Top of Form

Question 1: **Incorrect**

**What are the implications of using array indices as keys for list items in React?**

* 

**It simplifies the rendering process.**

**(Incorrect)**

* 

**It may lead to incorrect UI updates and performance issues.**

**(Correct)**

* 

**It makes the code more readable.**

* 

**It ensures that the list is sorted correctly.**

**Explanation**

Using array indices as keys can lead to incorrect UI updates and performance issues. React relies on keys to track elements in lists. Using indices may result in issues when items are added, removed, or re-ordered, causing React to produce unexpected results. Correctly chosen keys are crucial for accurate rendering.

Bottom of Form

Top of Form

Question 2: **Correct**

**Explain the concept of a 'component' in React.**

* 

**A reusable UI building block**

**(Correct)**

* 

**A database schema**

* 

**A server that serves web pages**

* 

**A routing mechanism**

**Explanation**

In React, a 'component' is a reusable UI building block. Components are self-contained, encapsulated units of the user interface that can be composed together to create complex UIs. They promote reusability and maintainability in React applications.

Bottom of Form

Top of Form

Question 3: **Incorrect**

**How can you optimize a React component to prevent unnecessary re-renders due to changes in 'state' or 'props'?**

* 

**Wrap the component in a 'React.memo' higher-order component**

**(Correct)**

* 

**Avoid using 'shouldComponentUpdate' method**

**(Incorrect)**

* 

**Manually skip rendering based on conditions**

* 

**Use 'componentShouldUpdate' lifecycle method**

**Explanation**

To optimize a React component and prevent unnecessary re-renders due to changes in 'state' or 'props,' you can wrap the component in a 'React.memo' higher-order component. 'React.memo' performs a shallow comparison of 'props' and only re-renders the component when there are actual changes. This can significantly improve performance and is a recommended optimization technique in React applications.

Bottom of Form

Top of Form

Question 4: **Correct**

**What issues might arise if you don't use unique keys for elements in a list in React?**

* 

**Enhanced security**

* 

**Reordering and reconciliation problems**

**(Correct)**

* 

**Improved performance and faster rendering**

* 

**No issues at all**

**Explanation**

If you don't use unique keys for elements in a list in React, you may encounter reordering and reconciliation problems. React relies on keys to identify and update list items efficiently. Without unique keys, React might re-render and reorder components incorrectly, leading to unexpected behavior and performance issues.

Bottom of Form

Top of Form

Question 5: **Correct**

**What is the significance of the 'non-' lifecycle method in React?**

* 

**It is executed before the component is created**

* 

**It is used for defining component props**

* 

**It is called after rendering the component**

**(Correct)**

* 

**It handles component unmounting**

**Explanation**

The 'componentDidMount' lifecycle method in React is significant because it is called after the component is rendered in the DOM. This is a good place to perform side effects, such as data fetching, interacting with the DOM, or initializing third-party libraries.

Bottom of Form

Top of Form

Question 6: **Correct**

**To conditionally render elements, you can use JavaScript operators like \_\_\_\_\_\_\_ and \_\_\_\_\_\_\_ inside JSX.**

* 

**while and switch**

* 

**ternary and logical &&**

**(Correct)**

* 

**map and reduce**

* 

**if and for**

**Explanation**

To conditionally render elements, you can use JavaScript operators like ternary (conditional) and logical && (AND) inside JSX. These operators allow you to express conditional rendering logic, showing or hiding elements based on certain conditions, making your React components more dynamic and responsive.

Bottom of Form

Top of Form

Question 7: **Correct**

**What is an 'event' in the context of React?**

* 

**A function that updates the component**

* 

**A user action or system trigger**

**(Correct)**

* 

**A component's internal state variable**

* 

**A data structure containing state changes**

**Explanation**

In React, an 'event' refers to a user action or system trigger, such as a mouse click, keyboard press, or touch input. Events are used to handle user interactions and trigger functions or methods within React components in response to these actions.

Bottom of Form

Top of Form

Question 8: **Incorrect**

**How does React's synthetic event system differ from the native browser event system?**

* 

**They are exactly the same**

* 

**React's synthetic events are not cross-browser compatible**

**(Incorrect)**

* 

**React's synthetic events are asynchronous**

**(Correct)**

* 

**React's events are not used for UI interactions**

**Explanation**

React's synthetic event system differs from the native browser event system because React's synthetic events are asynchronous. This means that React batches and optimizes event handling, improving performance. Native browser events, on the other hand, can be synchronous, potentially leading to performance issues. Understanding this difference is crucial for efficiently handling events in React applications.

Bottom of Form

Top of Form

Question 9: **Incorrect**

**How can you update the state of a React component?**

* 

**State can only be updated by Redux**

* 

**By calling the 'setState' method**

**(Correct)**

* 

**By directly modifying the state object**

**(Incorrect)**

* 

**State is immutable and can't be updated**

**Explanation**

You can update the state of a React component by calling the 'setState' method. It is the recommended way to modify state in React components. When 'setState' is called, React will re-render the component with the updated state, reflecting changes in the UI.

Bottom of Form

Top of Form

Question 10: **Incorrect**

**The method used to pass data from a child component back to its parent is known as \_\_\_\_\_\_\_.**

* 

**Rendering**

**(Incorrect)**

* 

**State management**

* 

**Callback**

**(Correct)**

* 

**Event handling**

**Explanation**

The method used to pass data from a child component back to its parent is known as a "Callback." Callbacks are functions that a parent component can pass as props to child components, allowing child components to communicate with their parent components and trigger specific actions when needed.

Bottom of Form

Top of Form

Question 11: **Correct**

**How can you use the ternary operator for conditional rendering in React?**

* 

**By using it within JSX to conditionally render one of two components**

**(Correct)**

* 

**By rendering both components simultaneously**

* 

**By modifying the state directly**

* 

**Ternary operator cannot be used in React**

**Explanation**

In React, you can use the ternary operator within JSX to conditionally render one of two components based on a condition. This is a common technique for achieving conditional rendering in React, allowing you to create dynamic UIs that adapt to different states.

Bottom of Form

Top of Form

Question 12: **Correct**

**In React, the method used to update the state of a component is called \_\_\_\_\_\_\_.**

* 

**modifyState()**

* 

**renderState()**

* 

**updateState()**

* 

**setState()**

**(Correct)**

**Explanation**

In React, the method used to update the state of a component is called setState(). This method is used to modify the state of a React component and trigger a re-render of the component to reflect the updated state.

Bottom of Form

Top of Form

Question 13: **Correct**

**You are tasked with displaying a component of user comments in a social media application. How would you ensure that each comment is rendered uniquely and efficiently?**

* 

**Render all comments using a map without specifying a key.**

* 

**Create a single component for all comments to improve efficiency.**

* 

**Use a key prop with a unique identifier when rendering each comment.**

**(Correct)**

* 

**Give each comment a random ID and use it for rendering.**

**Explanation**

To render user comments uniquely and efficiently, it's crucial to use a unique identifier as the "key" prop for each comment when rendering them. This helps React efficiently update and manage the list of comments, preventing issues like duplication or reordering, and optimizing performance. Using random IDs or not specifying a key can lead to rendering problems.

Bottom of Form

Top of Form

Question 14: **Correct**

**Explain the purpose and usage of the map() function in rendering lists in React.**

* 

**Map is used for geographic features**

* 

**Map is used for audio playback**

* 

**Map is a CSS styling method**

* 

**The map() function is for data iteration**

**(Correct)**

**Explanation**

The map() function in React is used for data iteration and rendering lists. It's commonly used to transform an array of data into an array of React elements (components). You can map over the data, create individual components, and render them in a list. This function is essential for rendering dynamic lists and helps improve code readability and maintainability in React applications.

Bottom of Form

Top of Form

Question 15: **Incorrect**

**React wraps all events in a wrapper called \_\_\_\_\_\_\_ to ensure they have consistent properties across different browsers.**

* 

**SyntheticEvent**

**(Correct)**

* 

**EventEmitter**

* 

**EventWrapper**

**(Incorrect)**

* 

**BrowserEvent**

**Explanation**

React wraps all events in a wrapper called SyntheticEvent to ensure they have consistent properties across different browsers. This abstraction allows developers to work with events consistently, regardless of browser differences, providing a cross-browser event-handling solution.

Bottom of Form

Top of Form

Question 16: **Correct**

**A colleague is facing an issue where a component is re-rendering too often, leading to performance issues. How would you troubleshoot and resolve this issue?**

* 

**Decrease the component's render method complexity**

* 

**Implement shouldComponentUpdate lifecycle method**

**(Correct)**

* 

**Use the 'PureComponent' class for the component**

* 

**Set the component to never re-render**

**Explanation**

To troubleshoot and resolve excessive re-rendering, you should implement the 'shouldComponentUpdate' lifecycle method. This method allows you to control when a component should re-render based on certain conditions, improving performance. Setting the component to never re-render isn't a practical solution. Using 'PureComponent' can help in some cases but may not always solve the issue. Reducing render method complexity is a good practice but may not address the core problem of unnecessary re-renders.

Bottom of Form

Top of Form

Question 17: **Correct**

**The lifecycle method which is invoked right before calling the render method on an initial mount and on every update is called \_\_\_\_\_\_\_.**

* 

**componentDidLoad**

* 

**componentRendered**

* 

**componentDidMount**

**(Correct)**

* 

**componentWillRender**

**Explanation**

The lifecycle method invoked right before calling the render method on the initial mount and on every update is componentDidMount. This method is commonly used for performing setup tasks or fetching data from external sources. It ensures that the component is ready to be displayed on the screen.

Bottom of Form

Top of Form

Question 18: **Correct**

**How can you pass data from a parent component to a child component in React?**

* 

**Through props and callback functions**

**(Correct)**

* 

**By directly modifying child component props**

* 

**Using state management libraries like Redux**

* 

**By modifying the child component's state directly**

**Explanation**

In React, you can pass data from a parent component to a child component through props and callback functions. The parent component can pass data as props, and the child component can use these props to receive and render the data, allowing for a unidirectional data flow.

Bottom of Form

Top of Form

Question 19: **Correct**

**You are tasked with optimizing a large React application. How would you ensure that a component does not re-render unnecessarily when its props remain the same?**

* 

**Use the shouldComponentUpdate method**

* 

**Make all components stateful to prevent re-renders**

* 

**Implement memoization and use React.memo for functional components**

**(Correct)**

* 

**Manually trigger a re-render each time props change**

**Explanation**

To prevent unnecessary re-renders when props remain the same, you can implement memoization using React.memo for functional components. This technique memoizes the component and only re-renders it when its props change. It is a performance optimization to avoid rendering components unnecessarily and is particularly useful in large applications with many components.

Bottom of Form

Top of Form

Question 20: **Correct**

**What is the purpose of 'state' in a React component?**

* 

**To store static data**

* 

**To handle routing and navigation**

* 

**To define styling for the component**

* 

**To manage and store dynamic data**

**(Correct)**

**Explanation**

In React, the 'state' of a component is used to manage and store dynamic data. It allows components to maintain and update their data over time, making them interactive and responsive to user interactions. 'State' is crucial for building dynamic UIs.

Bottom of Form

Top of Form

Question 21: **Correct**

**What is the primary purpose of React?**

* 

**To write backend server code**

* 

**To perform server-side operations**

* 

**To create interactive user interfaces**

**(Correct)**

* 

**To manage databases**

**Explanation**

React's primary purpose is to create interactive user interfaces. It allows developers to build UI components that efficiently update in response to data changes, making it ideal for building dynamic and responsive web applications.

Bottom of Form

Top of Form

Question 22: **Incorrect**

**When a list's order changes, React uses the key prop to efficiently \_\_\_\_\_\_\_ the list elements in the DOM.**

* 

**Render**

**(Incorrect)**

* 

**Reorder**

**(Correct)**

* 

**Remove**

* 

**Update**

**Explanation**

When a list's order changes, React uses the key prop to efficiently reorder the list elements in the DOM. Keys provide React with a way to identify and track each element in a list, ensuring that when the order changes, React can update the DOM efficiently without unnecessary re-renders, leading to improved performance.

Bottom of Form

Top of Form

Question 23: **Correct**

**How can you conditionally render a component in React?**

* 

**By changing the state in the parent component**

* 

**Using conditional rendering in JSX**

**(Correct)**

* 

**By directly modifying the component's props**

* 

**Using the for loop to iterate through components**

**Explanation**

You can conditionally render a component in React by using conditional rendering in JSX. This involves using if statements or the ternary operator to determine whether to render the component based on a condition. It's a common practice for rendering components dynamically based on state or props.

Bottom of Form

Top of Form

Question 24: **Correct**

**Imagine you're building a task manager application in React. How would you handle conditional rendering of tasks based on their completion status?**

* 

**Create a separate component for each task status.**

* 

**Use the ternary operator to conditionally render tasks based on their completion status.**

**(Correct)**

* 

**Use a switch statement to toggle rendering based on status.**

* 

**Utilize conditional rendering using the "if" statement within the component's render method.**

**Explanation**

To handle conditional rendering based on a task's completion status, it's recommended to use the ternary operator. You can check the completion status and render different content or styles based on the result, ensuring a clean and efficient way to display tasks differently. This approach simplifies the code and improves readability.

Bottom of Form

Top of Form

Question 25: **Incorrect**

**Imagine you're building a form in a React application. How would you handle form input changes and form submission events efficiently?**

* 

**Manually manipulate the DOM to handle input changes**

* 

**Store form data in a single state object**

**(Correct)**

* 

**Use the 'onChange' event handler for every form input field**

**(Incorrect)**

* 

**Submit the form directly without validation**

**Explanation**

To handle form input changes efficiently, it's best to store form data in a single state object. This allows you to maintain a clean and controlled form state, making it easier to manage and validate data. Using the 'onChange' event handler for individual fields can lead to performance issues. Submitting the form without validation is risky, and manipulating the DOM directly in React is discouraged.

Bottom of Form

Top of Form

Question 26: **Correct**

**Imagine you are building a dashboard with multiple widgets. How would you utilize React components and props to ensure reusability and maintainability?**

* 

**Create a single monolithic component for the entire dashboard**

* 

**Use only global state management libraries like Redux for all widget communication**

* 

**Utilize context API to manage widget communication**

* 

**Build each widget as a separate component and use props to pass data and configuration**

**(Correct)**

**Explanation**

To ensure reusability and maintainability in a React dashboard, it's best to build each widget as a separate component and use props to pass data and configuration. This approach promotes component reusability and makes it easier to manage and update individual widgets without affecting the entire dashboard. It also aligns with React's component-based architecture.

Bottom of Form

Top of Form

Question 27: **Incorrect**

**How can you prevent unnecessary re-renders when rendering a list of components in React?**

* 

**Use unique and stable keys for each list item.**

**(Correct)**

* 

**Avoid rendering lists in React.**

* 

**Use array indices as keys.**

**(Incorrect)**

* 

**Use long and complex keys for better identification.**

**Explanation**

To prevent unnecessary re-renders in a list of components in React, it's essential to use unique and stable keys for each list item. React uses keys to determine when elements change, and using consistent keys helps React optimize rendering, resulting in improved performance and a better user experience.

Bottom of Form

Top of Form

Question 28: **Correct**

**When 'props' are passed down to a React component, they are accessible via the \_\_\_\_\_\_\_ object.**

* 

**this**

* 

**children**

* 

**state**

* 

**props**

**(Correct)**

**Explanation**

When 'props' are passed down to a React component, they are accessible via the props object. The props object contains the data and configuration passed to the component, and you can access these values using this.props.propertyName. This allows you to use the external data to render and customize the component.

Bottom of Form

Top of Form

Question 29: **Incorrect**

**In React, the key prop is used to uniquely identify elements in a \_\_\_\_\_\_\_.**

* 

**Array**

**(Correct)**

* 

**Class**

* 

**Function**

* 

**Component**

**(Incorrect)**

**Explanation**

In React, the key prop is used to uniquely identify elements in an array. It is especially crucial when rendering lists of elements to help React efficiently update and re-render only the elements that have changed, enhancing performance and preventing issues with elements' order.

Bottom of Form

Top of Form

Question 30: **Correct**

**A junior developer is facing issues understanding how data flows between components in a React application. How would you explain the concept of "props drilling" to them?**

* 

**"Props drilling is a way to send data from child components to their parent components."**

* 

**"Props drilling refers to the process of sending data between sibling components."**

* 

**"Props drilling is a term used to describe how data is passed down from a parent component to its child components."**

**(Correct)**

* 

**"Props drilling involves using Redux for managing component communication."**

**Explanation**

"Props drilling is a term used to describe how data is passed down from a parent component to its child components. It means that when you have nested components, you pass data (props) from a higher-level parent component to its children. This is a fundamental concept in React for sharing data and maintaining component hierarchy and relationships."

Bottom of Form

Top of Form

Question 31: **Incorrect**

**To ensure that a component re-renders when its 'props' change, you should use the \_\_\_\_\_\_\_ lifecycle method in a class component.**

* 

**render**

**(Incorrect)**

* 

**componentDidUpdate**

**(Correct)**

* 

**shouldUpdate**

* 

**componentWillMount**

**Explanation**

To ensure that a component re-renders when its 'props' change, you should use the componentDidUpdate lifecycle method in a class component. This method allows you to compare the previous props with the current props and trigger a re-render when necessary. It's a crucial part of managing component updates in React.

Bottom of Form

Top of Form

Question 32: **Correct**

**Describe how you can use the && operator for conditional rendering in React.**

* 

**By conditionally rendering components**

**(Correct)**

* 

**By using the if-else statement**

* 

**Using a for loop**

* 

**It's not possible in React**

**Explanation**

In React, you can use the && operator for conditional rendering. When you use this operator, the component following it will render if the condition before && is true. This is a concise and common way to conditionally render components based on certain conditions or user interactions. It's more efficient than using if-else statements in some cases.

Bottom of Form

Top of Form

Question 33: **Correct**

**How does React facilitate the creation of interactive UIs?**

* 

**By using a virtual DOM**

**(Correct)**

* 

**By using HTML and CSS**

* 

**By managing server databases**

* 

**By executing server-side code**

**Explanation**

React facilitates the creation of interactive UIs by using a virtual DOM (Document Object Model). The virtual DOM allows React to efficiently update only the parts of the UI that have changed, resulting in faster and more responsive user interfaces.

Bottom of Form

Top of Form

Question 34: **Incorrect**

**In React, event handlers are typically passed as \_\_\_\_\_\_\_ to components.**

* 

**HTML attributes**

**(Correct)**

* 

**Inline styles**

* 

**Component props**

**(Incorrect)**

* 

**Class methods**

**Explanation**

In React, event handlers are typically passed as HTML attributes to components. These attributes are used to define the behavior of the component in response to user interactions, such as clicking a button or submitting a form.

Bottom of Form

Top of Form

Question 35: **Correct**

**What are the implications of mutating 'props' directly inside a React component?**

* 

**It can cause unexpected behavior and make the component unpredictable.**

**(Correct)**

* 

**Mutating 'props' directly is the recommended way to update component data.**

* 

**It has no effect on the component's behavior.**

* 

**It enhances the component's performance.**

**Explanation**

Mutating 'props' directly inside a React component can lead to unexpected behavior and make the component unpredictable. React follows a unidirectional data flow, and 'props' should be treated as read-only. Instead of mutating 'props,' you should update 'state' or use callback functions to modify data.

Bottom of Form

Top of Form

Question 36: **Correct**

**In React, data passed from a parent component to a child component is referred to as \_\_\_\_\_\_\_.**

* 

**Props**

**(Correct)**

* 

**Methods**

* 

**Events**

* 

**State**

**Explanation**

In React, data passed from a parent component to a child component is referred to as "Props" (short for properties). Props are a way to pass data and configuration settings from parent to child components, allowing child components to receive and display data from their parent components.

Bottom of Form

Top of Form

Question 37: **Correct**

**Explain how the 'props' object works in React and how it is different from 'state'.**

* 

**'Props' are used for managing data, while 'state' is used for configuring component settings.**

* 

**'Props' are used for configuring component settings, while 'state' is used for managing data that can change over time.**

**(Correct)**

* 

**'Props' and 'state' are used interchangeably in React.**

* 

**'Props' are used for internal component communication. 'State' is used to configure initial component settings.**

**Explanation**

In React, 'props' are used for configuring component settings and passing data from a parent to a child component. 'State,' on the other hand, is used to manage data that can change over time within a component. Understanding the difference between 'props' and 'state' is crucial for effective React development.

Bottom of Form

Top of Form

Question 38: **Correct**

**What are the implications of calling 'setState' multiple times in a row within a React component?**

* 

**It causes an immediate crash**

* 

**It has no impact on the component**

* 

**All 'setState' calls are applied sequentially**

**(Correct)**

* 

**Only the last 'setState' call takes effect**

**Explanation**

When 'setState' is called multiple times in a row within a React component, all 'setState' calls are applied sequentially. React batches state updates and re-renders the component efficiently. The latest 'setState' call doesn't override the previous ones; instead, it accumulates changes. This is important to maintain the integrity of component state and ensure predictable behavior.

Bottom of Form

Top of Form

Question 39: **Correct**

**How can 'Props' be used in a React component?**

* 

**'defaultProps' provide default values for the component's 'props' when they are not explicitly provided.**

**(Correct)**

* 

**'defaultProps' are used to define default styling for a component.**

* 

**'defaultProps' determine the initial routing behavior of a component.**

* 

**'defaultProps' allow you to set the default state for a component.**

**Explanation**

'defaultProps' in React allow you to specify default values for a component's 'props.' When a 'prop' is not explicitly passed to a component, it will use the default value specified in 'defaultProps.' This is helpful for ensuring that a component behaves predictably even when not all 'props' are provided.

Bottom of Form

Top of Form

Question 40: **Incorrect**

**How can you bind an event handler to a React component?**

* 

**By setting 'eventHandler.bind(this)'**

* 

**By using the 'bind' method**

**(Correct)**

* 

**By using arrow functions**

**(Incorrect)**

* 

**By adding 'handler: function' in the component's props**

**Explanation**

To bind an event handler to a React component, you can use the 'bind' method. This ensures that the function is called with the correct 'this' context when the event is triggered. Arrow functions also automatically bind 'this,' making them a concise alternative.

Bottom of Form

Top of Form

Question 41: **Incorrect**

**Using non-unique keys in a list can lead to incorrect component \_\_\_\_\_\_\_ and unexpected behavior in a React application.**

* 

**Rendering**

**(Correct)**

* 

**Routing**

* 

**Communication**

**(Incorrect)**

* 

**Styling**

**Explanation**

Using non-unique keys in a list can lead to incorrect component rendering and unexpected behavior in a React application. Unique keys are essential for React to distinguish elements in a list properly. Without unique keys, React may struggle to identify and render components accurately, potentially causing issues.

Bottom of Form

Top of Form

Question 42: **Correct**

**To faciple components based on an array of data, you can use the \_\_\_\_\_\_\_ function.**

* 

**iterate()**

* 

**renderArray()**

* 

**loopThrough()**

* 

**map()**

**(Correct)**

**Explanation**

In React, to render multiple components based on an array of data, you can use the map() function. The map() function allows you to iterate over an array, apply a function to each item, and return an array of React components, making it a common choice for dynamic rendering.

Bottom of Form

Top of Form

Question 43: **Incorrect**

**How does React determine what changes to make to the DOM when keys are used in lists?**

* 

**React ignores keys when updating the DOM.**

* 

**React uses array indices to identify changes.**

* 

**React scans the entire DOM to find changes.**

**(Incorrect)**

* 

**React uses the 'key' prop to match elements.**

**(Correct)**

**Explanation**

React uses the 'key' prop to identify elements in lists. When rendering lists, React matches elements by their 'key' values, enabling efficient updates by identifying added, removed, or re-ordered elements. Using proper keys is crucial for React to determine what changes to make in the DOM.

Bottom of Form

Top of Form

Question 44: **Incorrect**

**To optimize a component's performance, React provides a method called \_\_\_\_\_\_\_ which can be used to prevent unnecessary re-renders.**

* 

**shouldUpdate**

* 

**shouldComponentUpdate**

**(Correct)**

* 

**preventRender**

**(Incorrect)**

* 

**optimizeComponent**

**Explanation**

To optimize a component's performance and prevent unnecessary re-renders, React provides the shouldComponentUpdate method. Developers can implement this method to control when a component should re-render, thus improving performance by avoiding unnecessary rendering cycles.

Bottom of Form

Top of Form

Question 45: **Correct**

**What is conditional rendering in React?**

* 

**Rendering components based on server data**

* 

**Displaying components conditionally based on state or props**

**(Correct)**

* 

**Rendering components asynchronously based on user input**

* 

**Rendering static components always**

**Explanation**

Conditional rendering in React refers to the practice of displaying components conditionally based on the state or props. It allows developers to show or hide specific parts of the UI based on certain conditions, making the UI dynamic and responsive to user interactions.

Bottom of Form

Top of Form

Question 46: **Incorrect**

**In a large e-commerce application, you need to display a list of products. How would you implement conditional rendering to show either the product list or a loading spinner?**

* 

**Load the product list and spinner simultaneously, then toggle their visibility.**

* 

**Use React's built-in Suspense component to handle conditional rendering.**

**(Correct)**

* 

**Utilize a setTimeout function to delay rendering the spinner for a few seconds.**

**(Incorrect)**

* 

**Use an "if-else" statement to conditionally render the list or the spinner.**

**Explanation**

To implement conditional rendering for a product list and a loading spinner, React's built-in Suspense component is the recommended approach. Suspense allows you to manage asynchronous operations like data loading and gracefully handle the display of a fallback (spinner) while waiting for the data to load. It improves user experience and is more robust than manual "if-else" statements.

Bottom of Form

Top of Form

Question 47: **Correct**

**How can you prevent the default action of a form submission in React?**

* 

**By using 'formAction' attribute**

* 

**By setting 'form.preventSubmit' to 'true'**

* 

**By using 'event.preventFormSubmit()'**

* 

**By calling 'event.preventDefault()'**

**(Correct)**

**Explanation**

To prevent the default action of a form submission in React, you can call 'event.preventDefault()' in the event handler function. This prevents the form from submitting and allows you to handle the submission manually, such as making an asynchronous request.

Bottom of Form

Top of Form

Question 48: **Correct**

**The \_\_\_\_\_\_\_ operator is often used for inline conditional rendering in React components.**

* 

**If-Else**

* 

**Ternary**

**(Correct)**

* 

**For-Loop**

* 

**While**

**Explanation**

The ternary operator is often used for inline conditional rendering in React components. It allows you to render different content or components based on a condition, making your UI dynamic and responsive without the need for traditional if-else statements.

Bottom of Form

Top of Form

Question 49: **Correct**

**Components in React can be written as either \_\_\_\_\_\_\_ components or \_\_\_\_\_\_\_ components.**

* 

**Server-side, Client-side**

* 

**Dynamic, Static**

* 

**Class, Function**

**(Correct)**

* 

**Webpack, Babel**

**Explanation**

Components in React can be written as either "Class components" or "Function components." Class components use ES6 classes to define components and have access to state and lifecycle methods, while Function components are simpler and use JavaScript functions to define components. Both are valid ways to create components in React.

Bottom of Form

Top of Form

Question 50: **Incorrect**

**The lifecycle method that is called immediately after a component is inserted into the DOM is \_\_\_\_\_\_\_.**

* 

**rendered()**

* 

**initialize()**

**(Incorrect)**

* 

**componentWillMount()**

* 

**componentDidMount()**

**(Correct)**

**Explanation**

The lifecycle method that is called immediately after a component is inserted into the DOM is componentDidMount(). This method is commonly used to perform tasks like data fetching or setting up event listeners once the component is part of the DOM.

Bottom of Form

Top of Form

Question 51: **Correct**

**A React component should never modify its own \_\_\_\_\_\_\_ directly.**

* 

**DOM elements**

**(Correct)**

* 

**Virtual DOM representation**

* 

**CSS styles**

* 

**State and props**

**Explanation**

A React component should never modify its own DOM elements directly. Instead, React handles the DOM updates through its virtual DOM reconciliation process. Modifying the DOM directly can lead to unexpected behavior and issues. Components should manage their state and props, and React will take care of updating the DOM accordingly.

Bottom of Form

Top of Form

Question 52: **Incorrect**

**What is the difference between a functional component and a class component in React?**

* 

**Functional components have lifecycle methods**

**(Correct)**

* 

**Functional components are more efficient**

**(Incorrect)**

* 

**Functional components use class-based syntax**

* 

**Class components have a simpler syntax**

**Explanation**

The primary difference is that functional components are simpler and more lightweight than class components. Functional components don't have their own lifecycle methods but can use hooks for similar functionality. Class components use a different, more verbose syntax and have lifecycle methods.

Bottom of Form

Top of Form

Question 53: **Correct**

**Why is it important to use keys when rendering a list of components in React?**

* 

**Keys help React identify and track components in the list**

**(Correct)**

* 

**Keys provide styling information for components**

* 

**Keys ensure that components are rendered in a random order**

* 

**Keys help reduce the size of the component tree**

**Explanation**

Using keys in React when rendering a list is crucial to help React identify and track components efficiently. It allows React to optimize updates and maintain the correct component state during re-renders, ensuring a smoother user experience.

Bottom of Form

Top of Form

Question 54: **Incorrect**

**You are tasked with implementing a feature that fetches data from an API and updates the UI accordingly. How would you manage the component's state and lifecycle to ensure a smooth user experience?**

* 

**Fetch data in the render method**

* 

**Use local component state for data storage**

* 

**Utilize Redux or a state management library**

**(Correct)**

* 

**Use 'props' for API data**

**(Incorrect)**

**Explanation**

To ensure a smooth user experience, it's best to use a state management library like Redux to manage the component's state for data fetched from an API. Redux helps centralize data and actions, making it easier to handle asynchronous data fetching and ensuring efficient UI updates. Using local state can work but can lead to unnecessary re-renders. Fetching data in the render method is a performance anti-pattern, and 'props' should be used for passing data to child components.

Bottom of Form