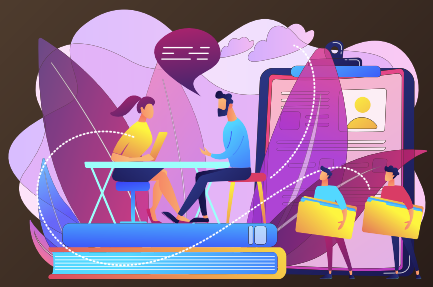
Top 50 React Js Interview QnA



1. What is React JS?

- React.js is an open-source JavaScript library used for building user interfaces. It allows developers to create reusable UI components and efficiently update and render them in response to data changes.

2. What are the key features of React JS?

- Key features of React.js include:
  - Virtual DOM for efficient rendering
  - Component-based architecture
  - JSX syntax for combining JavaScript and HTML
  - Unidirectional data flow
  - React Native for mobile app development
  - React Router for managing application routing

3. What is JSX in React.js?

- JSX (JavaScript XML) is a syntax extension for JavaScript used in React.js to describe the structure of the user interface. It allows developers to write HTML-like code within JavaScript.

4. What is the difference between state and props in React.js?

- State is an internal data store of a component that can be changed over time. Props (short for properties) are passed to a component from its parent and cannot be changed by the component itself.

5. What is a functional component in React.js?

- A functional component, also known as a stateless component, is a simpler way of defining a component in React.js. It is a JavaScript function that takes props as input and returns JSX elements.

6. What is a class component in React.js?

- A class component is a more traditional way of defining a component in React.js. It is a JavaScript class that extends the React.Component class and implements a render() method to return JSX elements.

7. What is the purpose of setState() in React.js?

- The setState() method is used to update the state of a component in React.js. It allows you to update the component's state and trigger a re-render of the component and its child components.

8. What is the difference between controlled and uncontrolled components in React.js?

- In controlled components, the value of form elements is controlled by React.js state. Changes to the form elements are handled by updating the state. In uncontrolled components, the form elements manage their own state internally.

9. What is the significance of keys in React.js?

- Keys are used in React.js to identify unique elements in a collection of elements. They help React optimize the rendering process by efficiently updating and reordering the elements.

10. What is the purpose of componentDidMount() lifecycle method in React.js?

- The componentDidMount() method is called immediately after a component is rendered for the first time. It is often used to perform initialization tasks, such as fetching data from an API.

11. What is the difference between props and state?

- Props (short for properties) are used to pass data from a parent component to its child components. They are read-only and cannot be modified by the child components. State is used to manage internal data in a component and can be changed over time using the setState() method.

12. Explain the concept of Virtual DOM in React.js.

- The Virtual DOM is a virtual representation of the actual DOM (Document Object Model) in memory. React.js uses the Virtual DOM to efficiently update and render components. When the state of a component changes, React calculates the difference between the new and old Virtual DOM and updates only the necessary parts of the actual DOM, resulting in better performance.

13. What is the purpose of shouldComponentUpdate() method in React.js?

- The should ComponentUpdate() method is called before a component is re-rendered. It allows you to control whether a component should update or not by returning true or false. Implementing this method can optimize the performance of your application by preventing unnecessary re-renders.

14. What are React Hooks?

- React Hooks are functions introduced in React 16.8 that allow you to use state and other React features in functional components. They provide a way to manage state and perform side effects without writing a class component.

15. Explain the difference between useState() and useEffect() hooks.

- useState() is a hook used to add state to functional components. It returns a stateful value and a function to update that value. useEffect() is a hook used to perform side effects in functional components. It allows you to run code after rendering and handle lifecycle events.

16. What is the purpose of the useReducer() hook?

- The useReducer() hook is used for state management in React. It is an alternative to useState() and provides a way to update complex state by specifying how the state should change based on the previous state and an action.

17. What is React Router?

- React Router is a popular library for managing routing in React.js applications. It allows you to define different routes and their corresponding components, enabling navigation between different parts of the application.

18. What is the significance of exact in React Router?

- The exact keyword is used in React Router to ensure that a route matches exactly and does not partially match other routes. It is used to avoid rendering multiple components when there is a partial route match.

19. What are Higher-Order Components (HOCs) in React.js?

- Higher-Order Components (HOCs) are functions that take a component and return a new enhanced component. They are used to share common functionality between multiple components without duplicating code.

20. What is Redux and how does it work with React.js?

- Redux is a predictable state management library for JavaScript applications, including React.js. It provides a centralized store to manage the state of an application and enables components to access and update the state using actions and reducers.

21. What are the three principles of Redux?

- The three principles of Redux are:
  - Single source of truth: The state of the whole application is stored in a single store.
  - State is read-only: The state can only be modified by dispatching actions.
  - Changes are made with pure functions: Reducers are pure functions that specify how the state should change based on actions.

22. What is the role of the mapStateToProps() function in Redux?

- The mapStateToProps() function is used to specify which parts of the Redux state should be passed to a connected component as props. It allows the component to access the required state data.

23. What is the role of the mapDispatchToProps() function in Redux?

- The mapDispatchToProps() function is used to specify which actions should be available in a connected component as props. It allows the component to dispatch the required actions.

24. What is Redux Thunk?

- Redux Thunk is a middleware for Redux that allows you to write asynchronous logic using functions called "thunks." Thunks are functions that can be dispatched and can contain asynchronous API calls and other side effects.

25. What is the purpose of the render() method in React.js?

- The render() method is responsible for rendering JSX elements to the DOM. It is called whenever the state or props of a component change.

26. What is the significance of the key attribute in React.js?

- The key attribute is used in React.js to give a unique identity to each element in an array of elements. It helps React efficiently update and reorder the elements when the array changes.

27. Explain the concept of "unidirectional data flow" in React.js.

- Unidirectional data flow is a pattern followed in React.js where the data flows in a single direction, from parent components to child components. It helps maintain a predictable and manageable state throughout the application.

22. What is the role of the mapStateToProps() function in Redux?

- The mapStateToProps() function is used to specify which parts of the Redux state should be passed to a connected component as props. It allows the component to access the required state data.

23. What is the role of the mapDispatchToProps() function in Redux?

- The mapDispatchToProps() function is used to specify which actions should be available in a connected component as props. It allows the component to dispatch the required actions.

24. What is Redux Thunk?

- Redux Thunk is a middleware for Redux that allows you to write asynchronous logic using functions called "thunks." Thunks are functions that can be dispatched and can contain asynchronous API calls and other side effects.

25. What is the purpose of the render() method in React.js?

- The render() method is responsible for rendering JSX elements to the DOM. It is called whenever the state or props of a component change.

26. What is the significance of the key attribute in React.js?

- The key attribute is used in React.js to give a unique identity to each element in an array of elements. It helps React efficiently update and reorder the elements when the array changes.

27. Explain the concept of "unidirectional data flow" in React.js.

- Unidirectional data flow is a pattern followed in React.js where the data flows in a single direction, from parent components to child components. It helps maintain a predictable and manageable state throughout the application.

28. What are React Fragments?

- React Fragments allow you to group multiple elements without adding extra nodes to the DOM. They are useful when you need to return multiple elements from a component's render method without wrapping them in a parent element.

29. What is the purpose of the dangerouslySetInnerHTML property in React.js?

The dangerouslySetInnerHTML property is used to insert raw HTML content into a React component. It is often used when you need to render HTML content received from an external source, but it should be used with caution to avoid security vulnerabilities.

30. What are React hooks rules or guidelines?

Some key rules and guidelines for using React hooks include:

- Hooks should only be used in functional components, not in regular JavaScript functions or class components.
- Hooks should always be called at the top level of a component, not inside loops, conditions, or nested functions.
- Hooks should be called in the same order in every render cycle to ensure consistency.

31. How can you optimize performance in React.js?

Some techniques to optimize performance in React.js include:

- Using React's built-in shouldComponentUpdate or PureComponent to avoid unnecessary re-renders.
- Implementing code splitting and lazy loading to load components only when needed.
- Using React.memo to memoize functional components.
- Properly utilizing keys when rendering lists of elements.
- Minimizing the use of inline function declarations in component props.

32. What is the purpose of the useContext() hook in React.js?

The useContext() hook is used to consume values from a React context. It allows functional components to access the value provided by a context without wrapping the component in a context consumer.

33. What is the purpose of the React.memo() function?

The React.memo() function is a higher-order component used to memoize functional components. It prevents unnecessary re-renders by caching the result of a component's render and reusing it when the component's props have not changed.

34. What is the concept of code splitting in React.js?

Code splitting is a technique used to split a large JavaScript bundle into smaller chunks that can be loaded on-demand. It helps reduce the initial loading time of an application by only loading the code that is needed for the current page or feature.

35. How can you handle forms in React.js?

The **useEffect()** hook is commonly used for handling side effects in React.js. It allows you to perform actions such as data fetching, subscriptions, or manually updating the DOM after the component has rendered.

36. What is the purpose of the useCallback() hook in React.js?

The useCallback() hook is used to memoize a function and prevent unnecessary re-creation of the function on each render. It is particularly useful when passing callbacks to child components, as it ensures that the callbacks are not re-created unless their dependencies change.

37. How can you optimize the performance of React.js applications for mobile devices?

To optimize the performance of React.js applications for mobile devices, you can:

- Use the React.lazy() function and code splitting to load components on demand.
- Implement responsive design techniques to ensure the UI adapts to different screen sizes.
- Optimize images and other media assets for mobile devices.
- Minimize network requests and reduce the size of JavaScript bundles.
- Use performance analysis tools like Lighthouse to identify and resolve bottlenecks.

38. What is the purpose of the forwardRef() function in React.js?

The forwardRef() function is used to forward a ref from a parent component to a child component. It allows the child component to access and interact with a DOM element or a React component created by the parent.

39. What is the role of the React.Fragment component?

The React.Fragment component is an alternative to using empty tags (<>...</>) when you need to render multiple elements without a wrapping parent element. It allows you to avoid adding extra nodes to the DOM.

40. What is the purpose of the useMemo() hook in React.js?

The **useMemo()** hook is used to memoize a value and prevent unnecessary re-computation. It takes a function and a dependency array and returns a memoized value that is only recalculated when the dependencies change.

41. What are the advantages of using React.js for building web applications?

Advantages of using React.js include:

- Efficient rendering using the Virtual DOM.
- Component-based architecture for reusability and maintainability.
- A large and active community with plenty of resources and libraries.
- Support for server-side rendering (SSR) for better performance and SEO.
- Easy integration with other libraries and frameworks.

42. What are the limitations of React.js?

Some limitations of React.js include:

- Steeper learning curve for beginners.
- Lack of clear guidelines for project structure and architecture.
- Performance concerns with large and complex applications.
- Limited built-in support for internationalization (i18n) and accessibility (a11y).
- React.js is focused on the view layer, so you need additional libraries for routing, state management, and other functionalities.

43. What is the purpose of the React.StrictMode component?

The React.StrictMode component is used to enable strict mode for React.js. It performs additional runtime checks and warnings to help identify potential problems in the application. It is mainly used during development and is not meant for production.

44. How can you handle errors in React.js?

- In React.js, you can handle errors using error boundaries. Error boundaries are React components that catch JavaScript errors anywhere in their child component tree and display fallback UI instead of crashing the entire application.

45. What is the purpose of the useRef() hook in React.js?

- The useRef() hook is used to create a mutable reference that persists across re-renders of a component. It is commonly used to access and modify DOM elements, store mutable values, or create a reference to a child component.

46. What is React Context and when should you use it?

- In class components, side effects can be handled using lifecycle methods such as componentDidMount(), componentDidUpdate(), and componentWillUnmount(). These methods provide hooks to perform actions like data fetching, subscriptions, or cleaning up resources.

47. What is the purpose of React DevTools?

- React DevTools is a browser extension that allows developers to inspect, debug, and profile React components and their state. It provides a visual representation of the component tree, along with the ability to view and manipulate component props and state.

48. What are the differences between a controlled component and an uncontrolled component in React.js forms?

- In a controlled component, the form data is controlled by React state. Any changes to the input fields are handled by updating the state, and the value of the input fields is set based on the state. In contrast, an uncontrolled component allows the form data to be handled by the DOM, without directly involving React state.

49. What is the purpose of React portals?

- React portals provide a way to render children components into a different DOM node outside the parent component's hierarchy. This allows components to be rendered at a different location in the DOM tree, which is useful for scenarios like modal dialogs or tooltips.

50. What is the concept of reconciliation in React.js?

- Reconciliation is the process through which React updates the user interface to reflect changes in component state or props. React compares the previous and current Virtual DOM representations and calculates the minimal changes required to update the actual DOM, resulting in optimal rendering performance.