1. Q: What is React?
   A: React is a JavaScript library for building user interfaces.

2. Q: What is JSX?
   A: JSX (JavaScript XML) is a syntax extension for JavaScript used in React to describe the structure of UI elements.

3. Q: How do you render an element in React?
   A: Use the `ReactDOM.render()` function.

4. Q: What is a component in React?
   A: A component is a reusable and self-contained piece of UI.

5. Q: Explain the difference between functional components and class components.
   A: Functional components are written as functions and use hooks for state management, while class components use the traditional `class` syntax and lifecycle methods.

6. Q: What are props?
   A: Props (short for properties) are inputs to a React component.

7. Q: How do you pass data from parent to child components?
   A: By using props.

8. Q: What are state and setState?
   A: State represents the internal data of a component, and `setState` is a method used to update the state.

9. Q: What is the purpose of `setState` in React?
   A: `setState` is used to update the state of a component and trigger a re-render.

10. Q: What is a controlled component?
    A: A controlled component is an input element whose value is controlled by React state.

11. Q: Explain the concept of virtual DOM.
    A: Virtual DOM is an in-memory representation of the actual DOM elements, which React uses to optimize rendering.

12. Q: What is a key in React and why is it important?
    A: A key is a special string attribute used in lists to give each element a stable identity, helping React optimize rendering and updates.

13. Q: What are React hooks?
    A: Hooks are functions that let you "hook into" React state and lifecycle features from functional components.

14. Q: List some built-in React hooks.
A: `useState`, `useEffect`, `useContext`, `useReducer`, `useCallback`, `useMemo`, `useRef`, `useLayoutEffect`, `useImperativeHandle`, `useDebugValue`.

15. Q: What is the purpose of the `useEffect` hook?
A: `useEffect` is used to perform side effects in functional components, like data fetching, subscriptions, or DOM manipulation.

16. Q: How can you prevent unnecessary re-renders in React?
A: Use the `React.memo` higher-order component or the `useMemo` hook to memoize components.

17. Q: What is the role of the `shouldComponentUpdate` method?
A: It determines whether a component should re-render or not.

18. Q: What is context in React?
A: Context provides a way to pass data through the component tree without having to pass props down manually at every level.

19. Q: How do you create context in React?
A: By using the `React.createContext()` function.

20. Q: How can you consume context in a component?
A: Use the `useContext` hook or the `Consumer` component.

21. Q: What is the purpose of the `Fragment` component?
A: `Fragment` is used to group multiple elements without adding an extra DOM element.

22. Q: Explain the concept of "lifting state up."
A: Lifting state up refers to moving the state from a child component to its parent component to share data and behavior.

23. Q: What is the significance of the `key` prop in lists?
A: The `key` prop helps React identify which items have changed, been added, or been removed in a list, improving performance during updates.

24. Q: How can you conditionally render elements in React?
A: Use conditional statements like `if` or the ternary operator, and embed them in JSX.

25. Q: What is the purpose of the `map()` function in React?
A: The `map()` function is used to iterate over an array and generate a list of React elements.

26. Q: How do you handle forms in React?

A: Use controlled components by binding form inputs to state values.

27. Q: What is the role of the `key` prop in the `map()` function?
A: The `key` prop is used to give each element in an array a unique identity when rendering a list of elements.

28. Q: Explain the concept of "unidirectional data flow" in React.
A: Unidirectional data flow means that the data flows in a single direction, from parent components to child components.

29. Q: What are the lifecycle methods in class components?
A: `componentDidMount`, `componentDidUpdate`, `componentWillUnmount`, and others.

30. Q: How can you handle events in React?
A: By passing event handlers as props to components or by using the `onEvent` attribute directly in JSX.

31. Q: What is the purpose of the `useReducer` hook?
A: `useReducer` is an alternative to `useState` for managing more complex state logic.

32. Q: What is the purpose of the `useMemo` hook?
A: `useMemo` is used to memoize the result of a function, preventing unnecessary re-computation.

33. Q: How does React handle server-side rendering (SSR)?
A: React can render components on the server and send the generated HTML to the client for faster initial loading.

34. Q: Explain the term "component prop drilling."
A: Component prop drilling refers to passing props through multiple layers of components, which can become cumbersome in deeply nested component trees.

35. Q: What is the purpose of the `useCallback` hook?
A: `useCallback` is used to memoize functions, preventing unnecessary re-creation of function instances.

36. Q: How can you optimize performance in React?
A: Use techniques like memoization, shouldComponentUpdate, PureComponent, and React's built-in optimization features.

37. Q: What is a Higher Order Component (HOC)?
A: A Higher Order Component is a function that takes a component and returns a new component with additional functionality.

38. Q: What is the React Router library used for?
A: React Router is used for routing and navigation in React applications.

39. Q: How can you pass data between components that are not directly related?
A: Use a shared state management solution like React Context or a state management library like Redux.

40. Q: What is Redux, and why might you use it?
A: Redux is a predictable state management library often used in large-scale React applications to manage global state.

41. Q: What is the purpose of the `dispatch` function in Redux?
A: The `dispatch` function is used to send actions to the Redux store, triggering state changes.

42. Q: Explain the concept of "reducers" in Redux.
A: Reducers are pure functions that specify how the application's state changes in response to actions.

43. Q: How does React differ from Angular and Vue.js?
A: React is a library for building UI components, while Angular and

Vue.js are full-fledged frameworks that include additional features like routing and state management.

44. Q: What is the significance of the `exact` prop in React Router?
A: The `exact` prop is used to ensure that a route is matched only if the path is an exact match.

45. Q: How can you perform type checking in React?
A: You can use PropTypes (for class components) or TypeScript (for both class and functional components).

46. Q: What is the purpose of the `useContext` hook?
A: `useContext` is used to consume data from a context provider in functional components.

47. Q: How can you update state based on the previous state?
A: Use the functional form of `setState` or the `useReducer` hook to ensure that state updates are based on the latest state.

48. Q: What is a portal in React?
A: A portal is a way to render a component's children in a different part of the DOM, often used for modal dialogs or popovers.

49. Q: What is the significance of the `componentDidUpdate` lifecycle method?
A: `componentDidUpdate` is called after a component's update has been flushed to the DOM, allowing you to perform side effects.

50. Q: How can you optimize the performance of a React app?
A: Optimize rendering, minimize re-renders, use code splitting, lazy loading, and minimize the use of unnecessary state.

51. Q: What is the purpose of the `useLayoutEffect` hook?
A: `useLayoutEffect` is similar to `useEffect`, but it fires synchronously after all DOM mutations.

52. Q: How does React handle code splitting?
A: React supports code splitting using dynamic imports to split the application into smaller chunks that are loaded on-demand.

53. Q: What is the significance of the `shouldComponentUpdate` method?
A: `shouldComponentUpdate` is used to optimize rendering by preventing unnecessary updates.

54. Q: How do you handle errors in React?
A: Use the `componentDidCatch` lifecycle method in class components or the `ErrorBoundary` component.

55. Q: Explain the purpose of the `useRef` hook.
A: `useRef` is used to access or refer to a DOM element or a value that persists between renders.

56. Q: What is the difference between state and props?
A: State is internal to a component and can be changed, while props are external inputs passed to a component.

57. Q: How do you perform server-side rendering (SSR) in React?
A: Use libraries like Next.js or Razzle, which provide built-in support for SSR.

58. Q: What is React Native?
A: React Native is a framework for building mobile applications using React components that render to native mobile components.

59. Q: How can you handle asynchronous operations in React?
A: Use promises, async/await, or libraries like `axios` for data fetching and asynchronous operations.

60. Q: What is the purpose of the `dangerouslySetInnerHTML` attribute in React?

A: It is used to insert raw HTML content into a component, but it should be used with caution due to security risks.

61. Q: How does the `useState` hook work?
A: `useState` returns a state value and a function to update it, ensuring that the component re-renders when the state changes.

62. Q: What is the significance of the `React.Fragment` element?
A: `React.Fragment` is used to group multiple elements without introducing an additional wrapping element in the DOM.

63. Q: How can you prevent a component from rendering in React?
A: Return `null` or `false` from the component's render method.

64. Q: Explain the concept of reconciliation in React.
A: Reconciliation is the process of comparing the current virtual DOM with the previous virtual DOM to determine the most efficient way to update the actual DOM.

65. Q: What is the significance of the `children` prop in React?
A: The `children` prop is a special prop that allows you to pass components or elements as children to other components.

66. Q: How do you pass functions as props to child components?
A: You can pass functions as props just like any other data, and then call them from the child component.

67. Q: What is the purpose of the `useImperativeHandle` hook?
A: `useImperativeHandle` allows you to customize the instance value that is exposed when using the `ref` attribute on a functional component.

68. Q: How can you avoid using a state management library like Redux?
A: For simpler applications, you can manage state using React's built-in state management features such as `useState` and `useReducer`.

69. Q: What is the significance of the `componentWillUnmount` lifecycle method?
A: `componentWillUnmount` is called when a component is about to be removed from the DOM, allowing you to clean up resources or subscriptions.

70. Q: How can you optimize the performance of a React app?
A: Optimize rendering, minimize re-renders, use code splitting, lazy loading, and minimize the use of unnecessary state.

71. Q: What is the purpose of the `forwardRef` function in React?

A: `forwardRef` is used to pass a `ref` down to a child component, enabling the parent component to directly interact with the child's DOM node.

72. Q: How can you use conditional rendering in JSX?
A: Use ternary operators or logical operators to conditionally render JSX elements.

73. Q: What is the significance of the `React.memo` function?
A: `React.memo` is a higher-order component that memoizes the rendering of a functional component, preventing unnecessary re-renders.

74. Q: What is the purpose of the `useDebugValue` hook?
A: `useDebugValue` is used to display custom labels for custom hooks in React DevTools.

75. Q: How can you change the state of a parent component from a child component?
A: Pass a callback function from the parent to the child component as a prop, and call that function in the child to update the parent's state.

76. Q: What is the significance of the `componentWillReceiveProps` lifecycle method?
A: `componentWillReceiveProps` is called when a component receives new props, allowing you to update its internal state based on the new props.

77. Q: What is the purpose of the `useDocumentTitle` custom hook?
A: A `useDocumentTitle` hook could be created to update the document title based on a component's state or props.

78. Q: How does React handle performance optimization for large lists?
A: React supports the `VirtualizedList` component, which only renders visible items in the list, improving performance for large datasets.

79. Q: What is the significance of the `memo` function in React?
A: The `memo` function is used to memoize the rendering of a functional component, preventing unnecessary re-renders.

80. Q: How can you render HTML entities in React?
A: You can use JSX syntax for HTML entities, or

use the `dangerouslySetInnerHTML` attribute with proper security measures.

81. Q: What is the purpose of the `useLayoutEffect` hook?
A: `useLayoutEffect` is used to perform side effects synchronously after DOM mutations, similar to `componentDidMount` in class components.

82. Q: How can you handle routing and navigation in a React app?

A: Use libraries like React Router to manage routes and navigation between different parts of the app.

83. Q: What is the significance of the `componentWillUpdate` lifecycle method?
A: `componentWillUpdate` is called before a component's update is applied to the DOM, allowing you to perform preparations.

84. Q: How can you create a reusable custom hook in React?
A: Create a function that uses React hooks and returns state and functions, and then use that function as a custom hook in different components.

85. Q: What is the purpose of the `useReducer` hook?
A: `useReducer` is used to manage complex state logic and actions in a functional component.

86. Q: How can you integrate third-party libraries with React?
A: Use React components provided by the library, or create wrapper components to interface with the library's functionality.

87. Q: What is the significance of the `contextType` property in class components?
A: `contextType` is a property used to consume context in class components.

88. Q: How can you pass data between sibling components?
A: Lift the shared state up to a common ancestor component, or use state management solutions like context or Redux.

89. Q: What is the purpose of the `useEffect` hook?
A: `useEffect` is used to perform side effects, like data fetching or DOM manipulation, in functional components.

90. Q: How can you conditionally apply styles to a React component?
A: Use inline styles with ternary operators or conditional classes based on state or props.

91. Q: What is the significance of the `componentWillMount` lifecycle method?
A: `componentWillMount` is called before a component is rendered, allowing you to perform preparations.

92. Q: How can you handle user input and forms in React?
A: Use controlled components by binding form inputs to state values and handling changes through event handlers.

93. Q: What is the purpose of the `PropTypes` library?
A: `PropTypes` is used for type checking and validation of props in React components.

94. Q: How can you manage state in a large-scale React application?
A: Use state management libraries like Redux or MobX to centralize and manage the application's state.

95. Q: What is the significance of the `componentWillMount` lifecycle method?
A: `componentWillMount` is called before a component is rendered, allowing you to perform preparations.

96. Q: How can you update the document title in a React component?
A: Use the `document.title` property or a custom hook to update the title based on component state or props.

97. Q: What is the significance of the `componentDidMount` lifecycle method?
A: `componentDidMount` is called after a component is rendered and added to the DOM, allowing you to perform initial setup.

98. Q: How can you perform side effects after a component renders in React?
A: Use the `useEffect` hook to perform side effects after rendering in functional components, or lifecycle methods in class components.

99. Q: What is the purpose of the `propTypes` property in a React component?
A: `propTypes` is used for type checking and validation of props in class components.

100. Q: How can you optimize the rendering of a list of elements in React?
A: Use the `map()` function with unique `key` attributes, consider using the `VirtualizedList` component, and implement pagination or infinite scrolling.

101. Q: What is the significance of the `componentWillReceiveProps` lifecycle method?
A: `componentWillReceiveProps` is called when a component is about to receive new props, allowing you to update its internal state based on the new props.

102. Q: How can you handle internationalization (i18n) in a React application?
A: Use libraries like `react-intl` or create custom solutions to manage translations and language-specific content.

103. Q: What is the purpose of the `defaultProps` property in a React component?
A: `defaultProps` is used to define default values for props in case they are not provided when the component is used.

104. Q: How can you manage application routes and navigation using React Router?
A: Define different routes using the `Route` component, and use the `Link` or `NavLink` components for navigation between routes.

105. Q: Explain the concept of "lifting state up" and "prop drilling" in React.

A: "Lifting state up" refers to moving shared state from child components to a common ancestor, while "prop drilling" is the process of passing props through several intermediate components.

106. Q: What is the difference between controlled and uncontrolled components in React?
A: Controlled components have their state controlled by React, while uncontrolled components manage their own state internally.

107. Q: How can you pass data and functions to nested components without prop drilling?
A: Use the `Context API` or state management libraries like Redux to share data across components without manually passing props.

108. Q: What is the role of the `shouldComponentUpdate` method in class components?
A: `shouldComponentUpdate` allows you to optimize rendering by preventing unnecessary updates when certain conditions are met.

109. Q: How can you perform data fetching in React?
A: Use techniques like the `fetch` API, Axios, or third-party libraries to fetch data from APIs or servers.

110. Q: What is the significance of the `componentDidCatch` lifecycle method?
A: `componentDidCatch` is used to handle errors in a component's subtree and display fallback UI when errors occur.

111. Q: How can you use the `useContext` hook to consume context in a functional component?
A: Import the context created using `React.createContext` and use the `useContext` hook to access the context's values.

112. Q: What is the purpose of the `shouldComponentUpdate` method in class components?
A: `shouldComponentUpdate` is used to optimize rendering by allowing you to control when a component should re-render.

113. Q: How can you use React DevTools to inspect and debug components?
A: Install the React DevTools browser extension, which allows you to inspect component hierarchies, state, and props.

114. Q: What is the purpose of the `useState` hook?
A: `useState` is used to add state to functional components, allowing them to store and manage data that can change over time.

115. Q: How can you pass data between sibling components without using context or a state management library?
A: You can lift the shared state to a common parent component and pass the data as props to both sibling components.

116. Q: What is the purpose of the `useEffect` hook's second argument?
A: The second argument of `useEffect` is an array of dependencies, which determines when the effect should re-run based on changes in these dependencies.

117. Q: How can you handle forms and user input validation in React?
A: Use controlled components, form submission handlers, and validation libraries like `yup` or custom validation logic.

118. Q: What is the significance of the `dangerouslySetInnerHTML` prop in React?
A: `dangerouslySetInnerHTML` allows you to inject raw HTML content into a component, but it should be used with caution due to security risks.

119. Q: How can you create a custom hook in React?
A: Create a function that uses one or more built-in hooks, and return state or functions to be used in other components.

120. Q: What is the purpose of the `memo` function in React?
A: The `memo` function is used to memoize the rendering of a functional component, preventing unnecessary re-renders.

121. Q: How can you handle animations in a React application?
A: Use CSS transitions, libraries like `react-spring`, or animation libraries like `framer-motion` for creating animations.

122. Q: Explain the concept of "conditional rendering" in React.
A: Conditional rendering refers to rendering different components or elements based on certain conditions or state values.

123. Q: How does React handle performance optimization for large lists?
A: React uses techniques like the `VirtualizedList` component and the `windowing` approach to efficiently render only visible items in a large list.

124. Q: What is the significance of the `React.StrictMode` component?
A: `React.StrictMode` is used during development to highlight potential problems and perform checks to ensure best practices.

125. Q: How can you handle user authentication and authorization in a React app?
A: Use authentication libraries, state management, and secure API calls to manage user authentication and control access to certain components or routes.

126. Q: What is the purpose of the `children` prop in React components?
A: The `children` prop allows you to pass components, elements, or data as children to another component and access them within that component.

127. Q: How can you implement a loading state for data fetching in React?
A: Use component state to track the loading status and conditionally render loading indicators or content based on the state.

128. Q: What is the significance of the `key` prop in React lists?
A: The `key` prop is used to help React identify items in a list and optimize the rendering and updating process.

129. Q: How can you handle user interactions and events in React components?
A: Attach event handlers to DOM elements or React components using `onClick`, `onChange`, and other event attributes.

130. Q: What is the purpose of the `useHistory` hook in React Router?
A: The `useHistory` hook provides access to the history object, allowing you to navigate and manipulate the browser history programmatically.

131. Q: How can you implement lazy loading of components in React?
A: Use the `React.lazy()` function along with dynamic imports to load components only when they are needed.

132. Q: What is the role of the `unmountComponentAtNode` function in React?
A: The `unmountComponentAtNode` function is used to unmount a React component from a specified DOM element.

133. Q: How can you handle different screen sizes and responsiveness in a React app?
A: Use CSS media queries, CSS frameworks like Bootstrap, or libraries like `react-responsive` to create responsive designs.

134. Q: What is the purpose of the `useMemo` hook?
A: `useMemo` is used to memoize the result of a computation, preventing unnecessary re-computation in functional components.

135. Q: How can you implement infinite scrolling or pagination in a React app

?
A: Load additional data when the user scrolls to the bottom of a list or page, using techniques like `IntersectionObserver` or custom scroll event handlers.

136. Q: What is the significance of the `React.Fragment` element?
A: `React.Fragment` is used to group multiple elements without adding an additional DOM element, helping to keep the rendered output clean.

137. Q: How can you perform server-side rendering (SSR) in a React app?

A: Use libraries like Next.js or Razzle that provide built-in support for server-side rendering and rendering React components on the server.

138. Q: What is the role of the `propTypes` and `defaultProps` properties in a React component?
A: `propTypes` define the expected types and shapes of props, while `defaultProps` provide default values for props that are not provided.

139. Q: How can you optimize the rendering of a React component using `React.PureComponent`?
A: `React.PureComponent` automatically implements a shallow comparison of props and state to prevent unnecessary re-renders.

140. Q: What is the purpose of the `useReducer` hook in React?
A: `useReducer` is an alternative to `useState` for managing state in complex scenarios, using a reducer function similar to Redux reducers.

141. Q: How can you use the `useCallback` hook to optimize functional components?
A: `useCallback` is used to memoize functions to prevent re-creation on every render, optimizing performance in child components.

142. Q: What is the significance of the `useEffect` hook's cleanup function?
A: The cleanup function returned by `useEffect` is used to perform cleanup tasks when a component is unmounted or when dependencies change.

143. Q: How can you implement a loading spinner or indicator during data fetching?
A: Use conditional rendering to show a loading spinner while data is being fetched, and then display the content when the data is ready.

144. Q: What is the role of the `useContext` hook in React?
A: `useContext` is used to consume context data without nesting multiple context consumer components.

145. Q: How can you handle errors and exceptions in React components?
A: Use `try` and `catch` blocks for JavaScript errors, and use error boundaries or global error handling for React components.

146. Q: What is the significance of the `React.Fragment` element?
A: `React.Fragment` is used to group multiple elements without adding an extra DOM element to the rendered output.

147. Q: How can you pass additional props to a child component?
A: Pass additional props to a child component when rendering it, alongside its existing props.

148. Q: What is the purpose of the `key` prop in React components?
A: The `key` prop is used to help React identify and track individual components when rendering lists or performing updates.

149. Q: How can you perform data fetching and state management in React applications using GraphQL?
A: Use libraries like Apollo Client to manage data fetching, caching, and state management with GraphQL APIs.

150. Q: What is the role of the `async` and `await` keywords in asynchronous programming with React?
A: `async` and `await` simplify asynchronous code by allowing you to write asynchronous operations in a more synchronous-looking style.

151. Q: How can you prevent memory leaks in React components?
A: Properly clean up resources and subscriptions in the `componentWillUnmount` lifecycle method or using the `useEffect` cleanup function.

152. Q: What is the purpose of the `useMemo` hook in React?
A: `useMemo` is used to memoize the result of a computation, preventing unnecessary re-computation in functional components.

153. Q: How can you implement client-side routing in a React app using React Router?
A: Define routes and use the `Link` or `NavLink` components to navigate between different parts of the application.

154. Q: What is the significance of the `React.StrictMode` component?
A: `React.StrictMode` is used during development to highlight potential problems and perform checks to ensure best practices.

155. Q: How can you implement authentication and user login functionality in a React app?
A: Use authentication libraries, secure API calls, and state management to manage user authentication and access control.

156. Q: What is the role of the `useRef` hook in React?
A: `useRef` is used to access or reference DOM elements or values that persist across renders without causing re-renders.

157. Q: How can you use the `componentDidUpdate` lifecycle method in class components?
A: `componentDidUpdate` is used to perform side effects or update the DOM after a component has updated.

158. Q: What is the purpose of the `useEffect` hook in React?

A: `useEffect` is used to perform side effects, like data fetching or DOM manipulation, in functional components.

159. Q: How can you implement client-side routing with dynamic route parameters using React Router?
A: Define dynamic routes with parameters, access those parameters using `useParams` hook or `withRouter` HOC, and render content accordingly.

160. Q: What is the role of the `defaultProps` property in React components?
A: `defaultProps` provide default values for props that are not explicitly provided when using a component.

161. Q: How can you create reusable UI components in React?
A: Create modular components with well-defined props and functionality that can be reused across different parts of the application.

162. Q: What is the significance of the `useLayoutEffect` hook in React?
A: `useLayoutEffect` is used to perform side effects synchronously after the DOM is updated, similar to `componentDidMount` in class components.

163. Q: How can you handle user authentication and authorization in a React app?
A: Implement authentication flows, secure token management, and use protected routes to control access to certain parts of the app.

164. Q: What is the role of the `componentWillUnmount` lifecycle method in class components?
A: `componentWillUnmount` is used to perform cleanup tasks before a component is unmounted and removed from the DOM.

165. Q: How can you manage and share global state between components in React applications?
A: Use state management libraries like Redux, MobX, or React Context to manage global state and share data between components.

166. Q: What is the purpose of the `useEffect` hook's dependency array?
A: The dependency array defines the values that the effect depends on, and the effect will re-run whenever these values change.

167. Q: How can you implement authentication using JSON Web Tokens (JWT) in a React app?
A: Use a secure authentication flow, validate JWT tokens, and manage user sessions to implement authentication.

168. Q: What is the role of the `componentWillMount` lifecycle method in class components?
A: `componentWillMount` is called before a component is rendered, allowing you to perform setup tasks.

169. Q: How can you handle user input and form validation using controlled components?
A: Use controlled components to track form input values, validate them, and provide feedback to users based on the

validation results.

170. Q: What is the significance of the `useLayoutEffect` hook in React?
A: `useLayoutEffect` is used to perform side effects synchronously after the DOM is updated, similar to `componentDidMount` in class components.

171. Q: How can you optimize the rendering performance of a React app?
A: Optimize rendering by using pure components, memoization, and virtualization techniques for large lists or grids.

172. Q: What is the role of the `React.Fragment` element in React components?
A: `React.Fragment` is used to wrap multiple elements without introducing an additional DOM element, helping keep the rendered output clean.

173. Q: How can you manage and update global state in a React app without using a state management library?
A: Use React Context or prop drilling to pass global state down through the component tree and update it when needed.

174. Q: What is the purpose of the `useEffect` hook in React?
A: `useEffect` is used to perform side effects, like data fetching or DOM manipulation, in functional components.

175. Q: How can you handle routing and navigation in a React app using React Router?
A: Define routes and use the `Link` or `NavLink` components to navigate between different parts of the application.

176. Q: What is the role of the `shouldComponentUpdate` method in class components?
A: `shouldComponentUpdate` is used to optimize rendering by controlling when a component should re-render.

177. Q: How can you implement a toggle functionality in a React component?
A: Use state to track the toggled state, and update it using event handlers or functions like `setState`.

178. Q: What is the significance of the `key` prop in React lists?
A: The `key` prop is used to help React identify and update individual elements in a list efficiently.

179. Q: How can you create a custom hook in React?
A: Define a function that uses React hooks to encapsulate state and logic, and then reuse that function in different components.

180. Q: What is the purpose of the `useReducer` hook in React?
A: `useReducer` is used to manage more complex state logic and actions in a functional component.

181. Q: How can you optimize the performance of a React app using code splitting?
A: Use dynamic imports and code splitting to load only the necessary code chunks for specific routes or components.

182. Q: What is the role of the `componentWillUnmount` lifecycle method in class components?
A: `componentWillUnmount` is used to perform cleanup tasks before a component is unmounted and removed from the DOM.

183. Q: How can you use the `useContext` hook to consume context in a functional component?
A: Import the context created using `React.createContext` and use the `useContext` hook to access the context's values.

184. Q: What is the purpose of the `dangerouslySetInnerHTML` prop in React components?
A: `dangerouslySetInnerHTML` is used to insert raw HTML content into a component, but it should be used with caution due to security risks.

185. Q: How can you optimize the rendering performance of a React app?
A: Optimize rendering by using pure components, memoization, and virtualization techniques for large lists or grids.

186. Q: What is the role of the `React.StrictMode` component in React?
A: `React.StrictMode` is used during development to highlight potential problems and perform checks to ensure best practices.

187. Q: How can you handle user authentication and authorization in a React app?
A: Implement authentication flows, secure token management, and use protected routes to control access to certain parts of the app.

188. Q: What is the purpose of the `useEffect` hook in React?
A: `useEffect` is used to perform side effects, like data fetching or DOM manipulation, in functional components.

189. Q: How can you implement client-side routing in a React app using React Router?
A: Define routes and use the `Link` or `NavLink` components to navigate between different parts of the application.

190. Q: What is the significance of the `useMemo` hook in React?
A: `useMemo` is used to memoize the result of a computation, preventing unnecessary re-computation in functional components.

191. Q: How can you manage and share global state between components in React applications?
A: Use state management libraries like Redux, MobX, or React Context to manage global state and share data between components.

192. Q: What is the role of the `useLayoutEffect` hook in React?
A: `useLayoutEffect` is used to perform side effects synchronously after the DOM is updated, similar to `componentDidMount` in class components.

193. Q: How can you handle user authentication and authorization in a React app?
A: Implement authentication flows, secure token management, and use protected routes to control access to certain parts of the app.

194. Q: What is the role of the `componentWillUnmount` lifecycle method in class components?
A: `componentWillUnmount` is used to perform cleanup tasks before a component is unmounted and removed from the DOM.

195. Q: How can you manage and update global state in a React app without using a state management library?
A: Use React Context

or prop drilling to pass global state down through the component tree and update it when needed.

196. Q: What is the purpose of the `useEffect` hook in React?
A: `useEffect` is used to perform side effects, like data fetching or DOM manipulation, in functional components.

197. Q: How can you handle routing and navigation in a React app using React Router?
A: Define routes and use the `Link` or `NavLink` components to navigate between different parts of the application.

198. Q: What is the role of the `shouldComponentUpdate` method in class components?
A: `shouldComponentUpdate` is used to optimize rendering by controlling when a component should re-render.

199. Q: How can you implement a toggle functionality in a React component?
A: Use state to track the toggled state, and update it using event handlers or functions like `setState`.

200. Q: What is the significance of the `key` prop in React lists?
A: The `key` prop is used to help React identify and update individual elements in a list efficiently.

Feel free to use these questions and answers to deepen your understanding of React or prepare for interviews!