|  |
| --- |
| REACT BASIC****--********1.What is React?**** React is a JavaScript library for building user interfaces, maintained by Facebook. It facilitates the creation of interactive and dynamic web applications by using a component-based architecture. React efficiently updates and renders the user interface through a virtual DOM, enhancing performance and providing a seamless development experience. ****2.What are the key features of React?**** Some key features of React include:  1.Virtual DOM for efficient rendering and performance optimization.  2.Component-based architecture for reusability and modular development.  3.One-way data flow for predictable and maintainable code.  4.Unidirectional data binding with state and props management.  5.React Native for building mobile applications using React."  **3.What is JSX? Why can't browsers read JSX?**  JSX stands for JavaScript XML. It is a React extension which allows writing JavaScript code that looks similar to HTML. It makes HTML file easy to understand. The JSX file makes the React application robust and boosts its performance. JSX provides you to write XML-like syntax in the same file where you write JavaScript code, and then preprocessor (i.e., transpilers like Babel) transform these expressions into actual JavaScript code. Just like XML/HTML, JSX tags have a tag name, attributes, and children.  Browsers cannot read JSX directly because they can only understand JavaScript objects, and JSX is not a regular JavaScript object. Thus, we need to transform the JSX file into a JavaScript object using transpilers like Babel and then pass it to the browser.  **4.What is conditional rendering in React?**  Conditional rendering in React involves rendering different components or content based on certain conditions. You can use if statements or ternary operators within JSX to conditionally render elements. ****5.What is a React Router, and why is it used?**** React Router is a library for handling routing in React applications. It allows developers to create declarative routing configurations, enabling navigation between different views or pages within a single-page application (SPA). React Router provides components like BrowserRouter, Route, and Link to define routes, match URLs, and navigate between views.  **6. What is the purpose of the callback function as an argument of setState()?**  The callback function is invoked when setState is finished and the component gets rendered. Since setState() is asynchronous the callback function is used for any post action.  Note: It is recommended to use the lifecycle method rather than this callback function.  Javascript  setState({ name: "John" }, () =>  console.log("The name has updated and component re-rendered")  );  **7.What is the difference between Shadow DOM and Virtual DOM?**  The goal of React Fiber is to increase its suitability for areas like animation, layout, and gestures. Its headline feature is incremental rendering: the ability to split rendering work into chunks and spread it out over multiple frames.  from documentation  Its main goals are:   * Ability to split interruptible work in chunks. * Ability to prioritize, rebase and reuse work in progress. * Ability to yield back and forth between parents and children to support layout in React. * Ability to return multiple elements from render(). * Better support for error boundaries.   **8.Why fragments are better than container divs?**  Below are the list of reasons to prefer fragments over container DOM elements,   * Fragments are a bit faster and use less memory by not creating an extra DOM node. This only has a real benefit on very large and deep trees. * Some CSS mechanisms like Flexbox and CSS Grid have special parent-child relationships, and adding divs in the middle makes it hard to keep the desired layout. * The DOM Inspector is less cluttered.   **9.How events are different in React?**  Handling events in React elements has some syntactic differences:  -React event handlers are named using camelCase, rather than lowercase.  -With JSX you pass a function as the event handler, rather than a string.    **REACT STATE & HOOKS--**  **10.What are React hooks?**  React hooks are functions that allow developers to use state and other React features in functional components. Hooks were introduced in React 16.8 as a way to write reusable logic and manage stateful behavior in functional components. Some commonly used hooks are useState, useEffect, and useContext.  **11.What are "hooks" in React, and how do they change the way you manage state and side effects?**  Hooks in React are functions that allow you to "hook into" React state and lifecycle features from functional components. They were introduced in React 16.8 to address some of the limitations and complexities associated with managing state and side effects in class components. Hooks make it easier to reuse logic across components and promote the use of functional components over class components. ****12.What is the purpose of state in React?**** State is used to manage and store data within a component. It represents the current state of the component and can be changed over time. When the state of a component is updated, React automatically re-renders the component and its child components to reflect the changes in the UI.  **13. Why should we not update the state directly?**  If you try to update the state directly then it won't re-render the component.  //Wrong  Javascript  this.state.message = "Hello world";  Instead, use the setState() method. It schedules an update to a component's state object. When state changes, the component responds by re-rendering.  //Correct  Javascript  this.setState({ message: "Hello World" });  Note: You can directly assign to the state object either in the constructor or using the latest javascript's class field declaration syntax.  **14.What is the purpose of the useState hook?**  The useState hook is used to add state to functional components. It takes an initial state value as an argument and returns an array with two elements: the current state value and a function to update the state. By using the useState hook, functional components can maintain their own state without using class components.  **15.What is the purpose of the useEffect hook?**  The useEffect hook in React is used to perform side effects in functional components. It allows you to execute code after the component has rendered or when specific dependencies have changed. Side effects can include API calls, subscriptions, or manipulating the DOM. The useEffect hook takes a callback function as its first argument and an optional array of dependencies as its second argument. 16.When would You use usecontext hook? You would use the useContext hook in React when you need to access data or functions from a context within a functional component. It simplifies consuming context values and is useful for managing global state, such as themes, user authentication, and application settings. ****17.What is the significance of the dependency array in the useEffect hook?**** The dependency array in the useEffect hook specifies the values that the effect depends on. When any of the values in the dependency array change, the effect is re-executed. If the dependency array is empty, the effect runs only once after the initial render. By specifying dependencies, you can control when the effect should be triggered or if it should be skipped. ****18.What is the difference between props and state?**** "Props and state are both used to pass data to components, but there are some key differences:  Props are passed from parent components to child components and are read-only within the child component. They are used to configure and customize a component.  State is managed within the component itself and can be changed using the setState() method. State represents the internal data of a component and can trigger re-rendering when updated."   **19.What is the purpose of the useReducer hook in React?**  The useReducer hook is used to manage complex state logic in a more centralized and predictable way. It is an alternative to using useState when the state transitions involve multiple values or complex logic. useReducer takes a reducer function and an initial state and returns the current state and a dispatch function to trigger state transitions. ****20.What is the purpose of the useContext hook?**** The useContext hook is used to consume a React context in a functional component. It allows you to access the value provided by a context provider higher up in the component tree without the need for prop drilling. The useContext hook takes the context object as its argument and returns the current context value. ****21.What is the purpose of the useRef hook in React and how is it used?**** The useRef hook is used to create a mutable reference that persists across component renders. It returns a mutable ref object with a .current property that can hold a value. useRef is commonly used for accessing DOM elements, storing mutable values, and preserving values between renders without triggering a re-render. 22.Do two components using the same Hook share state? No, two components using the same hook do not share state. Each component using a hook, like **useState**, **useEffect**, or **useRef**, maintains its own separate state and data. Hooks are used per component and don't share state or data between different instances of the component. This is a key principle of React's functional component model: each component manages its own state independently. 23.Explain the difference between usestate() and useref()? **useState**() is for managing and updating component state, triggering re-renders. **useRef**() is for accessing DOM elements and storing mutable values without causing re-renders. 24.What are the production use cases of useref()? Production use cases for **useRef**() in React include:   1. Managing DOM elements. 2. Controlling form input focus. 3. Integrating with third-party libraries. 4. Caching values across renders. 5. Managing animations and timers. 6. Improving performance in specific scenarios.   **25.What is the purpose of the setState method in React?**  The setState method in React is used to update a component's state. When the state changes, React automatically re-renders the component to reflect the updated state.  **26.How do you update the state in React?**  "You can update the state in React using the setState() method. For example:  Javascript  this.setState({ count: this.state.count + 1 });  **27.How do you initialize state in a React component?**  you can initialize state in a React component by defining a state property within the component's constructor. For example:  Javascript  class MyComponent extends React.Component {  constructor() {  super();  this.state = { count: 0 };  }  // ...  }"  **28.How can you update the State of a component?**  We can update the State of a component using this.setState() method. This method does not always replace the State immediately. Instead, it only adds changes to the original State. It is a primary method which is used to update the user interface(UI) in response to event handlers and server responses. ****29.What is the output of below code?**** Javascript  import { useState } from 'react';  export default function Counter() {  const [counter, setCounter] = useState(5);  return (  <>  <span>{counter}</span>  <button onClick={() => {  setCounter(counter + 5);  setCounter(counter + 5);  alert(counter);  setCounter(counter + 5);  setCounter(counter + 5);  }}>Increment</button>  </>  )  }"  **Answer**: 3(Alert with 5, 10)  State values are fixed(i.e, default value 5) in each render and setting the state only changes it for the next render. React will wait untill all the code executed with in an event handler before your state updates follwed by re-rendering the UI. Also, all the 3 setter function calls are replacing the calculated value. Hence, irrespective of how many times you call setCounter(counter + 5) the final value is 10(5+5).  This can be visuallized by substituting with state variable values in the particular render,  Javascript  <button onClick={() => {  setCounter(5 + 5);  setCounter(5 + 5);  alert(5);  setCounter(5 + 5);  setCounter(5 + 5);  }}>Increment</button>" ****ERROR, HANDLING & OPTIMIZATION IN REACT--********30.How do you handle errors in React applications?**** In React, you can handle errors using Error Boundaries. Error Boundaries are components that catch JavaScript errors in their child components' lifecycle methods or during rendering, and display fallback UI instead of crashing the whole application. You can define Error Boundary components by implementing the **componentDidCatch** lifecycle method. By wrapping components with Error Boundaries, you can gracefully handle errors and provide a better user experience. ****31.What are React error boundaries, and how can they be used to handle errors in components?**** React error boundaries are special components that catch JavaScript errors in their child components' lifecycle methods, constructors, and render methods. They help prevent the entire application from crashing due to an error in a single component. Error boundaries allow you to gracefully handle errors by displaying a fallback UI and logging the error information. They are useful in production environments to ensure that errors in components don't break the entire application. ****32.How can you optimize performance in a React application?**** You can optimize performance in a React application by using techniques like **shouldComponentUpdate** (for class components), PureComponent, memo ****33.How do you handle forms in React?**** You can handle forms in React by using controlled components, where the form elements are linked to React state, and their values are controlled by React. This allows you to easily handle form submission and validation. ****34.How do you handle events in React?**** You can handle events in React by passing event handlers as props to components. For example, to handle a button click event:  **35. How to bind methods or event handlers in JSX callbacks?**  There are 3 possible ways to achieve this in class components:  **1**.**Binding in Constructor**: In JavaScript classes, the methods are not bound by default. The same rule applies for React event handlers defined as class methods. Normally we bind them in constructor.  Javascript  class User extends Component {  constructor(props) {  super(props);  this.handleClick = this.handleClick.bind(this);  }  handleClick() {  console.log("SingOut triggered");  }  render() {  return <button onClick={this.handleClick}>SingOut</button>;  }  }  **2**.**Public class fields syntax**: If you don't like to use bind approach then public class fields syntax can be used to correctly bind callbacks. The Create React App eanables this syntax by default.  Javascript  handleClick = () => {  console.log("SingOut triggered", this);  };  <button onClick={this.handleClick}>SingOut</button>  **3**.**Arrow functions in callbacks**: It is possible to use arrow functions directly in the callbacks.  Javascript  handleClick() {  console.log('SingOut triggered');  }  render() {  return <button onClick={() => this.handleClick()}>SignOut</button>;  }  **Note**: If the callback is passed as prop to child components, those components might do an extra re-rendering. In those cases, it is preferred to go with .bind() or public class fields syntax approach considering performance.  **36.When to use useCallback()?**  Use **useCallback** in React to optimize performance by memoizing functions, particularly when functions depend on specific props, are passed as dependencies, or used as event handlers to prevent unnecessary re-creation and re-renders. 37.What are the common use case of usememo()? Common use cases for **useMemo** in React are:   * Memoizing expensive computations. * Preventing unnecessary component renders. * Optimizing performance by avoiding redundant calculations. * Memoizing props to prevent child component re-renders.  38.What are the Differences between React.memo() and usememo()? **React.memo()** memoizes functional components to prevent unnecessary renders.  **useMemo**() memoizes the result of a computation to optimize performance by re-computing only when dependencies change. ****39.Describe the concept of "virtualization" in React and how it can be applied to long lists or tables.**** Virtualization in React optimizes rendering long lists or tables by only rendering the visible items as the user scrolls. It saves memory and boosts performance, especially with large datasets, by reusing DOM elements and dynamically rendering content in the viewport. Libraries like react-window and react-virtualized simplify its implementation. ****40.Explain what "memoization" is and how it can be utilized for performance optimization in React.**** Memoization is a performance optimization technique that caches expensive function results. In React, you can use React.memo() to avoid unnecessary re-renders of components with the same props, and useMemo() to cache computed values based on dependencies, improving rendering performance. It's particularly useful for complex lists or heavy calculations in React components.   ****REACT COMPONENTS--**** **41. When to use a Class Component over a Function Component?**  After the addition of Hooks(i.e. React 16.8 onwards) it is always recommended to use Function components over Class components in React. Because you could use state, lifecycle methods and other features that were only available in the class component present in the function component too.  But even there are two reasons to use Class components over Function components.  If you need a React functionality whose Function component equivalent is not present yet, like Error Boundaries.  In older versions, If the component needs state or lifecycle methods then you need to use class component.  Note: You can also use reusable react error boundary third-party component without writing any class. i.e, No need to use class components for Error boundaries.  **42.What is the difference between a functional component and a class component in React?**  In React, a functional component is a plain JavaScript function that accepts props as an input and returns a React element. Functional components are simpler and easier to test and maintain. On the other hand, a class component is a JavaScript class that extends the React.Component class. Class components have additional features, such as lifecycle methods and local state. ****43.What is a controlled component in React?**** A controlled component in React is a component whose value is controlled by the state. Input elements like text fields and checkboxes are often controlled components in React. ****44.What is an uncontrolled component in React?**** An uncontrolled component in React is a component whose value is not controlled by the state. The value is typically handled directly by the DOM, and React does not manage it.  **45.What are stateless components?**  If the behavior of a component is independent of its state then it can be a stateless component. You can use either a function or a class for creating stateless components. But unless you need to use a lifecycle hook in your components, you should go for function components. There are a lot of benefits if you decide to use function components here; they are easy to write, understand, and test, a little faster, and you can avoid this keyword altogether.  **46.What are the different phases of React component's lifecycle?**  If the behavior of a component is independent of its state then it can be a stateless component. You can use either a function or a class for creating stateless components. But unless you need to use a lifecycle hook in your components, you should go for function components. There are a lot of benefits if you decide to use function components here; they are easy to write, understand, and test, a little faster, and you can avoid this keyword altogether.  React class components have a lifecycle that can be divided into several phases, each with its associated methods. These methods allow you to hook into different points in the component's existence and perform tasks or side effects. Here are the different phases of a React component's lifecycle:-  1.intilization  2.Mounting  3.Updating  4.Unmounting  5.Error Handling.  **47. What would be the common mistake of function being called every time the component renders?**  You need to make sure that the function is not being called while passing the function as a parameter.  Javascript  render() {  // Wrong: handleClick is called instead of passed as a reference!  return <button onClick={this.handleClick()}>{'Click Me'}</button>  }  Instead, pass the function itself without parenthesis:  Javascript  render() {  // Correct: handleClick is passed as a reference!  return <button onClick={this.handleClick}>{'Click Me'}</button>  } ****48.What is the significance of the key prop when rendering a list of components?**** The key prop in React is used to give each element in a list a unique identifier. When rendering a list of components, React uses the key prop to efficiently update and reorder the components. It helps React determine which components need to be updated, added, or removed when the list changes, improving performance and preventing unnecessary re-rendering. ****49.Explain the concept of "compound components" and provide an example of using them in a React application.**** "Compound components" is a design pattern in React that allows you to build a set of components that work together to achieve a specific functionality. These components are meant to be used together as a group, and they share a common parent to manage and coordinate their behavior. Compound components provide a structured way to create complex UI components with a clear and intuitive API.  **50.What is the purpose of the key prop in React lists?**  The key prop in React lists is used to help React identify each item in a list of elements. It ensures efficient updates when items are added, removed, or rearranged in the list.  **51. What is children prop?**  **Children is a prop (this.props.children) that allows you to pass components as data to other components, just like any other prop you use. Component tree put between component's opening and closing tag will be passed to that component as children prop.**  **There are several methods available in the React API to work with this prop. These include React.Children.map, React.Children.forEach, React.Children.count, React.Children.only, React.Children.toArray.** ****52.What is the purpose of the render method in a class component?**** The render method in a class component is responsible for returning the JSX that represents the component's UI. It is a required method and must return a single root element. ****53.How do you make an HTTP request in React?**** You can make an HTTP request in React using technologies like the fetch API or by using third-party libraries like Axios. These requests are typically made in lifecycle methods like componentDidMount or by using hooks like useEffect. ****54.What is a React fragment?**** A React fragment is a lightweight way to group multiple elements in React without adding an extra DOM node. It is useful when you need to return adjacent elements from a component without a wrapping container.   **55.What are React fragments and why are they useful?**  React fragments allow you to group multiple elements without adding an extra DOM element to the rendered output. Fragments are useful when you need to return multiple elements from a component's render method without introducing unnecessary markup. They can be declared using the <React.Fragment> syntax or the shorthand <> syntax. ****56.What is the role of a router in a React application?**** A router in a React application, like React Router, is used for handling client-side routing. It allows you to navigate between different views or components based on the URL, enabling a single-page application (SPA) experience. ****57.How do you handle routing in React using React Router?**** To handle routing in React using React Router, you need to define route components that specify which component to render for a given URL. You also set up a <Router> component that manages the overall routing configuration. ****58.What is the virtual DOM in React?**** The virtual DOM in React is a lightweight in-memory representation of the actual DOM. React uses it to efficiently update and render components by minimizing actual DOM manipulations. ****59.How do you pass props from a parent component to a child component?**** You can pass props to a child component by including them as attributes when rendering the child component. For example:  Javascript  <ChildComponent prop1={value1} prop2={value2} />  **60. What is Lifting State Up in React?**  When several components need to share the same changing data then it is recommended to lift the shared state up to their closest common ancestor. That means if two child components share the same data from its parent, then move the state to parent instead of maintaining local state in both of the child components  **61.What is context?**  Context provides a way to pass data through the component tree without having to pass props down manually at every level.  For example, authenticated users, locale preferences, UI themes need to be accessed in the application by many components.  Javascript  const { Provider, Consumer } = React.createContext(defaultValue); Intermediate****62.What are the biggest limitations of React?**** "In terms of limitations, React has a learning curve, especially for newcomers, and can involve a fair amount of boilerplate code. Configuration setup can be complex, and developers need to be mindful of performance optimization. React lacks built-in global state management, which can lead to choices and complexities. SEO can be affected by client-side rendering. Staying up-to-date with the evolving tooling ecosystem and dealing with library fragmentation can be challenging. Additionally, React doesn't offer built-in solutions for routing and AJAX, and mobile development often requires learning React Native separately." ****63.Explain the concept of conditional rendering in React.**** Conditional rendering in React refers to the ability to conditionally render components or content based on certain conditions or values. This can be done using conditional statements (like if or ternary operators) within the component's render method, or by using logical operators (&& or ||) to conditionally include or exclude elements from the JSX.  **64.What are inline conditional expressions?**  You can use either if statements or ternary expressions which are available from JS to conditionally render expressions. Apart from these approaches, you can also embed any expressions in JSX by wrapping them in curly braces and then followed by JS logical operator &&.  **Example**:-  Javascript  <h1>Hello!</h1>;  {  messages.length > 0 && !isLogin ? (  <h2>You have {messages.length} unread messages.</h2>  ) : (  <h2>You don't have unread messages.</h2>  );  } ****65.What is the purpose of the React Router library, and how does it handle client-side routing in React applications?**** React Router is a popular library used for client-side routing in React applications. It provides a declarative way to define routes and manage navigation between different views or pages within a single-page application. React Router uses the URL history API to keep the UI in sync with the current URL and enables navigation without page refreshes. It allows for dynamic routing, nested routes, and parameterized routes, making it easy to create complex routing configurations in React applications. ****66.Explain the purpose of render() in React.**** It is mandatory for each React component to have a render() function. Render function is used to return the HTML which you want to display in a component. If you need to rendered more than one HTML element, you need to grouped together inside single enclosing tag (parent tag) such as <div>, <form>, <group> etc. This function returns the same result each time it is invoked. ****67.What is arrow function in React? How is it used?**** The Arrow function is the new feature of the ES6 standard. If you need to use arrow functions, it is not necessary to bind any event to 'this.' Here, the scope of 'this' is global and not limited to any calling function. So If you are using Arrow Function, there is no need to bind 'this' inside the constructor. It is also called 'fat arrow '(=>) functions. ****68.Explain the Lists in React.**** Lists are used to display data in an ordered format. In React, Lists can be created in a similar way as we create it in JavaScript. We can traverse the elements of the list using the map() function.  **69.How do you implement Server Side Rendering or SSR?**  React is already equipped to handle rendering on Node servers. A special version of the DOM renderer is available, which follows the same pattern as on the client side.  Javascript  import ReactDOMServer from "react-dom/server";  import App from "./App";  Javascript  ReactDOMServer.renderToString(<App />);  This method will output the regular HTML as a string, which can be then placed inside a page body as part of the server response. On the client side, React detects the pre-rendered content and seamlessly picks up where it left off. |