

React Interview Questions with answer

Topics : [React JS](#)

Written on [January 03, 2024](#)

React Basics:

1. What is React?

- **Answer:** React is a JavaScript library for building user interfaces, developed by Facebook.

2. Explain the Virtual DOM in React.

- **Answer:** The Virtual DOM is a lightweight copy of the actual DOM. React uses it to improve performance by updating only the parts of the DOM that have changed.

3. What are JSX and how does it differ from HTML?

- **Answer:** JSX is a syntax extension for JavaScript recommended by React. It looks similar to XML/HTML but gets compiled to JavaScript. JSX allows mixing HTML with JavaScript.

4. How do you create a React component?

- **Answer:** Components can be created using either class components or functional components. For example:

```
// Functional component
function MyComponent() {
  return <div>Hello, React!</div>;
}
```

```
// Class component
class MyComponent extends React.Component {
  render() {
    return <div>Hello, React!</div>;
  }
}
```

5. What is the significance of the ReactDOM.render method?

- **Answer:** ReactDOM.render is used to render React elements into the DOM. It takes a JSX element and a target container as parameters.

6. What is the purpose of setState in React?

- **Answer:** setState is used to update the state of a component, triggering a re-render. It can take an object or a function returning an object to update state asynchronously.

7. Explain the difference between state and props.

- **Answer:** State is internal to a component and can be changed. Props are passed from a parent component and are immutable in the child component.

8. How does React handle input data binding?

- **Answer:** React uses a one-way data binding. Data flows from parent to child through props. For two-way binding, state and event handlers are used.

9. What is a controlled component in React?

- **Answer:** A controlled component is a component whose form elements are controlled by React state. The state is used to control the input value.

10. What are React hooks, and why were they introduced?

- **Answer:** React hooks are functions that let you use state and lifecycle features in functional components. They were introduced to enable the use of state and side-effects in functional components.

11. Explain the useEffect hook in React.

- **Answer:** useEffect is a hook used for side effects in functional components. It replaces lifecycle methods like componentDidMount and componentDidUpdate.

12. Describe the difference between class components and functional components.

- **Answer:** Class components are ES6 classes and have a render method. Functional components are simpler, use functions, and are introduced in React 16.8 with the introduction of hooks.

Component Lifecycle:

13. Describe the lifecycle methods of a class component.

- **Answer:** Class components have lifecycle methods like componentDidMount, componentDidUpdate, and componentWillUnmount. These methods are invoked at different stages of a component's lifecycle.

React Hooks:

14. What is the purpose of the useState hook?

- **Answer:** useState is a hook that allows functional components to have state. It returns an array with the current state value and a function to update the state.

15. **Explain the useEffect hook.**

- **Answer:** useEffect is used for side effects in functional components. It's similar to componentDidMount, componentDidUpdate, and componentWillUnmount in class components.

16. **When would you use the useEffect hook in React?**

- **Answer:** useEffect is used when you need to perform side effects in functional components, such as data fetching, subscriptions, or manually changing the DOM.

State Management:

17. **What is Redux?**

- **Answer:** Redux is a state management library for JavaScript applications. It helps manage the state of an application in a predictable way.

18. **How does Redux differ from local component state?**

- **Answer:** Redux provides a global state that can be accessed by any component. Local component state is limited to the component it belongs to.

19. **What is the purpose of the Redux store?**

- **Answer:** The Redux store is a single source of truth for the state of an application. It holds the complete state tree of the application.

20. **Explain the concept of actions in Redux.**

- **Answer:** Actions are payloads of information that send data from the application to the Redux store. They are plain JavaScript objects with a type property.

21. **What is a reducer in Redux?**

- **Answer:** A reducer is a pure function that specifies how the application's state changes in response to an action. It takes the current state and an action as arguments and returns the new state.

22. **What is the purpose of the connect function in React Redux?**

- **Answer:** The connect function is used to connect a React component to the Redux store. It takes mapStateToProps and mapDispatchToProps functions as arguments.

React Router:

23. **What is React Router?**

- **Answer:** React Router is a standard library for routing in React applications. It enables navigation among views of the application.

24. **How do you implement routing in a React application using React Router?**

- **Answer:** Install React Router using `npm install react-router-dom` and then use components like `BrowserRouter`, `Route`, and `Link` to define routes in your application.

25. **Explain the purpose of the Route component in React Router.**

- **Answer:** The `Route` component renders content based on the current location's pathname. It is used to define what should be rendered for a particular route.

26. **What is the purpose of the Link component in React Router?**

- **Answer:** The `Link` component is used to navigate between views in a React Router application. It renders an anchor tag and prevents the full page reload.

Styling in React:

27. **How can you apply styles to React components?**

- **Answer:** Styles can be applied using inline styles, separate stylesheets, CSS-in-JS libraries, or preprocessor solutions like Sass or Less.

28. **What is CSS-in-JS?**

- **Answer:** CSS-in-JS is an approach where styles are defined within JavaScript files. It allows dynamic styles based on props or state.

React Forms:

29. **How do you handle forms in React?**

- **Answer:** Forms in React are handled by state. You can use controlled components, where the form elements are controlled by React state.

30. **Explain the concept of controlled components in React forms.**

- **Answer:** Controlled components are form elements whose value is controlled by React state. The state is updated via event handlers, and the value is passed as a prop.

31. **What is the purpose of the onChange event in React forms?**

- **Answer:** The `onChange` event is triggered when the value of a form element changes. It is commonly used to update the state in controlled components.

React Hooks:

32. **What is the useCallback hook used for?**

- **Answer:** `useCallback` is used to memoize a function, preventing it from being recreated on each render.

33. **Explain the useMemo hook.**

- **Answer:** `useMemo` is used to memoize a value based on a computation. It helps avoid

unnecessary recalculations.

34. What is the purpose of the `useContext` hook?

- **Answer:** `useContext` is used to access the value of a context directly within a functional component.

React Testing:

35. How can you perform unit testing in React?

- **Answer:** Unit testing in React can be done using testing libraries like Jest along with testing utilities like React Testing Library or Enzyme.

36. Explain snapshot testing in Jest.

- **Answer:** Snapshot testing in Jest captures the output of a component and saves it to a file. On subsequent runs, Jest compares the output to the saved snapshot.

React Context:

37. What is React Context?

- **Answer:** React Context provides a way to share values (such as themes, user authentication status, etc.) across components without explicitly passing props.

38. How do you use React Context?

- **Answer:** Create a context using `React.createContext()`, provide it at a higher level in the component tree using `Context.Provider`, and consume the context value using `Context.Consumer` or the `useContext` hook.

React Fragments:

39. What are React Fragments?

- **Answer:** React Fragments are used to group multiple elements without introducing an additional parent element to the DOM.

40. How do you use React Fragments?

- **Answer:** You can use empty angle brackets (`<>` `</>`) or `<React.Fragment>` to wrap multiple elements without adding extra nodes to the DOM.

React Portals:

41. What are React Portals used for?

- **Answer:** React Portals allow rendering children into a DOM node that exists outside the hierarchy of the parent component.

42. How do you create a React Portal?

- **Answer:** Use the `ReactDOM.createPortal` method to render content outside the normal React component tree.

Performance Optimization:

43. What are the key performance optimization techniques in React?

- **Answer:** Techniques include using `React.memo`, `shouldComponentUpdate`, `PureComponent`, and optimizing render methods.

44. Explain the concept of code splitting in React.

- **Answer:** Code splitting is a technique to split your code into smaller pieces, allowing you to load only the necessary parts when needed, improving application performance.

Higher-Order Components (HOCs):

45. What is a Higher-Order Component (HOC) in React?

- **Answer:** A Higher-Order Component is a function that takes a component and returns a new component with additional props or behavior.

46. Why would you use a Higher-Order Component?

- **Answer:** HOCs are used for code reuse, cross-cutting concerns, and adding functionality to components without modifying their source code.

Error Handling:

47. How do you handle errors in React components?

- **Answer:** Use the `componentDidCatch` lifecycle method for class components or the `ErrorBoundary` component in React 16 and above.

React Hooks:

48. What is the purpose of the `useReducer` hook?

- **Answer:** `useReducer` is used to manage more complex state logic in functional components. It is often used as an alternative to `useState` when state transitions are more complex.

Context API:

49. What are the limitations of the Context API?

- **Answer:** Context API can cause unnecessary re-renders when using a global state, and it may not perform well for frequent updates.

Testing Library:

50. What is the React Testing Library, and how is it different from Enzyme?

- **Answer:** React Testing Library is designed to encourage testing the application from the

user's perspective. Enzyme, on the other hand, provides more utilities for interacting with and inspecting React components.

Server-Side Rendering (SSR):

51. What is Server-Side Rendering (SSR) in React?

- **Answer:** SSR is the process of rendering React components on the server rather than the client, resulting in a fully rendered page being sent to the browser.

52. What are the benefits of Server-Side Rendering in React?

- **Answer:** SSR improves initial page load performance, search engine optimization (SEO), and provides a better user experience.

React Hooks:

53. Explain the useRef hook in React.

- **Answer:** useRef creates a mutable object with a current property. It is often used to persist values across renders without causing re-renders.

54. What is the difference between useRef and useState in React?

- **Answer:** useState is used for managing state and causes re-renders. useRef is used for mutable values that do not trigger re-renders.

Redux Middleware:

55. What is Redux Middleware?

- **Answer:** Middleware in Redux is a way to extend the behavior of the store. It intercepts actions before they reach the reducer and can perform asynchronous tasks.

56. Explain the purpose of Redux Thunk.

- **Answer:** Redux Thunk is middleware for Redux that allows writing asynchronous logic in action creators. It enables dispatching functions instead of plain action objects.

Code Splitting:

57. What is dynamic import in JavaScript, and how is it used for code splitting?

- **Answer:** Dynamic import is a JavaScript feature that allows importing modules asynchronously. It is used for code splitting to load parts of the application only when needed.