDidCatch() method in React class components?**
    The componentDidCatch() method is a lifecycle method in React class components that is called when an error is thrown in a child component. It allows the parent component to handle the error and display fallback UI instead of the crashed component.
67. **What is the purpose of the getDerivedStateFromProps() method in React class components?**
    The getDerivedStateFromProps() method is a static lifecycle method in React class components that is called when the props of a component change. It allows the component to update its state based on the new props, but it is used less frequently due to its complexity.
68. **What is the purpose of the getSnapshotBeforeUpdate() method in React class components?**
    The getSnapshotBeforeUpdate() method is a lifecycle method in React class components that is called right before the changes from a component update are flushed to the DOM. It allows the component to capture information from the DOM before it potentially changes.
69. **What is the purpose of the ReactDOMServer package in React?**
    The ReactDOMServer package provides server-side rendering APIs for React. It allows you to render React components on the server and send the resulting HTML to the client, enabling faster initial page loads and better SEO.
70. **What is the purpose of the ReactDOM.hydrate() method?**
    The ReactDOM.hydrate() method is similar to ReactDOM.render(), but it is used for rehydrating server-rendered HTML. It attaches event listeners and preserves the existing server-renderedmarkup and behavior while allowing React to take over the management of the component tree.
71. **What is the purpose of the useCallback() hook?**
    The useCallback() hook is used to memoize functions in functional components. It returns a memoized version of the callback function that only changes if one of the dependencies has changed. It is useful for optimizing performance in scenarios where functions are passed as props.
72. **What is the purpose of the useMemo() hook?**
    The useMemo() hook is used to memoize values in functional components. It allows you to cache the result of an expensive computation and only recalculate it when the dependencies have changed. It is useful for optimizing performance in scenarios where calculations are computationally expensive.
73. **What is the purpose of the useReducer() hook?**
    The useReducer() hook is used to manage state in functional components using the reducer pattern. It is an alternative to useState() and is suitable for managing more complex state or state transitions. It returns the current state and a dispatch function to update the state.
74. **What is the purpose of the useRef() hook?**
    The useRef() hook is used to create a mutable reference that persists across component renders. It returns a mutable object with a current property that can be used to store values or reference DOM nodes or other React elements.
75. **What is the purpose of the useLayoutEffect() hook?**
    The useLayoutEffect() hook is similar to useEffect(), but it runs synchronously after all

DOM mutations. It is useful when you need to perform operations that require access to the DOM immediately after React has performed all updates.

76. **What is the purpose of the useContext() hook?**
    The useContext() hook is used to consume values from a React context. It allows functional components to access context values without nesting multiple layers of components or using render props.
77. **What is the purpose of the useImperativeHandle() hook?**
    The useImperativeHandle() hook allows a functional component to expose specific functions or values to the parent component through a ref. It is useful when you want to provide a more imperative API for a component that is primarily written as a functional component.
78. **What is the purpose of the useDebugValue() hook?**
    The useDebugValue() hook is used to display a custom label for custom hooks in React DevTools. It helps to provide more meaningful names for custom hooks when debugging and inspecting the component hierarchy.
79. **What is the purpose of the useTransition() hook?**
    The useTransition() hook is used in React to coordinate the rendering of concurrent transitions or animations. It allows for smoother user experiences by delaying the rendering of new updates until the transitions/animations have completed.
80. **What is the purpose of the useQuery() hook in React Query?**
    The useQuery() hook is a part of the React Query library and is used to fetch and manage data in React components. It provides a declarative way to define and execute queries, handle caching, and manage the loading and error states of the data.
81. **What is the purpose of the useMutation() hook in React Query?**
    The useMutation() hook is a part of the React Query library and is used to perform mutations and manage the state of data updates in React components. It provides a declarative way to define and execute mutations, handle loading and error states, and update the cache.
82. **What is the purpose of the useInfiniteQuery() hook in React Query?**
    The useInfiniteQuery() hook is a part of the React Query library and is used to fetch and manage paginated data in React components. It allows you to load data incrementally as the user scrolls or performs actions, handling pagination and caching automatically.
83. **What isthe purpose of the useMutation() hook in React Query?**
    The useMutation() hook is a part of the React Query library and is used to perform mutations and manage the state of data updates in React components. It provides a declarative way to define and execute mutations, handle loading and error states, and update the cache.
84. **What is the purpose of the useInfiniteQuery() hook in React Query?**
    The useInfiniteQuery() hook is a part of the React Query library and is used to fetch and manage paginated data in React components. It allows you to load data incrementally as the user scrolls or performs actions, handling pagination and caching automatically.
85. **What is server-side rendering (SSR) in React?**
    Server-side rendering (SSR) is a technique where a React application is rendered on the server and sent to the client as HTML. This enables faster initial page loads and better search engine optimization (SEO) compared to client-side rendering (CSR).

86. **What are some benefits of using React for web development?**
    Some benefits of using React for web development include:
87. Component-based architecture for better code organization and reusability.
88. Efficient virtual DOM rendering for improved performance.
89. A large and active community with a rich ecosystem of libraries and tools.
90. Support for server-side rendering for improved SEO and initial load times.
91. Unidirectional data flow for easier debugging and state management.
92. **What are some limitations or challenges of using React?**
    Some limitations or challenges of using React include:
93. Steeper learning curve for beginners due to the need to learn JSX and React's component-based approach.
94. Tooling complexity, especially for more advanced features like server-side rendering or build optimization.
95. Performance concerns when managing large and complex component hierarchies.
96. Compatibility issues with older browsers that may not support modern JavaScript features used by React.
97. **How does React differ from other JavaScript frameworks like Angular or Vue?**
    React differs from other frameworks like Angular or Vue in several ways:
98. React focuses solely on the view layer and does not provide a complete solution for building applications.
99. React uses a virtual DOM for efficient rendering, while Angular and Vue use a real DOM.
100. React emphasizes a component-based architecture, making it easy to build reusable UI components.
101. React relies on JavaScript for defining components and logic, while Angular and Vue use templates and declarative directives.
102. **What are some best practices for optimizing performance in React applications?**
    Some best practices for optimizing performance in React applications include:
103. Using shouldComponentUpdate() or React.memo() to prevent unnecessary re-renders.
104. Memoizing expensive computations using useMemo() or useCallback().
105. Splitting components into smaller, more focused components for better reusability and performance.
106. Implementing lazy loading and code splitting to reduce initial load times.
107. Avoiding unnecessary state updates and optimizing data fetching and processing.
108. **How can you handle forms in React?**
    Forms in React can be handled by using controlled components or uncontrolled components. Controlled components store form data in React state and update the state on user input. Uncontrolled components store form data internally and retrieve it using refs or other DOM methods.
109. **How can you handle routing in React?**
    Routing in React can be handled using libraries like React Router. React Router allows you to define routes, map them to specific components, and handle navigation between different views in a React application.
110. **What is Redux and how does it work with React?**
    Redux is a state management library for JavaScript applications, and it can be used with React to manage application state. Redux provides a central store that holds the

entire application state, and React components can access the state and dispatch actions to update it.

111. **What isReact Native and how is it different from React?**
React Native is a framework for building native mobile applications using React. It allows developers to write mobile apps using JavaScript and leverage the power of React to create reusable UI components. While React is primarily used for web development, React Native is focused on mobile development and uses native components and APIs specific to each platform.
112. **What is the purpose of the useState() hook in React?**
The useState() hook is used to add state to functional components in React. It returns a stateful value and a function to update that value. By calling the update function, the component can trigger a re-render with the updated state.
113. **What is the purpose of the useEffect() hook in React?**
The useEffect() hook is used to perform side effects in functional components. It allows you to run code after the component has rendered and handle actions such as data fetching, subscriptions, or manually updating the DOM. The effect runs after every render unless dependencies are specified.
114. **What is the purpose of the useContext() hook in React?**
The useContext() hook is used to consume values from a React context. It allows functional components to access context values without nesting multiple layers of components or using render props.
115. **What is the purpose of the useReducer() hook in React?**
The useReducer() hook is used to manage state in functional components using the reducer pattern. It is an alternative to useState() and is suitable for managing more complex state or state transitions. It returns the current state and a dispatch function to update the state.
116. **What is the purpose of the useRef() hook in React?**
The useRef() hook is used to create a mutable reference that persists across component renders. It returns a mutable object with a current property that can be used to store values or reference DOM nodes or other React elements.
117. **What is the purpose of the useMemo() hook in React?**
The useMemo() hook is used to memoize values in functional components. It allows you to cache the result of an expensive computation and only recalculate it when the dependencies have changed. It is useful for optimizing performance in scenarios where calculations are computationally expensive.
118. **What is the purpose of the useCallback() hook in React?**
The useCallback() hook is used to memoize functions in functional components. It returns a memoized version of the callback function that only changes if one of the dependencies has changed. It is useful for optimizing performance in scenarios where functions are passed as props.