

# REACT INTERVIEW QUESTIONS

👉 Q1. What is React?

Ans: React is an open-source JavaScript library for building user interfaces or UI components, developed by Facebook.

👉 Q2. What is JSX

Ans: JSX (JavaScript XML) is a syntax extension for JavaScript recommended by React for describing what the UI should look like.

👉 Q3. What is the virtual DOM?

Ans: The virtual DOM is a lightweight copy of the actual DOM in memory. React uses it to improve performance by updating only the changed parts of the actual DOM.

👉 Q4. What are state and props in React?

Ans: State is an internal data store that belongs to a specific component, and it can be changed over time. Props are properties passed to a component from its parent, and they are immutable.

👉 Q5. What is the difference between state and props?

Ans: State is internal to a component and can be changed over time, while props are external and passed to a component.

👉 Q6. What is the purpose of virtual DOM in React.js?

Ans: The virtual DOM is a lightweight copy of the actual DOM. React uses it to improve performance by minimizing direct manipulation of the DOM. Instead, React creates two copies of the virtual DOM (a new one and an old one). React compares the new virtual DOM with the old copy of the virtual DOM, then reflects only the changes to the actual DOM, updating only the parts that have changed.

👉 Q7. How does React handle data flow?

Ans: React follows a unidirectional data flow, meaning data flows in one direction—from parent to child components. This helps maintain a clear and predictable data flow, making the application easier to understand and debug.

👉 Q8. What are React Hooks?

Ans: React Hooks are functions that enable functional components to use state and other React features without writing a class. They allow developers to use state and other React features in functional components, which simplifies code and promotes reusability. We can not use hooks in class components.

👉 Q9. What is the significance of keys in React lists?

Ans: Keys are special attributes used by React to identify which items have changed, been added, or been removed in a list. They help React efficiently update the UI by minimizing re-renders of list items.

👉 Q10. What are the advantages of using React.js?

Ans: React.js offers several advantages, including improved performance with virtual DOM, reusability and modularity with component-based architecture, enhanced developer experience with JSX syntax, and strong community support and ecosystem.

👉 Q11. What is the difference between controlled and uncontrolled components?

Ans: Controlled components are those whose form elements (like input, textarea, select) are controlled by React state, while uncontrolled components are directly controlled by the DOM.

👉 Q12. What is the purpose of setState in React?

Ans: setState is used to update the state of a component and trigger a re-render. We use the setState method in class components, while in functional components, we use a hook called useState that gives us a setter function to update the state.

👉 Q13. What is React Router?

Ans: React Router is a library that enables navigation among views in a React application, allowing for the development of single-page applications. It allows you to define routes, nested routes, and dynamic routes in your application. and also ensures that the page does not reload after we navigate to another.

👉 Q14. Explain the useEffect hook.

Ans: The useEffect hook in React is used for side effects in functional components, such as data fetching, subscriptions, or manually changing the DOM. It replaces lifecycle methods like componentDidMount(), componentDidUpdate(), and componentWillUnmount().

👉 Q15. What is Redux, and why is it used?

Ans: Redux is a global state management library for JavaScript applications, commonly used with React. It helps manage the state of an application in a predictable, more organized and centralized way.

👉 Q16. Explain the concept of lifting state up in React.

Ans: Lifting state up means moving the state of a component to its parent component so that multiple child components can share and update that state. This helps in managing state more efficiently across components.

👉 Q17. Explain the concept of higher-order components (HOCs) in React.

Ans: Higher-order components are functions that take a component and return a new component with additional functionalities. They're used to share common functionalities between components without repeating code.

👉 Q18. What is the purpose of the memo function in React?

Ans: The memo function is used to optimize functional components by memoizing them. It helps prevent unnecessary re-renders by caching the result of the component's render function.

👉 Q19. What is the purpose of the context API in React?

Ans: Context API is used for managing global state in React applications. It allows passing data through the component tree without having to pass props manually at every level.

👉 Q20. Explain the concept of hooks in React and provide examples of some commonly used hooks.

Ans: Hooks are functions that allow you to use React features like state and lifecycle methods in functional components. Some commonly used hooks include:

useState: Allows functional components to have state.

useEffect: Performs side effects in functional components, like fetching data or subscribing to events.

useContext: Allows functional components to consume context.

👉 Q21. What is the significance of the Fragment element in React?

Ans: The Fragment element in React JS is like a special container that helps group multiple elements together without adding extra nodes to the HTML structure. It's useful when you want to return multiple elements from a component without needing to wrap them in a div or other container, which could affect your layout or styling.

👉 Q22. What is the role of the useReducer hook in state management?

Ans: The useReducer hook in React is used for managing more complex state logic in your components. It's like a more powerful version of useState. Instead of directly updating the state based on its current value, you can specify how the state should change based on an action. This is particularly helpful when you have state that depends on previous state or when you have multiple values that need to be updated together.


👉 Q23. Describe the purpose of the refs in React and How can we create refs?


Ans: Refs in React provide a way to access and interact with DOM elements or React components directly. They're useful for tasks like focusing on input fields, triggering animations. we can create refs using useRef hook.


👉 Q24. How does React handle events? differences between synthetic events and native events.

Ans: React handles events in a cross-browser compatible way using synthetic events. These synthetic events wrap around native browser events, providing consistent behavior across different browsers. The main difference between synthetic events and native events is that synthetic events are normalized by React to ensure consistency across browsers and provide additional features, while native events are the events directly provided by the browser.


👉 Q25. What is the purpose of the dangerouslySetInnerHTML attribute in React?


 Ans: The `dangerouslySetInnerHTML` attribute in React js is used to set HTML directly from React components. It's typically used when you need to render HTML content received from an external source, like a server or an API. However, it's called "dangerous" because it can expose your application to cross-site scripting (XSS) attacks if not used carefully, since it bypasses React's built-in XSS protections. So, it should only be used when you trust the source of the HTML content.


 Q26. What is the purpose of the `React.StrictMode` component?


 Ans: `React.StrictMode` is like a helper tool in React that helps you catch common mistakes and bugs early during development. It's not for production use but helps you write better React js code by highlighting potential problems.


 Q27. How does React handle forms and form validation?


 Ans: React helps you manage forms by storing form data in its state. It's recommended to use controlled components to handle forms. For validation, you can use built-in form features like `required` attribute, or you can create custom validation functions that check input values against certain criteria, like whether they're empty or meet a specific format.


 Q28. what is core components and custom components in react


 Ans: Core components in React are built-in components provided by React itself, like `div`, `span`, `input`, etc. Custom components are ones you create yourself for specific parts of your app. You can reuse them throughout your project to keep your code organized and maintainable.


 Q29. What is the role of `shouldComponentUpdate` and `PureComponent` in React optimization?

 Ans: `shouldComponentUpdate` is a method you can define in your React components to control when the component should re-render. `PureComponent` is a special type of component that automatically performs a shallow comparison of props and state, and if they haven't changed, it prevents unnecessary re-renders. These are used to optimize performance by avoiding unnecessary rendering cycles.

 Q30. How does React handle error boundaries?

 Ans: Error boundaries in React are components that catch JavaScript errors anywhere in their child component tree, log those errors, and display a fallback UI instead of crashing the whole app. This helps in making apps more robust by preventing unhandled errors from breaking the entire user interface.

 Q31. Explain the significance of the `useEffect` dependencies array and What can go wrong if we forget something in that list?

 Ans: The `useEffect` dependencies array tells React when to run the effect. If you put variables in this array, the effect will run whenever any of those variables change. Forgetting to include variables in this array can lead to bugs where the effect doesn't update when it should or updates unnecessarily.

👉 Q32. Explain the concept of suspense in React and its use cases.

📖 Ans: Suspense in React helps manage asynchronous operations, like fetching data or lazy-loading components. It allows you to show fallback content while waiting for the main content to load. It's useful for improving user experience by avoiding loading spinners and keeping users engaged.

👉 Q33. Explain the concept of code splitting in React and its advantages.

📖 Ans: . Code splitting means breaking your code into smaller chunks and loading them only when needed. In React, it's often used with dynamic imports to load parts of your app asynchronously. This speeds up initial load times and reduces the amount of code users need to download, improving performance.

👉 Q34. Describe the significance of the forwardRef function in React.

📖 Ans: forwardRef allows you to pass a ref from a parent component to a child component. It's useful when you need to access a child component's DOM node or React element directly from the parent. This can be important for integrating with third-party libraries or managing focus and scrolling behavior.

👉 Q35. Explain the purpose of the displayName property in React components.

📖 Ans: The displayName property is used for debugging and improving code readability. It gives a name to a component, making it easier to identify in React DevTools or error messages. It's not necessary for a component to function, but it helps developers understand the component's purpose and structure.

👉 Q36. How does React handle security concerns, such as preventing Cross-Site Scripting (XSS) attacks?

📖 Ans: React prevents security issues like Cross-Site Scripting (XSS) attacks by using a feature called JSX, which automatically escapes any potentially harmful content. It also provides features like prop validation and the use of libraries like DOMPurify to further secure applications.

👉 Q38. Explain the concept of React portals and when you might use them.

📖 Ans: React portals are a way to render child components outside of their parent component's DOM hierarchy. They're useful when you need to render a component at a different place in the DOM hierarchy or when dealing with modal dialogs, tooltips, or popovers.

👉 Q37. Discuss the differences between shallow rendering and full rendering in testing React components.

📖 Ans: Shallow rendering in testing React components only renders the component itself, without rendering its children. Full rendering, on the other hand, renders the component along with all of its children components. Shallow rendering is faster but may not catch issues deep within the component tree, while full rendering ensures a more comprehensive test but can be slower.

👉 Q39. why is State immutable in redux.

📖 Ans: In Redux, state is immutable to ensure that changes are predictable and traceable. It helps in managing application state more effectively by enforcing a single source of truth and making state changes easier to understand and debug.

👉 Q40. why Reducer is pure function in redux.

📖 Ans: Reducers in Redux are pure functions because they should not have side effects and should always return the same output for a given input. This predictability makes it easier to test and reason about the behavior of reducers. They take the previous state and an action, and return a new state without modifying the previous state directly.

👉 Q41. What are stateful and stateless components?

📖 Ans: Stateful components are ones that can hold and manage their own data, known as state. They have the ability to change and update data over time. Stateless components, on the other hand, don't manage their own data; they receive data from their parent components via props and render UI based on that data. They are purely presentational and don't have their own internal state. Class component also called stateful and Functional component called stateless because you have to use hooks in order to make it stateful.

👉 Q42. How do you update state in React?

📖 Ans: In React, state can be updated using the `setState()` method provided by the React library. This method takes an object as an argument, where keys represent the state properties to update and their corresponding values represent the new values of those properties. When `setState()` is called, React re-renders the component with the updated state, reflecting the changes in the user interface. also in functional component we can use setter function of `useState` hook.

👉 Q43. What is Reconciliation in react?

📖 Ans: Reconciliation is the process by which React updates the UI to match the most recent changes in the component's state or props. When state or props change, React compares the new tree of React elements with the previous one and determines the minimal set of changes needed to update from virtual DOM to the actual DOM. This process ensures that the UI remains in sync with the application's state efficiently, optimizing performance by only re-rendering what is necessary.

👉 Q44. What are the diffing algorithms in react ?

📖 Ans: React uses diffing algorithms to efficiently update the user interface when something changes. There are mainly two types :

Tree Diffing: Compares old and new trees to update specific parts of the web page, avoiding full-page re-renders.

Component Diffing: Assigns unique IDs to components and updates only those affected by state or props changes, optimizing performance.

👉 Q45. What is Recoil ?

■ Ans: Recoil is a library made by Facebook for managing the state of a React application. It simplifies the process of sharing and updating data across different parts of the app. In Recoil, you create units of state called "atoms" that hold pieces of data. Components can read from and write to these atoms to access and modify the application's state. Recoil provides an easy way to manage complex state logic and helps in keeping the code organized and efficient.

👉 Q46. Explain the purpose of lifecycle methods in React.

■ Ans: Lifecycle methods in React help you control what happens at different stages of a component's life, like when it's created, updated, or removed from the DOM. They allow you to perform actions like fetching data, updating the UI, or cleaning up resources.

👉 Q47. How does React handle data binding?

■ Ans: React uses a concept called "props" (short for properties) to pass data from one component to another. When a component's props change, React automatically re-renders the component with the new data.

👉 Q48. Explain the purpose of useMemo() and useCallback() hooks.

■ Ans: useMemo() is a hook used for memoization, which means it helps in optimizing performance by caching the result of expensive calculations. useCallback() is similar but specifically for memoizing functions, preventing unnecessary re-creation of functions on every render.

👉 Q49. What are the differences between useMemo() and useCallback()?

■ Ans: The main difference is in what they memoize. useMemo() memoizes the result of a function call, while useCallback() memoizes the function itself. So, useMemo() is for optimizing expensive calculations, and useCallback() is for optimizing function creation.

👉 Q50. What are the different ways to handle events in React?

■ Ans: There are a few ways to handle events in React:

Inline event handlers: Directly specify event handlers in JSX.

Using addEventListener(): Similar to traditional DOM manipulation, but less common in React.

Using SyntheticEvent: React provides a synthetic event system that normalizes events across different browsers.

Using event delegation: Handling events at a higher level in the component tree and letting them propagate down to child components.

👉 Q51. How would you optimize performance in a React application?

■ Ans: You can optimize performance in a React app by :

- Using React.memo() to memoize functional components.
- Implementing code splitting to load only necessary code.
- Using PureComponent or shouldComponentUpdate() to prevent unnecessary re-renders.
- Utilizing React's built-in performance tools like Profiler to identify bottlenecks.

👉 Q52. What is the purpose of ReactDOM.render()?



■ Ans: ReactDOM.render() is used to render React elements into the DOM (Document Object Model). It takes two arguments: the React element to render and the DOM element where the rendered content will be inserted.

👉 Q53. What is Babel and what is its role in React development?

■ Ans: Babel is a JavaScript compiler. It transforms modern JavaScript code into backward-compatible versions that can run in older browsers. In React development, Babel is often used to compile JSX (JavaScript XML) syntax into plain JavaScript code that browsers can understand.

👉 Q54. Explain the significance of propTypes in React.

■ Ans: propTypes are a way to define the type of data expected by a React component's props. They help catch bugs early by enforcing type checks on props passed to components. This makes code more reliable and easier to maintain, especially in larger projects.

👉 Q55. What are the benefits of using React with TypeScript?

■ Ans: Using React with TypeScript offers several benefits:

- Type safety: TypeScript provides static type checking, catching errors during development.
- Improved developer experience: IDEs can provide better auto-completion and documentation for TypeScript code.
- Easier refactoring: TypeScript helps maintain code integrity when making changes, reducing the risk of introducing bugs.
- Enhanced collaboration: With clearly defined types, collaborating on code becomes easier as it's clearer what each piece of code expects and returns.

👉 Q56. How does React differ from other JavaScript frameworks/libraries?

■ Ans: React focuses on building user interfaces by breaking them into reusable components. It uses a virtual DOM for efficient updates. Other frameworks like Angular and Vue have different approaches and may include more features out of the box.

👉 Q57. What is the difference between React and React Native?

■ Ans: React is for building web interfaces, while React Native is for building mobile applications. React uses web technologies (HTML, CSS, JavaScript), while React Native uses native components for iOS and Android.

👉 Q58. How do you handle SEO optimization in a React application?

■ Ans: To handle SEO in React, you can use server-side rendering (SSR) or prerendering techniques. This ensures that search engines can crawl and index your content properly. Additionally, you should use semantic HTML, proper meta tags, and structured data.

👉 Q59. Can you explain what is internationalization in a React application?

■ Ans: Internationalization in a React application refers to the process of designing and developing the application to support multiple languages and locales. This means making the



application adaptable to different languages, date formats, currencies, and other regional conventions. Internationalization often abbreviated as i18n.

👉 Q60. What are the best practices for optimizing performance in a React application?

📌 Ans: Best practices for optimizing performance in React include:

- Implementing code splitting to load only necessary code.
- Using PureComponent or memoization for components to prevent unnecessary renders.
- Avoiding unnecessary re-renders by optimizing state and props usage.
- Properly using keys in lists to help React identify which items have changed.
- Profiling the application using tools like React DevTools or Chrome DevTools to identify performance bottlenecks and optimize accordingly.

👉 Q61. What are the key features of React.js?

📌 Ans:

- Component-Based: React.js allows you to break down your UI into reusable components, making development easier and more organized.
- Virtual DOM: React uses a virtual representation of the DOM, which helps in improving performance by minimizing actual DOM manipulations.
- JSX: JSX is a syntax extension that allows you to write HTML-like code within JavaScript, making it easier to visualize and construct UI components.
- Unidirectional Data Flow: React follows a one-way data binding approach, making it easier to understand how data changes affect the UI.

👉 Q62. What are the benefits of using React.js for web development?

📌 Ans:

- Increased Productivity: React's component-based architecture and reusability features enable faster development.
- Improved Performance: React's virtual DOM ensures efficient rendering, leading to better performance.
- Strong Community Support: React has a large and active community, providing extensive resources and libraries for development.

👉 Q63. Can you explain how you would handle internationalization in a React application?

📌 Ans: To make your React app work in different languages, you can use special libraries (like - react-intl) that help with translations and formatting. These libraries let you keep language-specific content separate, so it's easier to manage. Your app can also be designed to automatically load the right language based on what the user prefers or their browser settings.

👉 Q64. How would you handle error tracking and monitoring in a React application?

📌 Ans: To keep track of errors in your React app, you can use tools like Sentry or Bugsnag. These tools help you identify when something goes wrong and give you details about the problem. React also has a feature called error boundaries that lets you catch errors and show a friendly message to users instead of crashing the whole app. Additionally, you can set up

logging to record errors and what's happening in your app in real-time, which helps you understand and fix issues faster.

👉 Q65. Explain the Flux architecture and its relationship with React.

■ Ans: Flux is a way of organizing how data flows through a web app. It's especially useful when you have a lot of data that changes frequently. While React doesn't require you to use Flux, they work well together. Flux helps you manage your app's data in a structured way, and React's component-based structure fits nicely with Flux's ideas. Libraries like Redux, which are based on Flux, are commonly used with React to keep things organized and predictable, especially in larger apps.

👉 Q66. Describe the process of server-side rendering (SSR) in React.

■ Ans: Server-side rendering (SSR) in React means building your React app on the server and sending a fully rendered page to the browser. It speeds up initial loading and helps search engines understand your site better for SEO.

👉 Q67. What is the difference between `React.createElement` and JSX?

■ Ans: `React.createElement` is a way to create React elements with pure JavaScript code. It is a function used to create React elements programmatically, while JSX is a syntax extension in React that allows you to write HTML-like code within JavaScript. JSX is more readable and concise compared to `React.createElement`.

👉 Q68. How does React handle styling? Discuss different approaches.

■ Ans: React offers various methods for styling, including inline styles, traditional CSS stylesheets, CSS modules, and libraries like styled-components. Each approach has its benefits and is suitable for different project needs.

👉 Q69. How can you optimize images for better performance in a React application?

■ Ans: To optimize images in a React app, you can resize images to the appropriate dimensions, use modern formats like WebP, lazy load images to improve page load times, and compress images to reduce file sizes while maintaining quality.

👉 Q70. How does React support accessibility (a11y) in web applications?

■ Ans: React supports accessibility by providing attributes like `aria-*` for defining roles, states, and properties that help screen readers and assistive technologies understand the UI. It encourages semantic HTML, offers tools like React ARIA, and emphasizes building accessible components for inclusive web experiences.

👉 Q71. How can you implement client-side routing in a React application without React Router?

■ Ans: You can create client-side routing in a React app without using React Router by managing routes manually using the history API or by conditional rendering of components based on URL changes.

👉 Q72. Describe the `useLayoutEffect` hook in React and how it differs from `useEffect`.

📖 Ans: `useLayoutEffect` is a hook in React that lets you perform actions after all DOM mutations. It's similar to `useEffect`, but it runs synchronously after all DOM changes, before the browser has a chance to paint. This can be useful for operations like measuring elements or performing animations.

👉 Q73. Explain the concept of serverless architecture and its relation to React applications.

📖 Ans: Serverless architecture is a way of building and deploying applications without managing servers. In the context of React js apps, serverless backends like AWS Lambda or Azure Functions can be used to handle backend logic, while React handles the front end. This allows for scalability and cost-effectiveness, as you only pay for what you use.

👉 Q74. How does React handle data fetching from APIs or external sources?

📖 Ans: React can handle data fetching from APIs or external sources using various methods like the `fetch` API or libraries like `Axios`. You can fetch data in React components using lifecycle methods, hooks like `useEffect`, or by using asynchronous functions directly.

👉 Q75. What are React Hooks? Explain the motivation behind introducing them.

📖 Ans: React Hooks are functions that let you use state and other React features without writing a class. They were introduced to simplify and streamline React code by allowing you to use state and side effects in functional components. The motivation behind introducing hooks was to make it easier to reuse logic between components and to encourage the use of functional components over class components.

👉 Q76. What are the differences between React and Angular in terms of architecture and features?

📖 Ans: React uses a component-based architecture with JSX for templating, while Angular is a full-fledged framework with built-in features like two-way data binding and routing. React offers more flexibility and a simpler learning curve, while Angular provides more opinionated solutions out of the box.

👉 Q77. What are the differences between React and Angular in terms of performance?

📖 Ans: React's virtual DOM and one-way data flow generally lead to better performance by updating only changed parts. Angular's two-way data binding can sometimes cause performance issues, but Ahead-of-Time (AOT) compilation can help mitigate them. React's lightweight nature and efficient updates contribute to its performance advantages.

👉 Q78. What are the differences between React and Vue.js?

📖 Ans: React and Vue.js both use virtual DOMs, but React promotes a more JavaScript-centric approach with JSX, while Vue.js offers simplicity and reactivity out of the box. React is more of a library, allowing developers to choose additional tools, while Vue.js feels more like a framework with more built-in solutions.

👉 Q79. What are the benefits of using React hooks over class components?

■ Ans: React Hooks allow functional components to manage state and lifecycle features previously exclusive to class components, promoting cleaner and more reusable code. Hooks avoid the complexities of class components and encourage a functional programming style, leading to simpler and more concise code with better organization.

👉 Q80. Explain the role of the render method in React class components.

■ Ans: In React class components, the render method defines how the UI should look based on the component's current state and props. It returns a React element representing the component's output. Whenever state or props change, React calls the render method to update the UI accordingly, ensuring it reflects the latest data.

👉 Q81. How does React handle browser events differently from traditional event handling?

■ Ans: React uses a system called synthetic events that wraps around the native browser events. This allows React to handle events consistently across different browsers and provides additional features like event pooling for better performance.

👉 Q82. Describe the significance of StrictMode in React and how it helps in debugging.

■ Ans: StrictMode is a tool in React that helps in identifying and fixing common issues in your code. It highlights potential problems and deprecated features, making debugging easier. It also enforces strict rendering and warns about unsafe lifecycle methods.

👉 Q83. Explain how you can integrate React with other JavaScript libraries or frameworks.

■ Ans: React provides ways to integrate with other libraries or frameworks like jQuery or Angular. You can use React's lifecycle methods to interact with these libraries, or you can use wrapper components to encapsulate external functionality. Additionally, React also offers APIs like ReactDOM.render() which allow you to render React components within non-React applications.

👉 Q84. What are React hooks rules, and why is it essential to follow them?

■ Ans: React hooks have rules to ensure they're used correctly. One important rule is to always call hooks at the top level of your React function components. This ensures hooks are called in the same order on every render, preventing bugs. Following these rules helps maintain the stability and predictability of your React components.

👉 Q85. Explain the concept of Redux middleware and provide examples of when you might use it.

■ Ans: Redux middleware is like a bridge between your Redux store and your application. It intercepts actions before they reach the reducers, allowing you to perform additional tasks such as logging, making asynchronous requests, or modifying actions.

👉 Q86. How does React handle errors and exceptions in components?

✅ Ans: React has a built-in way to catch errors that happen during rendering, lifecycle methods, and in constructors of whole component trees. It uses a special method called

componentDidCatch(). This helps to prevent the whole app from crashing if one component has an error.

👉 Q87. What are the benefits of using React Testing Library for testing React components?

✅ Ans: React Testing Library helps you test your React components by simulating how users interact with your app. It focuses on testing your components as if they were being used by real people, which helps to catch bugs early and ensure your app works well for your users.

👉 Q88. Describe the concept of "component composition" in React and its advantages.

✅ Ans: Component composition in React means building your app by combining small, reusable components to create more complex ones. This approach makes your code easier to understand, maintain, and reuse. It also promotes a modular and scalable architecture for your app.

👉 Q89. How does React handle browser compatibility issues, especially with older browsers?

✅ Ans: React provides tools like Babel and Polyfills to handle browser compatibility issues. Babel converts modern JavaScript code into a version that older browsers can understand. Polyfills fill in the gaps for missing features in older browsers, ensuring your React app works smoothly across different browsers.

👉 Q90. What are the advantages of using functional programming concepts in React development?

✅ Ans: Functional programming concepts like immutability and pure functions make React development easier and more predictable. They help in writing cleaner, more maintainable code by reducing side effects and making it easier to reason about your code. Additionally, functional programming concepts align well with React's component-based architecture.

👉 Q91. How does React handle memory management and prevent memory leaks?

✅ Ans: React takes care of cleaning up unused components and their associated memory automatically. It uses a mechanism called "garbage collection" to identify and remove components that are no longer needed, preventing memory leaks.

👉 Q92. Explain the benefits of using CSS-in-JS libraries with React applications.

✅ Ans: CSS-in-JS libraries help you write and manage CSS styles directly within your JavaScript code. This makes styling easier and more modular because styles are scoped to specific components. It also improves performance by reducing unnecessary style calculations.

👉 Q93. What are the differences between React and Angular in terms of syntax and templating?

✅ Ans: React uses JSX for templating, which is a syntax extension for JavaScript. It's more flexible and allows you to write HTML-like code directly in your JavaScript files. Angular, on the other hand, uses HTML templates with additional syntax for data binding and directives.

👉 Q94. Explain the concept of "render props" in React and how it's used.

✓ Ans: Render props is a pattern in React where a component's logic is encapsulated within a function that gets passed as a prop. This allows the parent component to control what is rendered by the child component, providing flexibility and reusability in component composition.

👉 Q95. What are the benefits of using React Router for client-side routing?

✓ Ans: React Router simplifies client-side routing in React applications. It allows you to define routes and map them to specific components, making it easy to manage navigation and state changes in a single-page application. It also provides features like nested routes and URL parameters for more advanced routing needs.

👉 Q96. How does React support server-side rendering (SSR) with Node.js?

✓ Ans: React can be used on the server side with Node.js to render web pages. It means React can create the initial HTML for a webpage on the server before sending it to the browser. This helps in faster page loading and better search engine optimization (SEO).

👉 Q97. Describe the concept of "composition vs. inheritance" in React component design.

✓ Ans: Composition means building components by combining smaller components together, like putting LEGO blocks together to create something bigger. Inheritance, on the other hand, involves creating new components by extending the features of existing ones, which is like inheriting traits from your parents. In React, it's generally better to use composition over inheritance because it's more flexible and leads to less complex code.

👉 Q98. How does React support code reuse and maintainability in large-scale applications?

✓ Ans: React encourages code reuse through component-based architecture, where UIs are built as a combination of reusable components. This makes it easier to maintain and update code because changes made to one component won't affect others.

👉 Q99. How does React handle memory management and garbage collection?

✓ Ans: React automatically manages memory by keeping track of components and their state. When a component is no longer needed, React will remove it from memory through a process called garbage collection. This helps in efficient memory usage and prevents memory leaks.

👉 Q100. What are the differences between React's `componentDidMount` and `componentDidUpdate` lifecycle methods?

✓ Ans: `componentDidMount` is called after a component is rendered for the first time, while `componentDidUpdate` is called after a component is updated (i.e., when props or state change). So, `componentDidMount` is used for tasks that should happen once after the initial render, like fetching data from a server, while `componentDidUpdate` is used for tasks that should happen after each update, like updating the DOM in response to prop or state changes.

👉 Q101. What is the difference between class components and functional components?

Ans: class components use ES6 classes and have additional features like state and lifecycle methods, while functional components are simpler and are often used with hooks. class

components also know as stateful components and functional components know as stateless components