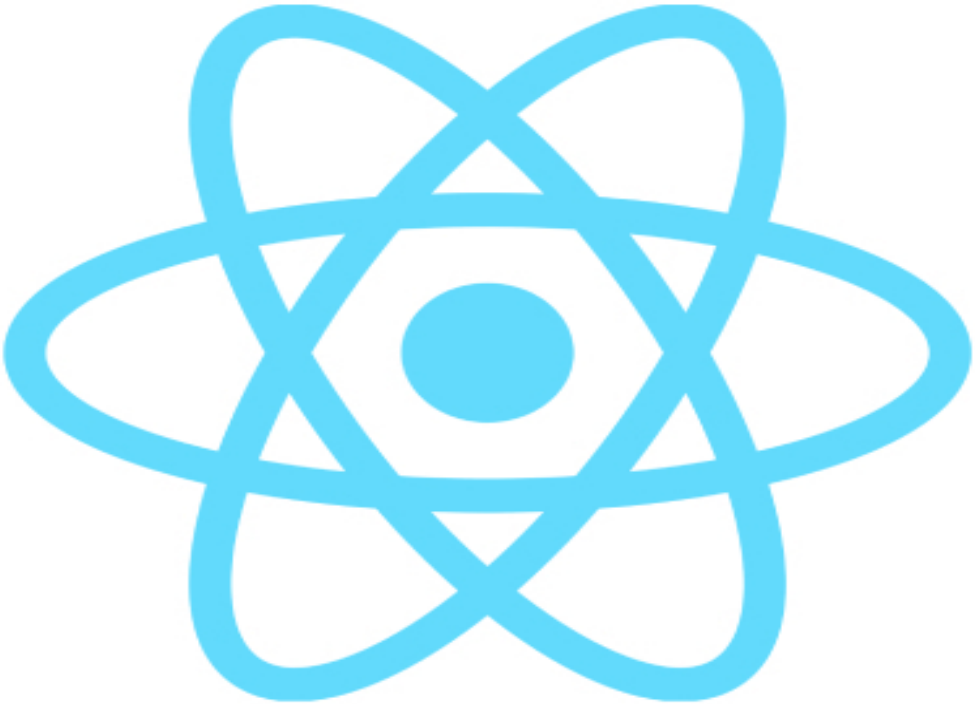
300+ React.js MCQs



Interview Questions and Answers

300+ REACT.JS

Interview Questions and Answers

MCQ Format

Created by: Manish Dnyandeo Salunke

Online Format: **https://bit.ly/online-courses-tests**

Question: What is React?

Answer Option 1: A programming language

Answer Option 2: A JavaScript library for building user interfaces

Answer Option 3: A database management system

Answer Option 4: A markup language

Correct Response: 2

Explanation: React is a JavaScript library for building user interfaces. It was developed by Facebook and is now maintained by a community of developers. React allows developers to create reusable UI components using JavaScript. These components can then be combined to create complex user interfaces.

[Reference: https://reactjs.org/]

Question: What are the major features of React?

Answer Option 1: Virtual DOM

Answer Option 2: Server-side rendering

Answer Option 3: Component reusability

Answer Option 4: One-way data binding

Correct Response: 1, 2, 3

Explanation: React has several major features, including a virtual DOM, server-side rendering, and component reusability. The virtual DOM helps improve performance by minimizing the number of updates needed to the actual DOM. Server-side rendering allows React to render components on the server before sending them to the client, improving performance and SEO. Component reusability allows developers to create reusable UI components using JavaScript. One-way data binding is not a major feature of React.

[Reference: https://reactjs.org/docs/faq-functions.html#what-is-react]

Question: What is JSX?

Answer Option 1: A new programming language

Answer Option 2: A JavaScript syntax extension

Answer Option 3: A design pattern

Answer Option 4: A testing framework

Correct Response: 2

Explanation: JSX is a JavaScript syntax extension that allows developers to write HTML-like code in JavaScript. JSX is not required to use React, but it is commonly used because it makes writing and managing UI components easier.

[Reference: https://reactjs.org/docs/introducing-jsx.html]

Question: What is the difference between Element and Component?

Answer Option 1: There is no difference

Answer Option 2: Element is a React class, while Component is a React function

Answer Option 3: Element represents a single HTML element, while Component can represent multiple HTML elements

Answer Option 4: Element is a plain JavaScript object, while Component is a function or class that returns an Element

Correct Response: 4

Explanation: In React, an element is a plain JavaScript object that represents a single HTML element. A component, on the other hand, is a function or class that returns an element (or multiple elements). Components are used to create reusable UI elements, while elements are used to represent the actual DOM elements that make up the UI.

[Reference: https://reactjs.org/docs/rendering-elements.html]

Question: How to create components in React?

Answer Option 1: Using plain HTML

Answer Option 2: Using a third-party library

Answer Option 3: Using React.createElement or JSX

Answer Option 4: Using jQuery

Correct Response: 3

Explanation: In React, components can be created using the React.createElement method or JSX syntax. React.createElement is a lower-level API for creating components, while JSX is a syntax extension that allows developers to write HTML-like code in JavaScript.

[Reference: https://reactjs.org/docs/components-and-props.html#function-and-class-components]

Question: When to use a Class Component over a Function Component?

Answer Option 1: When using Redux

Answer Option 2: When handling complex state and lifecycle methods

Answer Option 3: When rendering simple UI components

Answer Option 4: When using TypeScript

Correct Response: 2

Explanation: Class components in React are used when handling complex state and lifecycle methods. They provide additional functionality over function components, such as the ability to use lifecycle methods like componentDidMount and componentDidUpdate, and to maintain internal state. Function components are generally used for rendering simple UI components, while class components are used for more complex functionality.

[Reference: https://reactjs.org/docs/hooks-state.html]

Question: What are Pure Components?

Answer Option 1: Components that use the React.Pure API

Answer Option 2: Components that only have a render method

Answer Option 3: Components that implement shouldComponentUpdate

Answer Option 4: Components that use the useMemo hook

Correct Response: 3

Explanation: Pure components in React are components that implement the shouldComponentUpdate lifecycle method to improve performance. This method compares the current props and state to the next props and state, and determines whether the component needs to be re-rendered. If the props and state have not changed, the component can skip the rendering process.

[Reference: https://reactjs.org/docs/react-api.html#reactpurecomponent]

Question: What is state in React?

Answer Option 1: A component's properties

Answer Option 2: A component's internal data

Answer Option 3: A component's lifecycle methods

Answer Option 4: A component's HTML markup

Correct Response: 2

Explanation: In React, state refers to a component's internal data that can change over time. State is managed using the setState method, which allows developers to update the state of a component and trigger a re-render of the UI. State is used to keep track of data that changes in response to user input or other events.

[Reference: https://reactjs.org/docs/state-and-lifecycle.html#adding-local-state-to-a-class]

Question: What are props in React?

Answer Option 1: A component's internal data

Answer Option 2: A component's methods

Answer Option 3: A component's children

Answer Option 4: A component's properties

Correct Response: 4

Explanation: In React, props (short for "properties") are a component's input data that are passed down from a parent component. Props are read-only and cannot be changed by the child component. Props are used to customize the behavior and appearance of a component.

[Reference: https://reactjs.org/docs/components-and-props.html]

Question: What is the difference between state and props?

Answer Option 1: There is no difference

Answer Option 2: State is read-only, while props can be changed by the child component

Answer Option 3: Props are internal to a component, while state is external

Answer Option 4: State is managed within a component, while props are passed down from a parent component

Correct Response: 4

Explanation: The main difference between state and props in React is that state is managed within a component, while props are passed down from a parent component. State is used to manage a component's internal data, which can change over time, while props are used to customize a component's behavior and appearance.

[Reference: https://reactjs.org/docs/faq-state.html#what-is-the-difference-between-state-and-props]

Question: Why should we not update the state directly?

Answer Option 1: It can cause the component to re-render

Answer Option 2: It can cause race conditions

Answer Option 3: It can cause the application to crash

Answer Option 4: It can cause memory leaks

Correct Response: 1

Explanation: In React, state should not be updated directly because it can cause the component to re-render improperly. Instead, the setState method should be used to update state, which triggers a re-render of the component and ensures that the updated state is properly reflected in the UI.

[Reference: https://reactjs.org/docs/state-and-lifecycle.html#using-state-correctly]

Question: What is the purpose of callback function as an argument of setState()?

Answer Option 1: To handle errors

Answer Option 2: To improve performance

Answer Option 3: To update the state asynchronously

Answer Option 4: To synchronize state updates

Correct Response: 3

Explanation: The callback function as an argument of the setState method in React is used to update the state asynchronously. When the setState method is called, the state update may not happen immediately, which can cause issues when trying to access the updated state. The callback function is called after the state has been updated, ensuring that the updated state is available for use.

[Reference: https://reactjs.org/docs/react-component.html#setstate]

Question: What is the difference between HTML and React event handling?

Answer Option 1: There is no difference

Answer Option 2: HTML uses camelCase event names, while React uses kebab-case

Answer Option 3: React uses synthetic events, while HTML does not

Answer Option 4: HTML uses inline event handlers, while React uses event listeners

Correct Response: 3

Explanation: The main difference between HTML and React event handling is that React uses synthetic events, while HTML does not. Synthetic events are cross-browser compatible and behave consistently across different platforms. They are also optimized for performance by pooling event objects.

[Reference: https://reactjs.org/docs/events.html]

Question: How to bind methods or event handlers in JSX callbacks?

Answer Option 1: Using the bind method

Answer Option 2: Using arrow functions

Answer Option 3: Using the this keyword

Answer Option 4: Using the new keyword

Correct Response: 2

Explanation: Methods or event handlers can be bound in JSX callbacks using arrow functions. Arrow functions inherit the context of their parent scope, which allows them to access the component's methods and state. The bind method and the this keyword can also be used to bind methods or event handlers, but arrow functions are the recommended approach.

[Reference: https://reactjs.org/docs/handling-events.html#passing-arguments-to-event-handlers]

Question: How to pass a parameter to an event handler or callback?

Answer Option 1: Using the event object

Answer Option 2: Using the setState method

Answer Option 3: Using the bind method

Answer Option 4: Using an arrow function

Correct Response: 4

Explanation: A parameter can be passed to an event handler or callback in React by using an arrow function. The arrow function takes the event as a parameter, along with any additional parameters that need to be passed. This approach ensures that the event object is properly handled and that the correct parameters are passed to the event handler or callback.

[Reference: https://reactjs.org/docs/handling-events.html#passing-arguments-to-event-handlers]

Question: What are synthetic events in React?

Answer Option 1: Events that occur in a virtual environment

Answer Option 2: Events that are cross-browser compatible and behave consistently across different platforms

Answer Option 3: Events that trigger multiple times

Answer Option 4: Events that are not optimized for performance

Correct Response: 2

Explanation: Synthetic events in React are events that are cross-browser compatible and behave consistently across different platforms. They are also optimized for performance by pooling event objects, which allows them to be reused and prevents excessive memory allocation. Synthetic events have the same interface as native events, but they are implemented differently behind the scenes.

[Reference: https://reactjs.org/docs/events.html#event-pooling]

Question: What are inline conditional expressions?

Answer Option 1: Expressions that are evaluated during runtime

Answer Option 2: Expressions that are evaluated during compilation

Answer Option 3: Expressions that are used to conditionally render components

Answer Option 4: Expressions that are used to set component state

Correct Response: 3

Explanation: Inline conditional expressions in React are expressions that are used to conditionally render components based on a certain condition. They are typically used in JSX, and are written using the ternary operator. Inline conditional expressions are a simple and effective way to conditionally render components without the need for additional logic.

[Reference: https://reactjs.org/docs/conditional-rendering.html#inline-if-with-conditional-operator]

Question: What is "key" prop and what is the benefit of using it in arrays of elements?

Answer Option 1: A prop that specifies the position of an element in an array

Answer Option 2: A prop that provides a unique identifier for each element in an array

Answer Option 3: A prop that specifies the type of each element in an array

Answer Option 4: A prop that determines the order of elements in an array

Correct Response: 2

Explanation: The "key" prop in React is a special prop that provides a unique identifier for each element in an array of elements. It is used to optimize the performance of rendering by allowing React to identify which elements have changed, and to update only those elements instead of re-rendering the entire list. The "key" prop should be a unique and stable identifier for each element, such as an ID or index.

[Reference: https://reactjs.org/docs/lists-and-keys.html#keys]

Question: What is the use of refs?

Answer Option 1: To create references to DOM elements

Answer Option 2: To create references to component instances

Answer Option 3: To handle events

Answer Option 4: To manage state

Correct Response: 1

Explanation: Refs in React are used to create references to DOM elements. They allow developers to access and manipulate the properties and methods of DOM elements directly, without the need for additional logic or event handling. Refs are typically used when working with third-party libraries, or when manipulating the DOM directly is necessary.

[Reference: https://reactjs.org/docs/refs-and-the-dom.html#creating-refs]

Question: How to create refs?

Answer Option 1: Using the React.createRef method

Answer Option 2: Using the document.getElementById method

Answer Option 3: Using the this.refs object

Answer Option 4: Using the window object

Correct Response: 1

Explanation: Refs in React can be created using the React.createRef method. This method creates a new ref object, which can be attached to a DOM element or a React component. Refs can be accessed using the current property of the ref object.

[Reference: https://reactjs.org/docs/refs-and-the-dom.html#creating-refs]

Question: What are forward refs?

Answer Option 1: A way to pass props down to child components

Answer Option 2: A way to pass state up to parent components

Answer Option 3: A way to forward refs from parent components to child components

Answer Option 4: A way to declare and initialize a component's state

Correct Response: 3

Explanation: Forward refs are a way to pass a ref from a parent component to a child component. This allows the parent component to access and manipulate the child component's DOM node. Forward refs are created using the React.forwardRef() function.

[Reference: https://reactjs.org/docs/forwarding-refs.html]

Question: Why are String Refs legacy?

Answer Option 1: They are slow and inefficient

Answer Option 2: They are not supported in modern browsers

Answer Option 3: They can cause naming conflicts and bugs

Answer Option 4: They are no longer needed with the introduction of functional components

Correct Response: 3

Explanation: String refs, which allow developers to set refs using a string identifier, are considered legacy because they can cause naming conflicts and bugs. They have been replaced with callback refs and the React.createRef() API.

[Reference: https://reactjs.org/docs/refs-and-the-dom.html#legacy-api-string-refs]

Question: What is Virtual DOM?

Answer Option 1: A virtual machine that runs JavaScript code in the browser

Answer Option 2: A representation of the browser's DOM in memory

Answer Option 3: A type of web server used for server-side rendering

Answer Option 4: A tool for debugging React components

Correct Response: 2

Explanation: The Virtual DOM is a representation of the browser's DOM in memory. It is used by React to optimize and speed up updates to the actual DOM by minimizing the number of changes that need to be made. When a component's state changes, React updates the Virtual DOM and compares it to the previous version. It then calculates the most efficient way to update the actual DOM based on the differences between the two versions.

[Reference: https://reactjs.org/docs/faq-internals.html#what-is-the-virtual-dom]

Question: How Virtual DOM works?

Answer Option 1: It updates the actual DOM directly

Answer Option 2: It creates a new DOM tree on each update

Answer Option 3: It updates the Virtual DOM and calculates the most efficient way to update the actual DOM

Answer Option 4: It updates only the changed parts of the actual DOM

Correct Response: 3

Explanation: When a component's state changes in React, it updates the Virtual DOM and calculates the most efficient way to update the actual DOM based on the differences between the two versions. This approach minimizes the number of changes that need to be made to the actual DOM and improves performance.

[Reference: https://reactjs.org/docs/faq-internals.html#what-is-the-virtual-dom]

Question: What is the difference between Shadow DOM and Virtual DOM?

Answer Option 1: Shadow DOM is used for server-side rendering, while Virtual DOM is used for client-side rendering

Answer Option 2: Shadow DOM is a browser feature, while Virtual DOM is a React feature

Answer Option 3: Shadow DOM is used for styling and encapsulation, while Virtual DOM is used for performance optimization

Answer Option 4: There is no difference

Correct Response: 2

Explanation: Shadow DOM and Virtual DOM are two different concepts. Shadow DOM is a browser feature that allows developers to encapsulate styles and markup within components. Virtual DOM, on the other hand, is a React feature that allows for performance optimization by minimizing changes to the actual DOM.

[Reference: https://reactjs.org/docs/web-components.html#react-and-web-components]

Question: What is React Fiber?

Answer Option 1: A new programming language developed by Facebook

Answer Option 2: A new JavaScript engine developed by Facebook

Answer Option 3: A new rendering engine for React

Answer Option 4: A new component lifecycle method in React

Correct Response: 3

Explanation: React Fiber is a new rendering engine for React that was introduced in React 16. It is designed to improve performance and enable more flexibility in scheduling updates. With React Fiber, React can break up large updates into smaller chunks, prioritize updates, and pause and resume updates as needed.

[Reference: https://reactjs.org/blog/2017/09/26/react-v16.0.html]

Question: What is the main goal of React Fiber?

Answer Option 1: To improve server-side rendering performance

Answer Option 2: To improve client-side rendering performance

Answer Option 3: To simplify the React API

Answer Option 4: To improve the debugging experience in React

Correct Response: 2

Explanation: The main goal of React Fiber is to improve client-side rendering performance. It does this by introducing a new algorithm for rendering updates that is more efficient and flexible than the previous algorithm. With React Fiber, React can break up large updates into smaller chunks, prioritize updates, and pause and resume updates as needed.

[Reference: https://reactjs.org/blog/2017/09/26/react-v16.0.html]

Question: What are controlled components?

Answer Option 1: Components that are managed by React and cannot be updated directly

Answer Option 2: Components that are updated using refs

Answer Option 3: Components that are updated using the setState() method

Answer Option 4: Components that store their own state

Correct Response: 3

Explanation: Controlled components are components that store their state in a parent component and are updated using the setState() method. The parent component passes down the component's state as props and also passes down a function to update the state when needed. This approach allows for more control over the component's behavior and simplifies debugging.

[Reference: https://reactjs.org/docs/forms.html#controlled-components]

Question: What are uncontrolled components?

Answer Option 1: Components that are managed by React and cannot be updated directly

Answer Option 2: Components that are updated using refs

Answer Option 3: Components that are updated using the setState() method

Answer Option 4: Components that store their own state

Correct Response: 2

Explanation: Uncontrolled components are components that store their own state and are updated using refs. They are often used for simple form inputs, where managing the state in a parent component would be unnecessary overhead.

[Reference: https://reactjs.org/docs/uncontrolled-components.html]

Question: What is the difference between createElement and cloneElement?

Answer Option 1: createElement is used to create new DOM elements, while cloneElement is used to clone existing elements

Answer Option 2: createElement is used to clone existing elements, while cloneElement is used to create new elements

Answer Option 3: createElement is a class method, while cloneElement is an instance method

Answer Option 4: There is no difference

Correct Response: 1

Explanation: createElement is a method used to create new React elements, while cloneElement is a method used to clone existing React elements and pass new props to the cloned element.

[Reference: https://reactjs.org/docs/react-api.html#createelement]

[Reference: https://reactjs.org/docs/react-api.html#cloneelement]

Question: What is Lifting State Up in React?

Answer Option 1: A technique for passing data down from parent components to child components

Answer Option 2: A technique for passing data up from child components to parent components

Answer Option 3: A technique for managing component state in a centralized location

Answer Option 4: A technique for styling components using CSS-in-JS

Correct Response: 2

Explanation: Lifting State Up is a technique in React for passing data up from child components to parent components. This is useful when multiple components need to share the same state or when a child component needs to update the state of a parent component.

[Reference: https://reactjs.org/docs/lifting-state-up.html]

Question: What are Higher-Order components?

Answer Option 1: Components that render other components

Answer Option 2: Components that are used for server-side rendering

Answer Option 3: Components that use the shouldComponentUpdate() method

Answer Option 4: Components that enhance the behavior of other components

Correct Response: 4

Explanation: Higher-Order components (HOCs) are components that enhance the behavior of other components by adding additional functionality or props. HOCs take a component as input and return a new component that includes the additional behavior. This allows developers to reuse code and separate concerns in their applications.

[Reference: https://reactjs.org/docs/higher-order-components.html]

Question: How to create props proxy for HOC component?

Answer Option 1: By wrapping the component with another component that adds the additional props

Answer Option 2: By using the setState() method to add the additional props to the component's state

Answer Option 3: By passing the additional props as arguments to the HOC function

Answer Option 4: By using the this.props object to add the additional props to the component's existing props

Correct Response: 1

Explanation: To create a props proxy for an HOC component, you can wrap the component with another component that adds the additional props. This is done by defining a new component that takes the original component as input and returns a new component that includes the additional props.

[Reference: https://reactjs.org/docs/higher-order-components.html#convention-wrap-the-display-name-for-easy-debugging]

Question: What is context?

Answer Option 1: A global object used to store application state

Answer Option 2: A way to pass data down to child components without using props

Answer Option 3: A way to pass data up to parent components without using props

Answer Option 4: A JavaScript library for data visualization

Correct Response: 2

Explanation: Context is a way to pass data down to child components without using props. It is useful for data that needs to be accessed by many components at different levels of the component hierarchy. Context provides a way to avoid the "prop drilling" problem, where props need to be passed down through many layers of components.

[Reference: https://reactjs.org/docs/context.html]

Question: What is children prop?

Answer Option 1: A prop that contains the child components of a component

Answer Option 2: A prop that contains the parent component of a component

Answer Option 3: A prop that is used to update a component's state

Answer Option 4: A prop that is passed to the constructor() method of a component

Correct Response: 1

Explanation: The children prop is a special prop in React that contains the child components of a component. It allows components to render their child components directly, without having to pass them down through props. The children prop can be used with any component, including functional components.

[Reference: https://reactjs.org/docs/composition-vs-inheritance.html]

Question: How to write comments in React?

Answer Option 1: By using the // syntax

Answer Option 2: By using the <!-- --> syntax

Answer Option 3: By using the /* */ syntax

Answer Option 4: By using the {/* */} syntax

Correct Response: 4

Explanation: In React, comments can be written using the {/* */} syntax. This syntax allows comments to be included directly in JSX code without causing syntax errors. Comments can be used to provide additional information or documentation about components, or to temporarily disable parts of the code for debugging purposes.

[Reference: https://reactjs.org/docs/jsx-in-depth.html#jsx-comments]

Question: What is the purpose of using super constructor with props argument?

Answer Option 1: To create a new instance of a component

Answer Option 2: To access the parent component's state

Answer Option 3: To pass props to the parent component

Answer Option 4: To initialize the component's props and state

Correct Response: 4

Explanation: In React, the super() constructor with props argument is used to initialize the component's props and state. It is required when defining a constructor in a class component, and should always be called before accessing this.props or this.state.

[Reference: https://reactjs.org/docs/react-component.html#constructor]

Question: What is reconciliation?

Answer Option 1: The process of rendering a component to the DOM

Answer Option 2: The process of comparing two versions of a component and updating the DOM with the changes

Answer Option 3: The process of defining the structure and behavior of a component

Answer Option 4: The process of testing a component for bugs and errors

Correct Response: 2

Explanation: Reconciliation is the process of comparing two versions of a component and updating the DOM with the changes. When a component's state or props change, React compares the new version of the component to the previous version and calculates the minimum number of changes needed to update the DOM. This process is called reconciliation and is one of the key features of React that makes it so efficient.

[Reference: https://reactjs.org/docs/reconciliation.html]

Question: How to set state with a dynamic key name?

Answer Option 1: By using the setState() method with a variable key name

Answer Option 2: By using the this.state object with a variable key name

Answer Option 3: By using the this.props object with a variable key name

Answer Option 4: By using the this.context object with a variable key name

Correct Response: 1

Explanation: To set state with a dynamic key name, you can use the setState() method with a variable key name. This allows you to create new state keys based on user input or other dynamic data.

[Reference: https://stackoverflow.com/questions/29537299/react-how-to-setstate-with-a-dynamic-key-name]

Question: What would be the common mistake of function being called every time the component renders?

Answer Option 1: Not using React.memo() to memoize the component

Answer Option 2: Not using React.lazy() to lazy-load the component

Answer Option 3: Not using the shouldComponentUpdate() method to prevent unnecessary re-renders

Answer Option 4: Declaring the function inside the component's render method

Correct Response: 4

Explanation: A common mistake that can cause a function to be called every time the component renders is declaring the function inside the component's render method. This can cause the function to be recreated every time the component renders, even if the function's dependencies have not changed. To avoid this problem, functions should be declared outside the render method, or in a separate file if they are reused across multiple components.

[Reference: https://kentcdodds.com/blog/dont-call-a-function-inside-of-the-render-method]

Question: Why React uses className over class attribute?

Answer Option 1: To avoid naming conflicts with JavaScript class keyword

Answer Option 2: To make the markup more concise

Answer Option 3: To improve performance

Answer Option 4: To follow HTML5 standard

Correct Response: 1

Explanation: React uses className instead of class attribute to avoid naming conflicts with the JavaScript class keyword. Using className also makes it easier to use CSS modules and other tools that work with class names.

[Reference: https://reactjs.org/docs/dom-elements.html#classname]

Question: What are fragments?

Answer Option 1: Special React components

Answer Option 2: An alternative to using HTML tags for styling

Answer Option 3: A way to group multiple elements without adding an extra DOM node

Answer Option 4: A way to create custom HTML elements

Correct Response: 3

Explanation: Fragments are a way to group multiple elements without adding an extra DOM node. Fragments are useful when you want to return multiple elements from a component, but don't want to add an unnecessary container element to the DOM.

[Reference: https://reactjs.org/docs/fragments.html]

Question: Why fragments are better than container divs?

Answer Option 1: Fragments are more performant

Answer Option 2: Fragments are more semantic

Answer Option 3: Fragments are easier to use

Answer Option 4: Fragments are not better than container divs

Correct Response: 1

Explanation: Fragments are better than container divs because they are more performant. Using a container div adds an extra DOM node, which can slow down rendering and create problems with styling. Fragments, on the other hand, allow you to group elements without adding any extra nodes to the DOM.

[Reference: https://reactjs.org/docs/fragments.html#motivation]

Question: What are portals in React?

Answer Option 1: A way to communicate between components

Answer Option 2: A way to render a component's children in a different part of the DOM

Answer Option 3: A way to handle errors in components

Answer Option 4: A way to create dynamic forms

Correct Response: 2

Explanation: Portals in React provide a way to render a component's children in a different part of the DOM. This can be useful for creating modal dialogs, tooltips, and other UI components that need to be positioned outside the normal flow of the page.

[Reference: https://reactjs.org/docs/portals.html]

Question: What are stateless components?

Answer Option 1: Components that don't have any children

Answer Option 2: Components that don't use any state

Answer Option 3: Components that don't have any props

Answer Option 4: Components that are only used for layout

Correct Response: 2

Explanation: Stateless components, also known as functional components, are components that don't use any state. Stateless components are simpler and easier to test than stateful components, but they can't be used for more complex UI components that need to maintain state.

[Reference: https://reactjs.org/docs/components-and-props.html#functional-and-class-components]

Question: What are stateful components?

Answer Option 1: Components that don't use any state

Answer Option 2: Components that are only used for layout

Answer Option 3: Components that maintain state

Answer Option 4: Components that only have props

Correct Response: 3

Explanation: Stateful components, also known as class components, are components that maintain state. Stateful components are useful for creating more complex UI components that need to maintain state, but they are more difficult to test than stateless components.

[Reference: https://reactjs.org/docs/components-and-props.html#function-and-class-components]

Question: How to apply validation on props in React?

Answer Option 1: Use the PropTypes library

Answer Option 2: Use the Validation library

Answer Option 3: Use the React-Validator library

Answer Option 4: Use the ValidateJS library

Correct Response: 1

Explanation: In React, you can apply validation to props using the PropTypes library. PropTypes allow you to define the type and shape of a component's props, which helps catch errors and bugs early in development.

[Reference: https://reactjs.org/docs/typechecking-with-proptypes.html]

Question: What are the advantages of React?

Answer Option 1: Improved performance

Answer Option 2: Reusable components

Answer Option 3: Easy to learn

Answer Option 4: Good community support

Correct Response: 1, 2, 4

Explanation: React has several advantages, including improved performance, reusable components, and good community support. React's use of a virtual DOM and its focus on component-based architecture help improve performance and make it easier to develop complex UI components. React's component-based architecture also makes it easier to create reusable components that can be used across multiple projects. React has a large and active community of developers, which provides good support and resources for learning and development.

Question: What are the limitations of React?

Answer Option 1: Steep learning curve

Answer Option 2: Limited SEO optimization

Answer Option 3: Poor performance on large datasets

Answer Option 4: Only supports front-end development

Correct Response: 1, 2

Explanation: React has some limitations, including a steep learning curve and limited SEO optimization. React's focus on component-based architecture can make it more difficult for beginners to learn and understand, and its reliance on JavaScript can make it less friendly to search engines. React can also struggle with performance on large datasets or complex UI components. React is primarily used for front-end development and does not have built-in support for back-end development.

Question: What are error boundaries in React v16?

Answer Option 1: A way to handle errors in React components

Answer Option 2: A way to render components outside the normal DOM

Answer Option 3: A way to optimize rendering performance

Answer Option 4: A way to create dynamic forms

Correct Response: 1

Explanation: Error boundaries are a way to handle errors in React components. In React v16, error boundaries are special components that catch and handle errors that occur during rendering. This can help prevent the entire application from crashing due to an error in a single component.

[Reference: https://reactjs.org/docs/error-boundaries.html]

Question: How are error boundaries handled in React v15?

Answer Option 1: React v15 does not support error boundaries

Answer Option 2: Error boundaries are handled the same way as in React v16

Answer Option 3: Error boundaries are not supported in class components

Answer Option 4: Error boundaries are not supported in functional components

Correct Response: 1

Explanation: Error boundaries were not supported in React v15. Error handling in React v15 was less robust and could lead to the entire application crashing if an error occurred during rendering.

[Reference: https://reactjs.org/docs/error-boundaries.html#introducing-error-boundaries]

Question: What are the recommended ways for static type checking?

Answer Option 1: PropTypes library

Answer Option 2: TypeScript

Answer Option 3: Flow

Answer Option 4: ESLint

Correct Response: 2, 3

Explanation: There are several recommended ways to implement static type checking in React, including using TypeScript or Flow. TypeScript and Flow are both static type checkers that can help catch errors and bugs before they happen. The PropTypes library is another way to perform type checking, but it is less powerful than TypeScript or Flow. ESLint is a code linter that can help catch errors and enforce best practices, but it is not a type checker.

[Reference: https://reactjs.org/docs/static-type-checking.html]

Question: What is the use of react-dom package?

Answer Option 1: To create React components

Answer Option 2: To render React components to the DOM

Answer Option 3: To manage state in React components

Answer Option 4: To handle routing in React components

Correct Response: 2

Explanation: The react-dom package is used to render React components to the DOM. The react-dom package provides several methods for rendering components, including the render method and the hydrate method.

[Reference: https://reactjs.org/docs/react-dom.html]

Question: What is the purpose of render method of react-dom?

Answer Option 1: To render a React component to the server

Answer Option 2: To render a React component to the client

Answer Option 3: To render a React component to a canvas

Answer Option 4: To render a React component to a file

Correct Response: 2

Explanation: The render method of the react-dom package is used to render a React component to the client-side DOM. The render method takes two arguments: the component to render and the DOM element to render it to.

[Reference: https://reactjs.org/docs/react-dom.html#render]

Question: What is ReactDOMServer?

Answer Option 1: A way to render React components on the server

Answer Option 2: A way to manage state in React components

Answer Option 3: A way to handle routing in React components

Answer Option 4: A way to create dynamic forms in React components

Correct Response: 1

Explanation: ReactDOMServer is a module that allows you to render React components on the server. ReactDOMServer provides several methods for rendering components, including the renderToString method and the renderToStaticMarkup method.

[Reference: https://reactjs.org/docs/react-dom-server.html]

Question: How to use InnerHtml in React?

Answer Option 1: Use the dangerouslySetInnerHTML prop

Answer Option 2: Use the InnerHtml component

Answer Option 3: Use the innerHTML attribute

Answer Option 4: Use the HTML component

Correct Response: 1

Explanation: In React, you can use the dangerouslySetInnerHTML prop to set the inner HTML of a component. The dangerouslySetInnerHTML prop is used to bypass React's built-in sanitization and allow arbitrary HTML to be injected into a component. However, this should be used with caution, as it can pose a security risk.

[Reference: https://reactjs.org/docs/dom-elements.html#dangerouslysetinnerhtml]

Question: How to use styles in React?

Answer Option 1: Use the style attribute

Answer Option 2: Use the className attribute

Answer Option 3: Use the CSS module

Answer Option 4: Use the inline-style attribute

Correct Response: 1

Explanation: In React, you can use the style attribute to apply styles to a component. The style attribute takes an object that contains CSS properties and values, similar to inline styles in HTML.

[Reference: https://reactjs.org/docs/dom-elements.html#style]

Question: How events are different in React?

Answer Option 1: Events in React are synthetic events

Answer Option 2: React events are camelCase instead of lowercase

Answer Option 3: React events can be attached to any DOM element

Answer Option 4: React events have a different API than native events

Correct Response: 1, 2, 4

Explanation: Events in React are different from native events in several ways. React events are synthetic events that are cross-browser compatible and have the same API across all browsers. React events are also camelCase instead of lowercase (e.g. onClick instead of onclick). Finally, React events have a different API than native events, including the ability to call preventDefault and stopPropagation on the event object.

[Reference: https://reactjs.org/docs/events.html]

Question: What will happen if you use setState in constructor?

Answer Option 1: Nothing

Answer Option 2: It will cause a memory leak

Answer Option 3: It will update the component state

Answer Option 4: It will cause an error

Correct Response: 3

Explanation: If you use setState in a component's constructor, it will update the component's state. However, this is generally not recommended, as it can cause problems with initialization and lead to confusing code. It is better to set the initial state in the constructor using the this.state property.

[Reference: https://reactjs.org/docs/react-component.html#constructor]

Question: What is the impact of indexes as keys?

Answer Option 1: Improves performance

Answer Option 2: No impact

Answer Option 3: Causes issues with component reusability

Answer Option 4: Can cause issues with component state

Correct Response: 4

Explanation: Using indexes as keys can cause issues with component state, particularly if the order of the items in the list changes. It can also lead to issues with duplicate keys if items are added or removed from the list. It is generally recommended to use a unique identifier as the key, rather than an index.

[Reference: https://reactjs.org/docs/lists-and-keys.html#keys]

Question: What will happen if you use props in initial state?

Answer Option 1: It will work as expected

Answer Option 2: It will cause a runtime error

Answer Option 3: It will ignore the props

Answer Option 4: It will cause a memory leak

Correct Response: 3

Explanation: Using props in initial state will cause the initial state to ignore the props. This is because the initial state is only set once, when the component is first created. If you want to use props to initialize state, you should set the state in the constructor instead.

[Reference: https://reactjs.org/docs/react-component.html#constructor]

Question: How do you conditionally render components?

Answer Option 1: Use the "if" statement in JSX

Answer Option 2: Use the ternary operator in JSX

Answer Option 3: Use the switch statement in JSX

Answer Option 4: Use the "render" method in JSX

Correct Response: 2

Explanation: To conditionally render components in React, you can use the ternary operator in JSX. This allows you to specify a condition, and render one component if the condition is true, and another component if the condition is false. You can also use other conditional statements, such as "if" statements or switch statements, outside of the JSX to determine which component to render.

[Reference: https://reactjs.org/docs/conditional-rendering.html]

Question: Why we need to be careful when spreading props on DOM elements?

Answer Option 1: It causes a performance issue

Answer Option 2: It can cause security issues

Answer Option 3: It can cause the component to render incorrectly

Answer Option 4: It can spread invalid HTML attributes

Correct Response: 4

Explanation: When spreading props on DOM elements, it is important to be careful because it can spread invalid HTML attributes. This can cause the component to render incorrectly or even create security vulnerabilities. It is recommended to only spread props on components that you trust, or to manually whitelist the valid attributes.

[Reference: https://reactjs.org/docs/jsx-in-depth.html#spread-attributes]

Question: How you use decorators in React?

Answer Option 1: Use them as HOCs

Answer Option 2: Use them as class decorators

Answer Option 3: Use them as function decorators

Answer Option 4: Use them as event handlers

Correct Response: 2

Explanation: Decorators in React can be used as class decorators to add functionality to a component class. For example, you can use a decorator to add additional lifecycle methods or state to a component. Decorators can be written as higher-order components (HOCs) or as regular functions.

[Reference: https://reactjs.org/docs/higher-order-components.html#convention-wrap-the-display-name-for-easy-debugging]

Question: How do you memoize a component?

Answer Option 1: Use the "shouldComponentUpdate" lifecycle method

Answer Option 2: Use the "componentDidUpdate" lifecycle method

Answer Option 3: Use the "React.memo" higher-order component

Answer Option 4: Use the "React.useMemo" hook

Correct Response: 3

Explanation: To memoize a component in React, you can use the "React.memo" higher-order component. This will prevent the component from re-rendering if the props have not changed. You can also use the "shouldComponentUpdate" lifecycle method or the "React.useMemo" hook to achieve similar results.

[Reference: https://reactjs.org/docs/react-api.html#reactmemo]

Question: How you implement Server-Side Rendering or SSR?

Answer Option 1: Use a third-party library

Answer Option 2: Use the "ReactDOMServer" module

Answer Option 3: Use the "ReactServer" package

Answer Option 4: Use the "server-render" package

Correct Response: 2

Explanation: To implement server-side rendering (SSR) in React, you can use the "ReactDOMServer" module. This module provides a method called "renderToString" that allows you to render a component to HTML on the server. You can then send this HTML to the client, where it can be hydrated into a full React application.

[Reference: https://reactjs.org/docs/react-dom-server.html]

Question: How to enable production mode in React?

Answer Option 1: Set the NODE_ENV environment variable to "production"

Answer Option 2: Use the "process.env.NODE_ENV" variable in the code

Answer Option 3: Use the "React.setProductionMode(true)" method

Answer Option 4: Enable it in the browser console

Correct Response: 1

Explanation: To enable production mode in React, you can set the NODE_ENV environment variable to "production". This will trigger various optimizations and remove development-only code from the bundle. You can also use the "process.env.NODE_ENV" variable in the code to conditionally enable certain features.

[Reference: https://create-react-app.dev/docs/deployment/#enabling-production-mode]

Question: What is CRA and its benefits?

Answer Option 1: A testing framework for React

Answer Option 2: A tool for managing React dependencies

Answer Option 3: A build tool for React applications

Answer Option 4: A boilerplate for creating React applications

Correct Response: 4

Explanation: CRA stands for Create React App, which is a boilerplate for creating React applications. It provides a pre-configured setup for building, testing, and deploying React applications, allowing developers to focus on writing code rather than setting up the build toolchain. Some benefits of using CRA include easy setup, automatic configuration, and a built-in development server.

[Reference: https://create-react-app.dev/docs/getting-started/]

Question: What is the lifecycle methods order in mounting?

Answer Option 1: componentWillMount, componentDidMount, componentWillReceiveProps

Answer Option 2: componentDidMount, componentWillMount, componentWillReceiveProps

Answer Option 3: componentWillMount, componentWillReceiveProps, componentDidMount

Answer Option 4: componentDidMount, componentWillReceiveProps, componentWillMount

Correct Response: 3

Explanation: The lifecycle methods for mounting a component in React are as follows: componentWillMount, render, componentDidMount. The "componentWillMount" method is called before the component is mounted to the DOM, "render" is called to render the component, and "componentDidMount" is called after the component is mounted to the DOM.

[Reference: https://reactjs.org/docs/react-component.html#the-component-lifecycle]

Question: What are the lifecycle methods going to be deprecated in React v16?

Answer Option 1: componentWillMount and componentWillReceiveProps

Answer Option 2: componentWillUpdate and componentDidUpdate

Answer Option 3: componentWillMount and componentWillUpdate

Answer Option 4: componentWillReceiveProps and componentWillUpdate

Correct Response: 4

Explanation: The "componentWillReceiveProps" and "componentWillUpdate" lifecycle methods are going to be deprecated in React v16. These methods will be replaced with new lifecycle methods that are more efficient and easier to reason about. The new methods are "getDerivedStateFromProps" and "getSnapshotBeforeUpdate".

[Reference: https://reactjs.org/blog/2018/03/27/update-on-async-rendering.html]

Question: What is the purpose of getDerivedStateFromProps() lifecycle method?

Answer Option 1: To update the state based on props changes

Answer Option 2: To update the props based on state changes

Answer Option 3: To fetch data from an API

Answer Option 4: To handle events

Correct Response: 1

Explanation: The "getDerivedStateFromProps" lifecycle method in React is used to update the state based on changes to props. This method is called every time the component is updated and can return a new state object. It is commonly used to synchronize the state with the props in response to user input or server-side changes.

[Reference: https://reactjs.org/docs/react-component.html#static-getderivedstatefromprops]

Question: What is the purpose of getSnapshotBeforeUpdate() lifecycle method?

Answer Option 1: To update the state based on props changes

Answer Option 2: To update the props based on state changes

Answer Option 3: To fetch data from an API

Answer Option 4: To capture information from the DOM before it changes

Correct Response: 4

Explanation: The "getSnapshotBeforeUpdate" lifecycle method in React is used to capture information from the DOM before it changes. This method is called right before the DOM is updated and can return a value that will be passed to the "componentDidUpdate" method. It is commonly used to preserve scroll position or other user interface state during updates.

[Reference: https://reactjs.org/docs/react-component.html#getsnapshotbeforeupdate]

Question: Do Hooks replace render props and higher order components?

Answer Option 1: Yes, Hooks can replace both render props and higher order components

Answer Option 2: No, Hooks cannot replace either render props or higher order components

Answer Option 3: Hooks can only replace render props

Answer Option 4: Hooks can only replace higher order components

Correct Response: 1

Explanation: Hooks in React can replace both render props and higher order components in some cases. Hooks can provide similar functionality to both of these patterns, while also simplifying the code and making it more reusable. For example, the "useEffect" Hook can replace the "componentDidMount" and "componentDidUpdate" lifecycle methods, as well as some uses of higher order components.

[Reference: https://reactjs.org/docs/hooks-faq.html#do-hooks-cover-all-use-cases-for-classes]

Question: What is the recommended way for naming components?

Answer Option 1: Use a simple name that describes the component

Answer Option 2: Use a long name that includes the component's functionality

Answer Option 3: Use a name that is the same as the component's parent folder

Answer Option 4: Use a name that is the same as the component's file name

Correct Response: 1

Explanation: The recommended way for naming components in React is to use a simple name that describes the component. The name should be a noun or noun phrase that accurately represents the component's purpose. This makes it easier to understand and maintain the code, as well as to reuse the component in other parts of the application.

[Reference: https://reactjs.org/docs/jsx-in-depth.html#choosing-the-type-at-runtime]

Question: What is the recommended ordering of methods in component class?

Answer Option 1: Constructor, render, lifecycle methods, event handlers

Answer Option 2: Lifecycle methods, constructor, render, event handlers

Answer Option 3: Constructor, event handlers, render, lifecycle methods

Answer Option 4: Render, constructor, lifecycle methods, event handlers

Correct Response: 1

Explanation: The recommended ordering of methods in a React component class is as follows: constructor, render, lifecycle methods, event handlers. This ordering groups related methods together and makes it easier to understand and maintain the code.

[Reference: https://reactjs.org/docs/react-component.html#render]

Question: What is a switching component?

Answer Option 1: A component that changes its state based on user input

Answer Option 2: A component that switches between multiple child components

Answer Option 3: A component that provides an interface for switching between different pages

Answer Option 4: A component that animates between different states

Correct Response: 2

Explanation: A switching component in React is a component that renders one of several child components based on some condition or state. This is commonly used to switch between different views or user interface elements in response to user input or other events.

[Reference: https://reactjs.org/docs/conditional-rendering.html#inline-if-with-logical--operator]

Question: Why we need to pass a function to setState()?

Answer Option 1: To avoid infinite loops

Answer Option 2: To ensure proper synchronization of state updates

Answer Option 3: To handle errors gracefully

Answer Option 4: To improve performance

Correct Response: 1

Explanation: In React, we need to pass a function to the "setState" method in order to avoid infinite loops. If we pass an object directly, it can cause the component to re-render and update the state endlessly. By passing a function, we can ensure that the state is updated correctly and avoid these issues.

[Reference: https://reactjs.org/docs/react-component.html#setstate]

Question: What is strict mode in React?

Answer Option 1: A mode that improves performance by reducing unnecessary updates

Answer Option 2: A mode that enables the use of experimental features

Answer Option 3: A mode that highlights potential problems in the code

Answer Option 4: A mode that disables certain security features

Correct Response: 3

Explanation: Strict mode in React is a mode that highlights potential problems in the code, such as deprecated lifecycle methods or unsafe practices. It can help identify issues early on and improve the overall quality of the code. Strict mode can be enabled globally or for individual components.

[Reference: https://reactjs.org/docs/strict-mode.html]

Question: What are React Mixins?

Answer Option 1: Reusable code snippets that can be applied to multiple components

Answer Option 2: Components that are composed of other components

Answer Option 3: Techniques for improving the performance of React applications

Answer Option 4: Methods for handling async operations in React

Correct Response: 1

Explanation: React Mixins are reusable code snippets that can be applied to multiple components in order to provide shared functionality. They are a way to encapsulate logic and behavior that can be used across multiple components, allowing for code reuse and reducing duplication. However, they are not recommended in modern versions of React and have been largely replaced by higher-order components and render props.

[Reference: https://reactjs.org/docs/mixins.html]

Question: Why is isMounted() an anti-pattern and what is the proper solution?

Answer Option 1: It causes performance issues

Answer Option 2: It is not supported in React v16

Answer Option 3: It can lead to race conditions and bugs

Answer Option 4: It is not needed in modern versions of React

Correct Response: 3

Explanation: The "isMounted" method in React is considered an anti-pattern because it can lead to race conditions and bugs. This method checks if the component is mounted to the DOM, but it can return a false positive if it is called during the unmounting phase. The proper solution is to use the "componentDidMount" and "componentWillUnmount" lifecycle methods to manage component state and cleanup.

[Reference: https://reactjs.org/blog/2015/12/16/ismounted-antipattern.html]

Question: What are the Pointer Events supported in React?

Answer Option 1: Mouse events only

Answer Option 2: Touch events only

Answer Option 3: Both mouse and touch events

Answer Option 4: None of the above

Correct Response: 3

Explanation: React supports both mouse and touch events through the use of Pointer Events. Pointer Events are a standardized event model that provide a unified way to handle mouse, touch, and stylus input. React provides a set of event handlers for Pointer Events, such as "onPointerDown" and "onPointerMove", that can be used to create responsive and touch-friendly user interfaces.

[Reference: https://reactjs.org/docs/events.html#pointer-events]

Question: Why should component names start with capital letter?

Answer Option 1: It is a convention that makes the code easier to read

Answer Option 2: It is a requirement of the React compiler

Answer Option 3: It ensures proper scoping of the component

Answer Option 4: It improves the performance of the component

Correct Response: 1

Explanation: In React, component names should start with a capital letter in order to distinguish them from regular HTML tags and make the code easier to read. This is a convention that is widely used in the React community and helps developers understand the purpose and scope of different parts of the code.

[Reference: https://reactjs.org/docs/jsx-in-depth.html#user-defined-components-must-be-capitalized]

Question: How to loop inside JSX?

Answer Option 1: Using a for loop

Answer Option 2: Using a while loop

Answer Option 3: Using the map() method

Answer Option 4: Using the forEach() method

Correct Response: 3

Explanation: To loop inside JSX in React, you can use the map() method to map an array of data to a set of React elements. This allows you to dynamically generate UI elements based on data, such as a list of items or a set of images.

[Reference: https://reactjs.org/docs/lists-and-keys.html#rendering-multiple-components]

Question: How do you access props in attribute quotes?

Answer Option 1: {this.props}

Answer Option 2: {props}

Answer Option 3: {this.props.someProp}

Answer Option 4: {someProp}

Correct Response: 3

Explanation: In React, you can access props in attribute quotes by using the "this.props" syntax and the name of the prop. For example, to access a prop named "someProp", you would use the syntax "{this.props.someProp}" inside the attribute quotes. This allows you to dynamically set attributes based on props, such as setting the value of an input field or the source of an image.

[Reference: https://reactjs.org/docs/introducing-jsx.html#embedding-expressions-in-jsx]

Question: What is the difference between constructor and getInitialState?

Answer Option 1: There is no difference, they both initialize component state

Answer Option 2: constructor is used in ES6 classes, while getInitialState is used in ES5 classes

Answer Option 3: constructor is called before getInitialState

Answer Option 4: getInitialState is only used in functional components

Correct Response: 2

Explanation: The "constructor" method and "getInitialState" method are both used to initialize the state of a component in React, but they are used in different class styles. The "constructor" method is used in ES6 classes, while "getInitialState" is used in ES5 classes. In general, it is recommended to use the "constructor" method in modern React code.

[Reference: https://reactjs.org/docs/react-component.html#constructor]

Question: What is the difference between super() and super(props) in React using ES6 classes?

Answer Option 1: There is no difference, they both call the superclass constructor

Answer Option 2: super() calls the superclass constructor, while super(props) passes props to the constructor

Answer Option 3: super(props) calls the superclass constructor, while super() passes props to the constructor

Answer Option 4: super() is only used in functional components

Correct Response: 2

Explanation: In React using ES6 classes, "super()" is used to call the constructor of the superclass, while "super(props)" is used to pass props to the constructor of the superclass. The "super(props)" syntax is necessary if you need to access props in the constructor of a subclass.

[Reference: https://reactjs.org/docs/react-component.html#constructor]

Question: What is React PropType array with shape?

Answer Option 1: A way to validate an array of values with a specific shape

Answer Option 2: A way to validate an object with a specific shape

Answer Option 3: A way to validate an array of objects with a specific shape

Answer Option 4: A way to validate a string with a specific shape

Correct Response: 3

Explanation: The "shape" propType in React allows you to validate an object with a specific shape, while the "arrayOf" propType allows you to validate an array of values. The "arrayOf" propType can be combined with the "shape" propType to validate an array of objects with a specific shape. This is useful for validating data structures in React components.

[Reference: https://reactjs.org/docs/typechecking-with-proptypes.html#proptypes]

Question: How to conditionally apply class attributes?

Answer Option 1: Using the "class" attribute and a ternary operator

Answer Option 2: Using the "className" attribute and a ternary operator

Answer Option 3: Using the "class" attribute and an if statement

Answer Option 4: Using the "className" attribute and an if statement

Correct Response: 2

Explanation: In React, you can conditionally apply class attributes by using the "className" attribute and a ternary operator. This allows you to add or remove classes based on some condition, such as the state of the component. The "className" attribute is used instead of the "class" attribute in React, because "class" is a reserved keyword in JavaScript.

[Reference: https://reactjs.org/docs/conditional-rendering.html#rendering-a-component]

Question: What is the difference between React and ReactDOM?

Answer Option 1: React is for creating components, while ReactDOM is for rendering components to the DOM

Answer Option 2: React is for handling state and events, while ReactDOM is for handling rendering and updating

Answer Option 3: There is no difference, they are the same thing

Answer Option 4: React is a server-side rendering library, while ReactDOM is a client-side rendering library

Correct Response: 1

Explanation: React is a library for creating components and managing the state and events of those components, while ReactDOM is a library for rendering those components to the DOM. React provides the programming interface for working with components, while ReactDOM provides the methods for rendering and updating the components in the browser.

[Reference: https://reactjs.org/docs/react-dom.html]

Question: Why ReactDOM is separated from React?

Answer Option 1: To improve performance and reduce bundle size

Answer Option 2: To support server-side rendering in Node.js

Answer Option 3: To provide a more modular architecture for React

Answer Option 4: To support different rendering targets, such as mobile devices or game engines

Correct Response: 1

Explanation: ReactDOM is separated from React in order to improve performance and reduce the bundle size of React applications. Separating the rendering logic from the component logic allows for more efficient updates and reduces the amount of JavaScript that needs to be downloaded by the client. This separation also allows for easier integration with other rendering targets, such as native mobile apps or desktop applications.

[Reference: https://reactjs.org/docs/react-dom.html#why-separate]

Question: How to use React label element?

Answer Option 1: <label>My Label</label>

Answer Option 2: <Label>My Label</Label>

Answer Option 3: <label for="my-input">My Label</label>

Answer Option 4: <Label for="my-input">My Label</Label>

Correct Response: 3

Explanation: In React, you can use the standard HTML "label" element to associate a label with an input field or other form element. To do this, you can use the "for" attribute on the label element, and set it to the "id" of the input field. In JSX, you can use the "htmlFor" attribute instead of "for", because "for" is a reserved keyword in JavaScript.

[Reference: https://reactjs.org/docs/forms.html#the-label-element]

Question: How to combine multiple inline style objects?

Answer Option 1: Using the Object.assign() method

Answer Option 2: Using the spread operator

Answer Option 3: Using the Array.concat() method

Answer Option 4: Using the Object.merge() method

Correct Response: 2

Explanation: In React, you can combine multiple inline style objects by using the spread operator. This allows you to merge the properties of multiple style objects into a single object that can be applied to a component. For example: <div style={{...style1, ...style2}}>My Component . This will create a new style object that contains the properties of both style1 and style2.

[Reference: https://reactjs.org/docs/dom-elements.html#style]

Question: How to re-render the view when the browser is resized?

Answer Option 1: Use a window event listener and call forceUpdate()

Answer Option 2: Use a state variable and update it on resize

Answer Option 3: Use a ref and measure the size of the container element

Answer Option 4: Use a media query and conditional rendering

Correct Response: 2

Explanation: In React, you can re-render the view when the browser is resized by using a state variable and updating it on resize. This will trigger a re-render of the component, and allow you to update the layout or other properties based on the new size of the browser window. You can use the "window.addEventListener" method to listen for the "resize" event, and update the state variable accordingly.

[Reference: https://stackoverflow.com/questions/19014250/rerender-view-on-browser-resize-with-react]

Question: What is the difference between setState and replaceState methods?

Answer Option 1: There is no difference, they both update component state

Answer Option 2: setState merges the new state with the old state, while replaceState overwrites the old state

Answer Option 3: replaceState is deprecated in React v16, while setState is still supported

Answer Option 4: setState is used in class components, while replaceState is used in functional components

Correct Response: 2

Explanation: In React, the "setState" method is used to update component state, and it merges the new state with the old state. The "replaceState" method is similar, but it overwrites the old state completely with the new state. However, "replaceState" is deprecated in React and should not be used. Instead, you should use the "setState" method to update state in a React component.

[Reference: https://reactjs.org/docs/react-component.html#replacestate]

Question: How to listen to state changes?

Answer Option 1: Use a setInterval() function

Answer Option 2: Use a window event listener

Answer Option 3: Use the componentDidUpdate() lifecycle method

Answer Option 4: Use a ref and measure the size of the container element

Correct Response: 3

Explanation: In React, you can listen to state changes by using the "componentDidUpdate()" lifecycle method. This method is called after a component's state has been updated, and you can use it to perform additional actions or update the UI based on the new state. For example, you can use this method to update the DOM, call a web API, or trigger an animation.

[Reference: https://reactjs.org/docs/react-component.html#componentdidupdate]

Question: What is the recommended approach of removing an array element in react state?

Answer Option 1: Use the splice() method

Answer Option 2: Use the slice() method

Answer Option 3: Use the filter() method

Answer Option 4: Use the map() method

Correct Response: 3

Explanation: In React, the recommended approach for removing an element from an array in state is to use the "filter()" method. This allows you to create a new array that contains all the elements of the original array except the one you want to remove. This is a safer and more efficient approach than using the "splice()" method, which mutates the original array and can cause unexpected side effects.

[Reference: https://reactjs.org/docs/update.html#updating-state-arrays]

Question: Is it possible to use React without rendering HTML?

Answer Option 1: No, React is designed for rendering HTML

Answer Option 2: Yes, React can be used for non-HTML rendering, such as SVG or canvas

Answer Option 3: Yes, React can be used for server-side rendering only

Answer Option 4: Yes, React can be used for building mobile apps with React Native

Correct Response: 2

Explanation: In React, it is possible to use the library for non-HTML rendering, such as SVG or canvas. React provides a flexible programming interface for creating components and managing state, and it can be used to generate any kind of output, not just HTML. React is also used for building mobile apps with React Native, which uses the same programming model as React for building UIs on iOS and Android platforms.

[Reference: https://reactjs.org/docs/react-dom.html#what-is-react-dom]

Question: How to pretty print JSON with React?

Answer Option 1: Use the JSON.stringify() method

Answer Option 2: Use the JSON.parse() method

Answer Option 3: Use the JSON.prettify() method

Answer Option 4: Use the JSON.format() method

Correct Response: 1

Explanation: In React, you can pretty print JSON data by using the "JSON.stringify()" method with a "null" value for the "replacer" parameter and a number value for the "space" parameter. This will format the JSON data with indentation and line breaks for readability. For example: JSON.stringify(myData, null, 2).

[Reference: https://stackoverflow.com/questions/4810841/how-can-i-pretty-print-json-using-javascript]

Question: Why you can't update props in React?

Answer Option 1: Props are read-only and should not be modified directly

Answer Option 2: Props are immutable and cannot be changed

Answer Option 3: React does not provide a method for updating props

Answer Option 4: Updating props can cause performance issues in React

Correct Response: 2

Explanation: In React, props are immutable and cannot be changed directly. This is because React is designed to be a one-way data flow, where data is passed down from parent components to child components through props. If you need to update the data, you should do so in the parent component and pass the updated data down as props to the child components.

[Reference: https://reactjs.org/docs/components-and-props.html#props-are-read-only]

Question: How to focus an input element on page load?

Answer Option 1: Use the autoFocus attribute on the input element

Answer Option 2: Use the focus() method in componentDidMount()

Answer Option 3: Use the onFocus() event in the render method

Answer Option 4: Use the onLoad() event on the body element

Correct Response: 2

Explanation: In React, you can focus an input element on page load by using the "focus()" method in the "componentDidMount()" lifecycle method. This will set the focus to the input element after the component has been mounted in the DOM. For example: componentDidMount() { this.myInput.focus(); }.

[Reference: https://stackoverflow.com/questions/28889826/how-to-set-focus-on-input-field]

Question: How can we find the version of React at runtime in the browser?

Answer Option 1: Use the "npm list react" command in the console

Answer Option 2: Use the "React.version" property in the console

Answer Option 3: Use the "document.head" property in the console

Answer Option 4: Use the "window.React.version" property in the console

Correct Response: 2

Explanation: In React, you can find the version of React at runtime in the browser by using the "React.version" property in the console. This will display the current version of React that is being used on the page.

[Reference: https://reactjs.org/docs/faq-versioning.html#how-do-i-check-the-react-version-at-runtime]

Question: What are the approaches to include polyfills in your create-react-app?

Answer Option 1: Using the "core-js" library

Answer Option 2: Using the "@babel/preset-env" package

Answer Option 3: Using the "babel-polyfill" package

Answer Option 4: Using the "react-app-polyfill" package

Correct Response: 1, 2, 4

Explanation: In Create React App, you can include polyfills to support older browsers by using the "core-js" library, the "@babel/preset-env" package, or the "react-app-polyfill" package. These packages provide polyfills for various ES6+ features and browser APIs, and can be configured in the "babel.config.js" or "webpack.config.js" files.

[Reference: https://create-react-app.dev/docs/supported-browsers-features/]

Question: How to use https instead of http in create-react-app?

Answer Option 1: Use the HTTPS=true environment variable

Answer Option 2: Use the HTTP=false environment variable

Answer Option 3: Use the SSL=true environment variable

Answer Option 4: Use the SSL_CERTIFICATE=true environment variable

Correct Response: 1

Explanation: In Create React App, you can use HTTPS instead of HTTP by setting the "HTTPS=true" environment variable. This will start the development server with HTTPS enabled, and will allow you to test your application with SSL/TLS encryption.

[Reference: https://create-react-app.dev/docs/using-https-in-development/]

Question: How to avoid using relative path imports in create-react-app?

Answer Option 1: Use absolute path imports with a "jsconfig.json" file

Answer Option 2: Use webpack aliases with a "webpack.config.js" file

Answer Option 3: Use environment variables with a ".env" file

Answer Option 4: Use npm modules with the "dependencies" field in package.json

Correct Response: 1

Explanation: In Create React App, you can avoid using relative path imports by using absolute path imports with a "jsconfig.json" file. This will allow you to import modules from any directory in your project using a relative path, such as "src/components/MyComponent".

[Reference: https://create-react-app.dev/docs/importing-a-component/#absolute-imports]

Question: How to add Google Analytics for react-router?

Answer Option 1: Use the "react-ga" library with the "react-router-ga" package

Answer Option 2: Use the "google-analytics" library with the "react-router-analytics" package

Answer Option 3: Use the "react-router-ga" library with the "google-analytics-react" package

Answer Option 4: Use the "react-analytics" library with the "router-react-analytics" package

Correct Response: 1

Explanation: In React, you can add Google Analytics for React Router by using the "react-ga" library with the "react-router-ga" package. This will allow you to track pageviews and events for your React Router pages in Google Analytics, and can be configured in the "index.js" or "App.js" files.

[Reference: https://github.com/react-ga/react-ga/wiki/React-Router-v4-withTracker]

Question: How to update a component every second?

Answer Option 1: Use the setInterval() function in the constructor method

Answer Option 2: Use the setInterval() function in the componentDidMount() method

Answer Option 3: Use the setTimeout() function in the componentDidUpdate() method

Answer Option 4: Use the setTimeout() function in the render() method

Correct Response: 2

Explanation: In React, you can update a component every second by using the "setInterval()" function in the "componentDidMount()" lifecycle method. This will create a timer that updates the component state every second, causing the component to re-render with the new state values. For example: componentDidMount() { this.timerID = setInterval(() => this.tick(), 1000); }.

[Reference: https://reactjs.org/docs/state-and-lifecycle.html#adding-lifecycle-methods-to-a-class]

Question: How do you apply vendor prefixes to inline styles in React?

Answer Option 1: Use the "autoprefixer" library with CSS-in-JS

Answer Option 2: Use the "postcss" library with a custom plugin

Answer Option 3: Use the "react-prefixer" library with inline styles

Answer Option 4: Use the "react-style-prefixer" library with inline styles

Correct Response: 4

Explanation: In React, you can apply vendor prefixes to inline styles by using the "react-style-prefixer" library. This library automatically adds vendor prefixes to CSS properties based on the browser's user agent, and can be used with inline styles or CSS-in-JS solutions like styled-components or emotion.

[Reference: https://github.com/rofrischmann/react-style-prefixer]

Question: How to import and export components using react and ES6?

Answer Option 1: Use the "require" keyword to import components

Answer Option 2: Use the "export" keyword to export components

Answer Option 3: Use the "import" keyword to import components

Answer Option 4: Use the "module.exports" object to export components

Correct Response: 3

Explanation: In React and ES6, you can import and export components using the "import" and "export" keywords. To import a component, you can use the "import" keyword followed by the component name and file path. To export a component, you can use the "export" keyword before the component declaration. For example: import React from 'react'; export class MyComponent extends React.Component { ... }.

[Reference: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/import]

Question: What are the exceptions on React component naming?

Answer Option 1: Components must start with a capital letter

Answer Option 2: Components must be named with camelCase convention

Answer Option 3: Components can include numbers, but not as the first character

Answer Option 4: Components can include special characters, such as underscores or hyphens

Correct Response: 1, 2, 3

Explanation: In React, there are several rules for naming components, including that components must start with a capital letter, use camelCase convention, and can include numbers, but not as the first character. Components should also have a descriptive name that reflects their purpose or function in the application.

[Reference: https://reactjs.org/docs/jsx-in-depth.html#user-defined-components-must-be-capitalized]

Question: Why is a component constructor called only once?

Answer Option 1: The constructor is only called when the component is first created

Answer Option 2: The constructor is called for each new instance of the component

Answer Option 3: The constructor is not needed in functional components

Answer Option 4: The constructor is not used in React components

Correct Response: 1

Explanation: In React, the constructor is called only once when the component is first created. This is because the constructor is used to initialize the component state and bind event handlers, and these operations only need to be performed once. After the component has been created, subsequent updates will use the "componentDidUpdate()" method to update the state and re-render the component.

[Reference: https://reactjs.org/docs/react-component.html#constructor]

Question: How to define constants in React?

Answer Option 1: Use the "var" keyword to define constants

Answer Option 2: Use the "let" keyword to define constants

Answer Option 3: Use the "const" keyword to define constants

Answer Option 4: Use the "define" method to define constants

Correct Response: 3

Explanation: In React, you can define constants using the "const" keyword. Constants are variables that cannot be reassigned or redeclared, and are used to store values that will not change throughout the lifetime of the component. For example: const API_KEY = '12345';.

[Reference: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/const]

Question: How to programmatically trigger click event in React?

Answer Option 1: Use the "trigger" method in the "react-click-events" library

Answer Option 2: Use the "simulate" method in the "react-test-renderer" package

Answer Option 3: Use the "onClick" attribute with a function call

Answer Option 4: Use the "document.dispatchEvent" method with a custom event

Correct Response: 2

Explanation: In React, you can programmatically trigger a click event by using the "simulate()" method in the "react-test-renderer" package. This method allows you to simulate a click event on a React component, and can be useful for testing or automation purposes. For example: TestRenderer.act(() => { buttonInstance.simulate('click'); });.

[Reference: https://reactjs.org/docs/test-renderer.html#testinstancesimulateeventname-args]

Question: Is it possible to use async/await in plain React?

Answer Option 1: Yes, async/await can be used with plain React components

Answer Option 2: No, async/await requires a library or framework like Redux or Apollo

Answer Option 3: Yes, async/await can be used with the "react-async" library

Answer Option 4: No, async/await is not supported in plain JavaScript

Correct Response: 1

Explanation: In plain React, you can use async/await to handle asynchronous operations like API calls or Promises. Async/await is a feature of ECMAScript 2017, and can be used with modern browsers or transpiled code. However, you may need to use a library or framework like Redux or Apollo to manage the async state and data flow in your application.

[Reference: https://javascript.info/async-await]

Question: What are the common folder structures for React?

Answer Option 1: Separate folders for components, containers, and utilities

Answer Option 2: Separate folders for actions, reducers, and selectors

Answer Option 3: Separate folders for assets, styles, and tests

Answer Option 4: Separate folders for models, services, and views

Correct Response: 1, 3

Explanation: Common folder structures for React projects include separate folders for components, containers, and utilities, as well as assets, styles, and tests. This structure helps organize the code and make it more maintainable, and can be customized based on the specific needs of the project.

[Reference: https://www.robinwieruch.de/react-folder-structure]

Question: What are the popular packages for animation?

Answer Option 1: React Motion

Answer Option 2: React Pose

Answer Option 3: React Transition Group

Answer Option 4: React Popper

Correct Response: 1, 2, 3

Explanation: Popular packages for animation in React include React Motion, React Pose, and React Transition Group. These packages provide easy-to-use APIs for creating complex animations and transitions, and can be customized to fit the specific needs of the project.

[Reference: https://blog.bitsrc.io/15-react-animation-libraries-for-2021-63a147f88c38]

Question: What is the benefit of styles modules?

Answer Option 1: Styles modules allow you to use CSS with React components

Answer Option 2: Styles modules prevent CSS class naming collisions

Answer Option 3: Styles modules allow for dynamic styling with props

Answer Option 4: Styles modules are required for server-side rendering

Correct Response: 2

Explanation: Styles modules in React allow you to avoid CSS class naming collisions by generating unique class names at runtime. This helps prevent conflicts with other styles in the project, and allows you to use descriptive class names without worrying about naming conventions.

[Reference: https://github.com/css-modules/css-modules]

Question: What are the popular React-specific linters?

Answer Option 1: ESLint

Answer Option 2: Prettier

Answer Option 3: Stylelint

Answer Option 4: React ESLint

Correct Response: 1, 3, 4

Explanation: Popular linters for React include ESLint with React-specific rules, React ESLint, and Stylelint. These linters help enforce coding standards and best practices, and can be integrated into the development workflow for automatic linting and code formatting.

[Reference: https://blog.bitsrc.io/10-tools-every-reactjs-developer-should-have-in-2021-a6c0714a7b76]

Question: How to make AJAX call and In which component lifecycle methods should I make an AJAX call?

Answer Option 1: Use the "fetch" API in the "componentDidMount()" method

Answer Option 2: Use the "XMLHttpRequest" object in the "componentDidUpdate()" method

Answer Option 3: Use the "axios" library in the "componentWillMount()" method

Answer Option 4: Use the "jQuery.ajax" function in the "componentWillReceiveProps()" method

Correct Response: 1

Explanation: In React, you can make an AJAX call by using the "fetch" API or a library like Axios or jQuery. The recommended lifecycle method to make an AJAX call is "componentDidMount()", which is called once the component has been mounted to the DOM. This method is a good place to fetch data from an API or server, and update the component state with the new data.

[Reference: https://reactjs.org/docs/state-and-lifecycle.html#adding-lifecycle-methods-to-a-class]

Question: What are render props?

Answer Option 1: A technique for passing functions as props to child components

Answer Option 2: A technique for rendering components as props to child components

Answer Option 3: A technique for passing state as props to child components

Answer Option 4: A technique for rendering state as props to child components

Correct Response: 1

Explanation: Render props is a technique in React where a component exposes a function prop that returns another component or element. This allows the child component to use the parent's logic or state, and can be used to create reusable, composable components. For example: <MyComponent render={(data) => <ChildComponent data={data} />} />.

[Reference: https://reactjs.org/docs/render-props.html]

Question: What is React Router?

Answer Option 1: A library for managing state in React applications

Answer Option 2: A library for creating animations and transitions in React

Answer Option 3: A library for handling server-side rendering in React

Answer Option 4: A library for routing and navigation in React

Correct Response: 4

Explanation: React Router is a library for routing and navigation in React applications. It provides a declarative API for creating and managing routes, as well as handling dynamic routing and nested routes. React Router can be used with various rendering techniques like server-side rendering, and is widely used in modern React applications.

[Reference: https://reactrouter.com/web/guides/quick-start]

Question: How React Router is different from history library?

Answer Option 1: React Router is a wrapper around the history library

Answer Option 2: History library is a wrapper around the React Router

Answer Option 3: React Router is used for client-side routing and navigation

Answer Option 4: History library is used for server-side routing and navigation

Correct Response: 1

Explanation: React Router is a higher-level abstraction on top of the history library. It provides a declarative API for routing and navigation in React applications, and manages the browser history and URL updates. The history library, on the other hand, is a low-level library that provides a simple interface for manipulating the browser history, and can be used directly in non-React applications.

[Reference: https://github.com/ReactTraining/history]

Question: What is the purpose of push and replace methods of history?

Answer Option 1: To add or replace a new route to the history stack

Answer Option 2: To clear the history stack and start a new session

Answer Option 3: To navigate to the previous route in the history stack

Answer Option 4: To update the current route without adding a new entry to the history stack

Correct Response: 1

Explanation: The push and replace methods of the history object are used to add or replace a new route to the history stack. The push method adds a new entry to the history stack and navigates to the specified route, while the replace method updates the current entry in the history stack without adding a new one. These methods are commonly used for programmatic navigation and managing the browser history in React applications.

[Reference: https://reactrouter.com/web/api/history]

Question: How do you programmatically navigate using React router v4?

Answer Option 1: Use the "window.location" object with a URL string

Answer Option 2: Use the "this.props.history.push" method with a route path

Answer Option 3: Use the "document.navigate" function with a URL string

Answer Option 4: Use the "this.props.navigate" method with a route path

Correct Response: 2

Explanation: In React Router v4, you can programmatically navigate by using the "history" object provided by the router. To navigate to a new route, you can use the "this.props.history.push" method with a route path as a string. This method adds a new entry to the history stack and navigates to the specified route. For example: this.props.history.push('/new-route');.

[Reference: https://reactrouter.com/web/api/history]

Question: How to get query parameters in React Router v4

Answer Option 1: Use the "this.props.location.query" object

Answer Option 2: Use the "this.props.location.search" property

Answer Option 3: Use the "this.props.params" object

Answer Option 4: Use the "this.props.query" object

Correct Response: 2

Explanation: In React Router v4, you can get query parameters by using the "location.search" property of the current location object. This property contains the query string of the URL, including the "?" character and the parameter keys and values. To parse the query string, you can use a library like "query-string" or "URLSearchParams". For example: const queryParams = new URLSearchParams(this.props.location.search); const foo = queryParams.get('foo');.

[Reference: https://reactrouter.com/web/example/query-parameters]

Question: Why you get "Router may have only one child element" warning?

Answer Option 1: Because you have more than one <Router> component

Answer Option 2: Because you have more than one child component inside the <Router>

Answer Option 3: Because you have nested <Route> components without a parent <Switch>

Answer Option 4: Because you have multiple routes with the same path

Correct Response: 3

Explanation: In React Router v4, you can get the "Router may have only one child element" warning if you have nested <Route> components without a parent <Switch>. This is because the <Switch> component renders the first matching child <Route> and prevents multiple routes from rendering at the same time. To fix the warning, wrap your <Route> components inside a <Switch> component.

[Reference: https://reactrouter.com/web/api/Switch]

Question: How to pass params to history.push method in React Router v4?

Answer Option 1: Pass an object with a "params" property

Answer Option 2: Pass an object with a "query" property

Answer Option 3: Pass a string with the route path and query parameters

Answer Option 4: Pass an object with a "state" property

Correct Response: 4

Explanation: In React Router v4, you can pass params to the history.push method by using the "state" property of the location object. This property can contain any data that you want to pass along with the route, such as query parameters, form data, or session information. For example: this.props.history.push({ pathname: '/new-route', state: { foo: 'bar' } });.

[Reference: https://reactrouter.com/web/api/history]

Question: How to implement default or NotFound page?

Answer Option 1: Use the <Route path="*" component={NotFound} /> syntax

Answer Option 2: Use the <Route component={NotFound} /> syntax

Answer Option 3: Use the <Redirect to="/not-found" /> component

Answer Option 4: Use the <Switch><Route path="*" component={NotFound} /></Switch> syntax

Correct Response: 4

Explanation: In React Router v4, you can implement a default or NotFound page by using a <Switch> component to render the first matching child <Route>, and adding a <Route path="*" component={NotFound} /> as the last child. This route will match any path that has not been matched by other routes, and render the NotFound component.

[Reference: https://reactrouter.com/web/example/no-match]

Question: How to get history on React Router v4?

Answer Option 1: Use the "this.history" object provided by the router

Answer Option 2: Use the "history" object provided by the "react-router-dom" module

Answer Option 3: Use the "history" object provided by the "history" module

Answer Option 4: Use the "this.props.history" object passed to the component

Correct Response: 4

Explanation: In React Router v4, you can get the history object by using the "this.props.history" object passed to the component. This object contains the current history of the router, including the current location, previous locations, and navigation history. You can use this object to programmatically navigate, access the current URL, or manage the browser history.

[Reference: https://reactrouter.com/web/api/history]

Question: How to perform automatic redirect after login?

Answer Option 1: Use the "window.location" object with the new URL

Answer Option 2: Use the "history.push" method with the new route

Answer Option 3: Use the "Redirect" component from React Router

Answer Option 4: Use the "Link" component from React Router

Correct Response: 2

Explanation: In React, you can perform an automatic redirect after login by using the "history.push" method provided by the router. This method adds a new entry to the history stack and navigates to the specified route. You can use this method after successful authentication to redirect the user to a different page, such as a dashboard or a profile page. For example: history.push('/dashboard');.

[Reference: https://reactrouter.com/web/api/history]

Question: What is React-Intl?

Answer Option 1: A library for internationalization in React

Answer Option 2: A library for managing state in React

Answer Option 3: A library for testing React components

Answer Option 4: A library for styling React components

Correct Response: 1

Explanation: React-Intl is a library for internationalization (i18n) in React applications. It provides a set of components, utilities, and formatting functions to handle different languages, locales, and cultural conventions. With React-Intl, you can easily translate text, format dates and numbers, and manage pluralization and gender.

[Reference: https://formatjs.io/docs/react-intl/]

Question: What are the two ways of formatting in React Intl?

Answer Option 1: Using the <FormattedMessage> component

Answer Option 2: Using the "formatMessage" function

Answer Option 3: Using the "formatNumber" function

Answer Option 4: Using the "formatDate" function

Correct Response: 2, 4

Explanation: React Intl provides two ways of formatting text and data, which are using the "formatMessage" function and the "formatDate" and "formatNumber" functions. The "formatMessage" function is used for text translation and interpolation, and allows you to define message templates with placeholders for dynamic data. The "formatDate" and "formatNumber" functions are used for formatting dates and numbers, respectively, and accept options and formats for customization.

[Reference: https://formatjs.io/docs/react-intl/]

Question: How to use FormattedMessage as placeholder using React Intl?

Answer Option 1: <FormattedMessage>{placeholder}
</FormattedMessage>

Answer Option 2: <FormattedMessage values={{placeholder}} />

Answer Option 3: <FormattedMessage placeholder={placeholder} />

Answer Option 4: <FormattedMessage><placeholder />
</FormattedMessage>

Correct Response: 2

Explanation: In React Intl, you can use the <FormattedMessage> component to format text and data with message templates and placeholders. To use a <FormattedMessage> as a placeholder, you can pass an object with the placeholder as a value to the "values" prop of the component. For example: <FormattedMessage id="myMessage" values={{ placeholder: <FormattedMessage id="myPlaceholder" /> }} />. This will replace the placeholder in the "myMessage" message with the formatted text from the "myPlaceholder" message.

[Reference: https://formatjs.io/docs/react-intl/components#formattedmessage]

Question: How to access current locale with React Intl

Answer Option 1: Use the "formatMessage" function with the "locale" option

Answer Option 2: Use the "intl" object provided by the "injectIntl" higher-order component

Answer Option 3: Use the "Intl" object provided by the browser

Answer Option 4: Use the "navigator.language" property

Correct Response: 2

Explanation: In React Intl, you can access the current locale by using the "intl" object provided by the "injectIntl" higher-order component. This object contains information about the current language, locale, and formatting options, and can be used to format text and data in the correct format. For example: const { locale } = this.props.intl;.

[Reference: https://formatjs.io/docs/react-intl/components#injectintl-higher-order-component]

Question: How to format date using React Intl?

Answer Option 1: Use the <FormattedDate> component

Answer Option 2: Use the "formatMessage" function with the "date" option

Answer Option 3: Use the "formatDate" function

Answer Option 4: Use the "Intl.DateTimeFormat" object

Correct Response: 1, 3, 4

Explanation: In React Intl, you can format dates using the <FormattedDate> component, the "formatDate" function, or the "Intl.DateTimeFormat" object. The <FormattedDate> component accepts a "value" prop with a date object, and optional "format" and "timeZone" props for customization. The "formatDate" function and the "Intl.DateTimeFormat" object accept a date object and optional options for formatting, such as "year", "month", "day", "hour", "minute", and "second".

[Reference: https://formatjs.io/docs/react-intl/components#formatteddate]

Question: What is Shallow Renderer in React testing?

Answer Option 1: A testing framework for React

Answer Option 2: A tool for simulating component rendering without deep rendering

Answer Option 3: A package for generating test data

Answer Option 4: A package for mocking HTTP requests

Correct Response: 2

Explanation: Shallow Renderer is a tool for simulating component rendering without deep rendering. It can be used for testing components in isolation and allows developers to easily check the output of a component's render method.

[Reference: https://reactjs.org/docs/shallow-renderer.html]

Question: What is TestRenderer package in React?

Answer Option 1: A package for mocking HTTP requests

Answer Option 2: A package for generating test data

Answer Option 3: A tool for simulating component rendering without deep rendering

Answer Option 4: A tool for rendering React components to JSON format

Correct Response: 4

Explanation: TestRenderer is a package that allows developers to render React components to a JSON format, making it easier to test and debug components. It is used for snapshot testing and is similar to the ReactTestUtils package.

[Reference: https://reactjs.org/docs/test-renderer.html]

Question: What is the purpose of ReactTestUtils package?

Answer Option 1: To generate test data

Answer Option 2: To mock HTTP requests

Answer Option 3: To simulate component rendering without deep rendering

Answer Option 4: To test React components

Correct Response: 4

Explanation: The ReactTestUtils package provides a set of utilities for testing React components. It allows developers to simulate events, perform component rendering, and find components in the rendered output. It is commonly used in combination with Jest or another testing framework.

[Reference: https://reactjs.org/docs/test-utils.html]

Question: What is Jest?

Answer Option 1: A package for generating test data

Answer Option 2: A tool for simulating component rendering without deep rendering

Answer Option 3: A package for mocking HTTP requests

Answer Option 4: A testing framework for React

Correct Response: 4

Explanation: Jest is a testing framework for JavaScript applications, including React. It is developed by Facebook and is widely used in the React community. Jest includes features such as snapshot testing, mocking, and code coverage analysis. It can be used to test React components and other JavaScript code.

[Reference: https://jestjs.io/docs/en/getting-started]

Question: Give a simple example of Jest test case

Answer Option 1: testing a Redux reducer

Answer Option 2: testing an API call

Answer Option 3: testing a component's render method

Answer Option 4:

Correct Response: 3

Explanation: A simple example of a Jest test case would be testing a component's render method. For example, you could test that a component renders a specific HTML element or that it correctly sets a state variable.

[Reference: https://jestjs.io/docs/en/tutorial-react]

Question: What is Flux?

Answer Option 1: A testing framework for React

Answer Option 2: A database management system

Answer Option 3: A design pattern for managing state in React applications

Answer Option 4: A CSS preprocessor

Correct Response: 3

Explanation: Flux is a design pattern for managing state in React applications. It was developed by Facebook and is often used in combination with React. Flux emphasizes a unidirectional data flow, in which data flows in a single direction through the application. This helps prevent issues with data inconsistency and makes it easier to manage state in large applications.

[Reference: https://facebook.github.io/flux/docs/in-depth-overview]

Question: What is Redux?

Answer Option 1: A design pattern for managing state in React applications

Answer Option 2: A testing framework for React

Answer Option 3: A CSS preprocessor

Answer Option 4: A database management system

Correct Response: 1

Explanation: Redux is a design pattern and library for managing state in JavaScript applications, including React. It emphasizes a single source of truth for application state and a unidirectional data flow. Redux allows developers to manage complex state in large applications, and is often used in combination with React.

[Reference: https://redux.js.org/introduction/getting-started]

Question: What are the downsides of Redux compared to Flux?

Answer Option 1: More complex setup

Answer Option 2: Steep learning curve

Answer Option 3: Boilerplate code

Answer Option 4: More difficult to debug

Correct Response: 1, 2, 3

Explanation: Compared to Flux, Redux can have a more complex setup, a steep learning curve, and more boilerplate code. This can make it more difficult to get started with Redux, especially for smaller applications. However, Redux does offer advantages such as better performance and easier testing.

[Reference: https://blog.logrocket.com/why-you-should-choose-redux-over-flux-for-state-management-in-react-apps/]

Question: What is the difference between mapStateToProps() and mapDispatchToProps()?

Answer Option 1: mapStateToProps() maps state to props, while mapDispatchToProps() maps props to state

Answer Option 2: mapStateToProps() is used for dispatching actions, while mapDispatchToProps() is used for mapping state to props

Answer Option 3: mapStateToProps() maps state to props, while mapDispatchToProps() maps dispatch functions to props

Answer Option 4: There is no difference

Correct Response: 3

Explanation: mapStateToProps() is used to map the application state to the props of a React component, while mapDispatchToProps() is used to map dispatch functions to the props of a React component. The dispatch functions are used to dispatch actions to the Redux store, allowing the component to update the application state.

[Reference: https://react-redux.js.org/using-react-redux/connect-mapdispatch]

Question: Can I dispatch an action in reducer?

Answer Option 1: Yes, it is a common practice

Answer Option 2: No, it violates the principle of unidirectional data flow

Answer Option 3:

Answer Option 4:

Correct Response: 2

Explanation: No, it is not recommended to dispatch actions in a reducer function. This violates the principle of unidirectional data flow in Redux, in which actions flow in a single direction from the view to the reducer. Dispatching an action in a reducer can lead to unexpected behavior and make the application more difficult to reason about.

[Reference: https://redux.js.org/style-guide/style-guide#reducers-must-not-have-side-effects]

Question: How to access Redux store outside a component?

Answer Option 1: Use a global variable

Answer Option 2: Use the useContext hook

Answer Option 3: Use the useSelector hook

Answer Option 4: Use the getState() method

Correct Response: 4

Explanation: To access the Redux store outside of a component, you can use the getState() method provided by the Redux store. This allows you to access the current state of the store and is often used in middleware or in other parts of the application that do not have direct access to the store.

[Reference: https://redux.js.org/api/store#getstate]

Question: What are the drawbacks of MVW pattern?

Answer Option 1: Tightly coupled code

Answer Option 2: Difficulty in testing

Answer Option 3: Lack of separation of concerns

Answer Option 4: Difficulty in scaling

Correct Response: 1, 2, 3

Explanation: MVW (Model-View-Whatever) patterns like MVC, MVP, and MVVM can have drawbacks such as tightly coupled code, difficulty in testing, and lack of separation of concerns. These patterns can also become difficult to scale as the application grows larger.

[Reference: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller#Criticism]

Question: Are there any similarities between Redux and RxJS?

Answer Option 1: Both are used for state management

Answer Option 2: Both are libraries for React

Answer Option 3: Both are based on the observer pattern

Answer Option 4: Both use middleware for side effects

Correct Response: 1, 3

Explanation: Redux and RxJS have some similarities, including that both are used for state management and both are based on the observer pattern. However, they are different libraries, with Redux being a state management library and RxJS being a reactive programming library.

[Reference: https://medium.com/javascript-scene/what-is-reactive-programming-bc9fa142ef2c]

Question: How to dispatch an action on load?

Answer Option 1: Use a lifecycle method in a component

Answer Option 2: Use a middleware

Answer Option 3: Dispatch the action in the reducer

Answer Option 4:

Correct Response: 1

Explanation: To dispatch an action on load, you can use a lifecycle method in a React component, such as componentDidMount(). This method is called once the component has mounted, allowing you to dispatch an action to the Redux store.

[Reference: https://redux.js.org/recipes/componentdidmount]

Question: How to use connect from React Redux?

Answer Option 1: Use it as a HOC to connect a component to the Redux store

Answer Option 2: Use it as a middleware for handling async actions

Answer Option 3: Use it to create a new Redux store

Answer Option 4: Use it to reset the Redux state

Correct Response: 1

Explanation: connect is a Higher Order Component (HOprovided by React Redux that allows you to connect a component to the Redux store. It provides the component with access to the store and allows the component to subscribe to changes in the store's state.

[Reference: https://react-redux.js.org/api/connect]

Question: How to reset state in Redux?

Answer Option 1: Dispatch a RESET action

Answer Option 2: Modify the state directly

Answer Option 3: Use the combineReducers function

Answer Option 4:

Correct Response: 1

Explanation: To reset the state in Redux, you can dispatch a RESET action to the Redux store. This action can be handled by a reducer function that returns the initial state of the application. This allows you to reset the application state to its initial values.

[Reference: https://stackoverflow.com/questions/35622588/how-to-reset-the-state-of-a-redux-store]

Question: What's the purpose of at symbol in the redux connect decorator?

Answer Option 1: It is used to reference a component's props

Answer Option 2: It is used to reference the Redux store's state

Answer Option 3: It is used to reference a component's state

Answer Option 4:

Correct Response: 2

Explanation: The @ symbol in the Redux connect decorator is used to reference the state of the Redux store. This allows the component to access the store's state without having to pass it down as props.

[Reference: https://stackoverflow.com/questions/39178834/what-is-the-symbol-in-react-redux-connect-for]

Question: What is the difference between React context and React Redux?

Answer Option 1: React context is built into React, while React Redux is a separate library

Answer Option 2: React Redux is used for state management, while React context is used for passing data between components

Answer Option 3: React Redux provides a more powerful API for managing state

Answer Option 4: React context is recommended for small applications, while React Redux is recommended for large applications

Correct Response: 1, 2

Explanation: React context and React Redux are both used for passing data between components, but they have some differences. React context is built into React and is recommended for simple use cases, while React Redux is a separate library that provides a more powerful API for managing state and is recommended for larger applications.

[Reference: https://stackoverflow.com/questions/50526678/react-redux-vs-react-context]

Question: Why are Redux state functions called reducers?

Answer Option 1: Because they reduce the state of the application

Answer Option 2: Because they reduce the complexity of the application

Answer Option 3: Because they reduce the size of the application

Answer Option 4: Because they reduce the coupling between components

Correct Response: 1

Explanation: Redux state functions are called reducers because they take the current state of the application and an action as input, and return a new state as output. This process of reducing the state of the application is the core principle of Redux.

[Reference: https://redux.js.org/basics/reducers]

Question: How to make AJAX request in Redux?

Answer Option 1: Use the fetch() method in a component

Answer Option 2: Use a middleware such as Redux Thunk or Redux Saga

Answer Option 3: Use the getState() method to access the Redux store's state

Answer Option 4:

Correct Response: 2

Explanation: To make an AJAX request in Redux, you can use a middleware such as Redux Thunk or Redux Saga. These middleware allow you to handle asynchronous actions, such as making an AJAX request, and dispatching actions based on the response.

[Reference: https://redux.js.org/advanced/async-actions]

Question: Should I keep all component's state in Redux store?

Answer Option 1: Yes, it is recommended to keep all component state in the Redux store

Answer Option 2: No, it is recommended to keep only the necessary state in the Redux store

Answer Option 3:

Answer Option 4:

Correct Response: 2

Explanation: It is not recommended to keep all component state in the Redux store. While Redux is useful for managing state in large applications, it can also add unnecessary complexity and boilerplate code to small applications. It is recommended to keep only the necessary state in the Redux store and use component state for simpler state management needs.

[Reference: https://redux.js.org/faq/general#should-i-only-keep-plain-objects-in-my-store-state]

Question: What is the proper way to access Redux store?

Answer Option 1: Use the getState() method

Answer Option 2: Use the connect() function

Answer Option 3: Use the Provider component

Answer Option 4: Use the useContext() hook

Correct Response: 2

Explanation: The proper way to access the Redux store is to use the connect() function provided by React Redux. This allows you to connect a component to the store and access its state and dispatch functions.

[Reference: https://react-redux.js.org/api/connect]

Question: What is the difference between component and container in React Redux?

Answer Option 1: Components are stateful, while containers are stateless

Answer Option 2: Components are connected to the Redux store, while containers are not

Answer Option 3: Components are responsible for rendering, while containers are responsible for state management

Answer Option 4:

Correct Response: 2

Explanation: Components and containers in React Redux have different responsibilities. Components are responsible for rendering UI elements and receive props from their parent components, while containers are connected to the Redux store and manage the application state.

[Reference: https://redux.js.org/basics/usage-with-react#presentational-and-container-components]

Question: What is the purpose of the constants in Redux?

Answer Option 1: To make the code easier to read and understand

Answer Option 2: To prevent changes to the action types

Answer Option 3: To enable minification and other performance optimizations

Answer Option 4:

Correct Response: 2

Explanation: Constants in Redux are used to define action types, which are used to identify the type of action being dispatched to the Redux store. By using constants, you can prevent unintended changes to the action types and make it easier to understand the purpose of each action.

[Reference: https://redux.js.org/style-guide/style-guide#use-constants-for-action-types]

Question: What are the different ways to write mapDispatchToProps()?

Answer Option 1: As an object with action creators

Answer Option 2: As a function that returns an object with action creators

Answer Option 3: As a function that returns a function that dispatches an action

Answer Option 4: As a separate file with action creators

Correct Response: 1, 2, 3

Explanation: There are different ways to write mapDispatchToProps() in React Redux, including as an object with action creators, as a function that returns an object with action creators, or as a function that returns a function that dispatches an action. Each approach has its own advantages and disadvantages depending on the specific use case.

[Reference: https://redux.js.org/usage/using-react-redux#defining-mapdispatchtoprops]

Question: What is the use of the ownProps parameter in mapStateToProps() and mapDispatchToProps()?

Answer Option 1: It provides access to the Redux store's state

Answer Option 2: It provides access to the component's own props

Answer Option 3: It provides access to the dispatch function

Answer Option 4:

Correct Response: 2

Explanation: The ownProps parameter in mapStateToProps() and mapDispatchToProps() provides access to the component's own props. This can be useful when you need to use a prop value to compute a new prop value or when you need to pass a prop to an action creator.

[Reference: https://react-redux.js.org/api/connect#ownprops]

Question: How to structure Redux top level directories?

Answer Option 1: By feature

Answer Option 2: By layer

Answer Option 3: By file type

Answer Option 4:

Correct Response: 1

Explanation: A common way to structure top-level directories in a Redux application is by feature. This means grouping all related actions, reducers, and components into a single directory for each feature of the application. This helps keep the code organized and makes it easier to reason about.

[Reference: https://redux.js.org/style-guide/style-guide#structure-files-as-feature-folders-or-ducks]

Question: What is redux-saga?

Answer Option 1: A middleware library for Redux

Answer Option 2: A tool for code splitting in React

Answer Option 3: A data visualization library for Redux

Answer Option 4:

Correct Response: 1

Explanation: redux-saga is a middleware library for Redux that allows you to handle side effects, such as asynchronous data fetching or complex state updates, in a declarative and testable way. It uses ES6 Generators to provide a more readable and maintainable way to handle complex logic in your Redux application.

[Reference: https://redux-saga.js.org/]

Question: What is the mental model of redux-saga?

Answer Option 1: Asynchronous call stack

Answer Option 2: Finite state machine

Answer Option 3: Observer pattern

Answer Option 4:

Correct Response: 2

Explanation: The mental model of redux-saga is that of a finite state machine. It allows you to define a sequence of steps for handling a specific side effect and provides a way to manage the state of that sequence. This makes it easier to reason about and test complex logic in your Redux application.

[Reference: https://redux-saga.js.org/docs/introduction/BeginnerTutorial.html#mental-model]

Question: What are the differences between call and put in redux-saga?

Answer Option 1: call is used to call a function, while put is used to dispatch an action

Answer Option 2: call is synchronous, while put is asynchronous

Answer Option 3: call is blocking, while put is non-blocking

Answer Option 4: call is used to handle side effects, while put is used to update the state

Correct Response: 1, 3

Explanation: call and put are both effects provided by redux-saga, but they have different purposes. call is used to call a function, which may be a blocking or non-blocking operation, while put is used to dispatch an action to the Redux store. call is a blocking effect, which means that the generator will wait for the function to return before continuing, while put is a non-blocking effect, which means that the generator will continue immediately after dispatching the action.

[Reference: https://redux-saga.js.org/docs/basics/DeclarativeEffects.html]

Question: What is Redux Thunk?

Answer Option 1: A Redux middleware for handling asynchronous actions

Answer Option 2: A React component for handling forms

Answer Option 3: A JavaScript testing framework

Answer Option 4: A UI toolkit for building web applications

Correct Response: 1

Explanation: Redux Thunk is a middleware for Redux that allows you to write asynchronous logic that interacts with a Redux store. It enables you to dispatch asynchronous actions, such as API requests, and handle them in a synchronous way.

[Reference: https://github.com/reduxjs/redux-thunk]

Question: What are the differences between redux-saga and redux-thunk?

Answer Option 1: Approach to handling asynchronous logic

Answer Option 2: Use of ES6 Generators

Answer Option 3: Integration with React Router

Answer Option 4: Support for optimistic updates

Correct Response: 1, 2, 4

Explanation: Both redux-saga and redux-thunk are middleware solutions for Redux that allow you to write asynchronous logic that interacts with a Redux store. However, they differ in their approach to handling asynchronous logic. Redux Thunk uses function composition to handle asynchronous actions, while Redux Saga uses ES6 Generators. Additionally, Redux Saga supports optimistic updates, which allow you to immediately update the UI while waiting for a response from an asynchronous operation. React Router integration is not a feature that is specific to either middleware.

[Reference: https://blog.logrocket.com/redux-saga-vs-redux-thunk/]

Question: What is Redux DevTools?

Answer Option 1: A Redux middleware for handling asynchronous actions

Answer Option 2: A debugging tool for Redux applications

Answer Option 3: A UI toolkit for building web applications

Answer Option 4: A React component for handling forms

Correct Response: 2

Explanation: Redux DevTools is a browser extension and development tool that provides a visual interface for debugging and inspecting the state of a Redux application. It allows you to track and debug state changes, as well as inspect actions and reducers in real-time.

[Reference: https://github.com/zalmoxisus/redux-devtools-extension]

Question: What are Redux selectors and why to use them?

Answer Option 1: Functions that transform Redux state into a more useful format

Answer Option 2: A feature of React that allows you to select DOM elements

Answer Option 3: A way to handle asynchronous actions in Redux

Answer Option 4: A tool for debugging Redux applications

Correct Response: 1

Explanation: Redux selectors are functions that transform the Redux state into a more useful format for the application. They allow you to abstract the details of the state shape and provide a simpler interface for accessing and manipulating the state. This helps to keep the code more maintainable and reduces the risk of bugs when the state shape changes.

[Reference: https://redux.js.org/introduction/learning-resources#selectors]

Question: What is Redux Form?

Answer Option 1: A library for managing forms in Redux applications

Answer Option 2: A middleware for handling asynchronous actions

Answer Option 3: A React component for building UI forms

Answer Option 4: A feature of React for managing application state

Correct Response: 1

Explanation: Redux Form is a library that provides a simple and efficient way to manage forms in Redux applications. It allows you to easily create and manage form components using a simple declarative syntax. Redux Form provides a range of features, including form validation, field normalization, and asynchronous form submission.

[Reference: https://redux-form.com/]

Question: What are the main features of Redux Form?

Answer Option 1: Form validation

Answer Option 2: Field normalization

Answer Option 3: Asynchronous form submission

Answer Option 4: Integration with React Router

Correct Response: 1, 2, 3

Explanation: Redux Form provides several features for managing forms in Redux applications. Form validation allows you to validate user input and provide feedback to the user when input is invalid. Field normalization allows you to modify user input before it is submitted to the server. Asynchronous form submission allows you to submit form data asynchronously, for example, by using AJAX. Integration with React Router is not a feature that is specific to Redux Form.

[Reference: https://redux-form.com/]

Question: How to add multiple middlewares to Redux?

Answer Option 1: Use the applyMiddleware() function with an array of middleware functions

Answer Option 2: Add each middleware function to the Redux store one at a time

Answer Option 3: Use the combineMiddlewares() function to combine multiple middleware functions

Answer Option 4: There is no way to add multiple middlewares to Redux

Correct Response: 1

Explanation: To add multiple middlewares to Redux, you can use the applyMiddleware() function provided by the Redux library. This function takes an arbitrary number of middleware functions as arguments and returns a single function that can be passed to the createStore() function. The applyMiddleware() function applies the middleware functions to the store in the order they are provided.

[Reference: https://redux.js.org/advanced/middleware#applying-multiple-middlewares]

Question: How to set initial state in Redux?

Answer Option 1: Set the initialState property in the reducer function

Answer Option 2: Dispatch an action to set the initial state

Answer Option 3: Use the createStore() function to set the initial state

Answer Option 4: There is no way to set the initial state in Redux

Correct Response: 1

Explanation: To set the initial state in Redux, you can set the initialState property in the reducer function when creating the Redux store. This allows you to set the initial state for the entire Redux store or for a specific slice of the store.

[Reference: https://redux.js.org/recipes/structuring-reducers/initializing-state]

Question: How Relay is different from Redux?

Answer Option 1: Relay is a library for handling forms in Redux applications, while Redux is a state management library

Answer Option 2: Relay is a state management library, while Redux is a library for handling network requests

Answer Option 3: Relay is a library for handling data fetching and caching, while Redux is a state management library

Answer Option 4: Relay is a library for handling routing in Redux applications, while Redux is a state management library

Correct Response: 3

Explanation: Relay is a library for handling data fetching and caching in React applications. It is often used in conjunction with GraphQL APIs. While Redux is a state management library, Relay focuses specifically on data fetching and caching. Relay provides a number of features, such as declarative data requirements, automatic query generation, and optimistic updates.

[Reference: https://relay.dev/docs/getting-started/introduction/]

Question: What is an action in Redux?

Answer Option 1: A function that transforms the Redux store state

Answer Option 2: An object that describes a change in the Redux store state

Answer Option 3: A middleware function that handles asynchronous actions

Answer Option 4: A component that is rendered in response to user input

Correct Response: 2

Explanation: An action in Redux is an object that describes a change in the Redux store state. Actions are the only way to update the state of the Redux store, and they must have a type property that describes the type of action being performed. In addition to the type property, actions can also have additional data that is used to update the state.

[Reference: https://redux.js.org/tutorials/fundamentals/part-2-concepts-data-flow]

Question: What is the difference between React Native and React?

Answer Option 1: React Native is a mobile app development framework, while React is a web development framework

Answer Option 2: React Native is a JavaScript library, while React is a markup language

Answer Option 3: React Native is used for developing web applications, while React is used for developing mobile applications

Answer Option 4: React Native and React are the same thing

Correct Response: 1

Explanation: React Native is a mobile app development framework that allows developers to build mobile applications using JavaScript and React. It is a separate technology from React, which is a JavaScript library for building user interfaces on the web. While both React and React Native use a similar programming model, they have different APIs and are optimized for different platforms.

[Reference: https://reactnative.dev/docs/getting-started]

Question: How to test React Native apps?

Answer Option 1: Use the Jest testing framework

Answer Option 2: Manually test the app on a device or emulator

Answer Option 3: Use the Mocha testing framework

Answer Option 4: There is no way to test React Native apps

Correct Response: 1

Explanation: To test React Native apps, you can use the Jest testing framework. Jest is a JavaScript testing framework that is widely used in the React and React Native communities. Jest provides a range of features for testing React Native apps, including snapshot testing, mocking, and coverage reports.

[Reference: https://jestjs.io/docs/en/tutorial-react-native]

Question: How to do logging in React Native?

Answer Option 1: Use the console.log() function

Answer Option 2: Use the alert() function

Answer Option 3: Use the debugger statement

Answer Option 4: All of the above

Correct Response: 1

Explanation: To do logging in React Native, you can use the console.log() function, which writes messages to the console in the same way as in a web browser. You can also use other console functions, such as console.warn() and console.error(), to log warning and error messages.

[Reference: https://reactnative.dev/docs/debugging]

Question: How to debug your React Native app?

Answer Option 1: Use console.log() statements

Answer Option 2: Use the React Native Debugger tool

Answer Option 3: Use the Chrome DevTools

Answer Option 4: There is no way to debug React Native apps

Correct Response: 2

Explanation: To debug your React Native app, you can use the React Native Debugger tool. This is a standalone debugging tool that provides a range of features for debugging React Native apps, including console logging, breakpoints, and network inspection. The React Native Debugger tool can be used with both iOS and Android devices and emulators.

[Reference: https://reactnative.dev/docs/debugging]

Question: What is reselect and how it works?

Answer Option 1: A middleware for handling asynchronous actions in Redux

Answer Option 2: A library for managing forms in Redux applications

Answer Option 3: A tool for debugging React Native applications

Answer Option 4: A library for optimizing Redux selectors

Correct Response: 4

Explanation: Reselect is a library for optimizing Redux selectors. It provides a way to create memoized selectors that only recompute when their input selectors have changed. This can help to improve performance and reduce unnecessary re-renders in React applications. Reselect works by caching the results of previous computations and only re-executing the computation if the input selectors have changed.

[Reference: https://github.com/reduxjs/reselect]

Question: What is Flow?

Answer Option 1: A JavaScript testing framework

Answer Option 2: A state management library

Answer Option 3: A type checking tool for JavaScript

Answer Option 4: A styling library for React

Correct Response: 3

Explanation: Flow is a type checking tool for JavaScript that is commonly used in React and React Native applications. Flow allows you to add static types to your JavaScript code, which can help to catch errors at compile-time instead of at runtime. Flow is particularly useful in large codebases where it can be difficult to keep track of all the variables and function calls.

[Reference: https://flow.org/]

Question: What is the difference between Flow and PropTypes?

Answer Option 1: Flow is a type checking tool for JavaScript, while PropTypes is a runtime type checking library

Answer Option 2: Flow and PropTypes are the same thing

Answer Option 3: PropTypes is a type checking tool for JavaScript, while Flow is a runtime type checking library

Answer Option 4: PropTypes is a runtime validation library for React components

Correct Response: 1

Explanation: The main difference between Flow and PropTypes is that Flow is a type checking tool for JavaScript that checks types at compile-time, while PropTypes is a runtime type checking library for React components that checks types at runtime. Flow is particularly useful in large codebases where it can be difficult to keep track of all the variables and function calls, while PropTypes is useful for catching errors in React components at runtime.

[Reference: https://reactjs.org/docs/typechecking-with-proptypes.html, https://flow.org/]

Question: How to use font-awesome icons in React?

Answer Option 1: Use the Font Awesome component library for React

Answer Option 2: Import the Font Awesome CSS file and use the class names

Answer Option 3: Use the React Native vector icons library

Answer Option 4: There is no way to use Font Awesome icons in React

Correct Response: 1

Explanation: To use Font Awesome icons in React, you can use the Font Awesome component library for React. This library provides a set of pre-built components for rendering Font Awesome icons, which you can use in your React applications. Alternatively, you can import the Font Awesome CSS file and use the class names directly in your components.

[Reference: https://fontawesome.com/how-to-use/on-the-web/using-with/react]

Question: What is React Dev Tools?

Answer Option 1: A library for managing forms in Redux applications

Answer Option 2: A tool for testing React components

Answer Option 3: A debugging tool for React applications

Answer Option 4: A state management library for React

Correct Response: 3

Explanation: React Dev Tools is a browser extension and development tool that provides a visual interface for debugging and inspecting the state of a React application. It allows you to track and debug the state and props of your components, as well as inspect the component tree and the performance of your application. React Dev Tools is available for Chrome, Firefox, and other major browsers.

[Reference: https://reactjs.org/blog/2019/08/15/new-react-devtools.html]

Question: Why is DevTools not loading in Chrome for local files?

Answer Option 1: DevTools is not supported for local files

Answer Option 2: The browser security settings prevent DevTools from loading for local files

Answer Option 3: The React app is not running in development mode

Answer Option 4: None of the above

Correct Response: 2

Explanation: DevTools may not load for local files in Chrome due to the browser's security settings. To fix this issue, you can start Chrome with the --allow-file-access-from-files flag, which disables the security check for local files. Alternatively, you can run the React app on a local web server instead of opening the HTML file directly in the browser.

[Reference: https://stackoverflow.com/questions/18586921/why-is-chrome-ignoring-my-jquery-ajax-call-when-i-load-a-local-file]

Question: How to use Polymer in React?

Answer Option 1: Use the Polymer CLI to generate a React app with Polymer components

Answer Option 2: Use the @polymer/reactive-elements library

Answer Option 3: Use the React Polymer bridge library

Answer Option 4: There is no way to use Polymer in React

Correct Response: 2

Explanation: To use Polymer in React, you can use the @polymer/reactive-elements library. This library provides React components that wrap Polymer elements, allowing you to use them in your React applications. The @polymer/reactive-elements library also provides a way to create new React components that extend existing Polymer elements.

[Reference: https://github.com/Polymer/reactive-elements]

Question: What are the advantages of React over Vue.js?

Answer Option 1: Stronger community support

Answer Option 2: Better performance with large data sets

Answer Option 3: More flexible component structure

Answer Option 4: Better integration with existing apps

Correct Response: 1, 2, 4

Explanation: React and Vue.js are both popular front-end JavaScript frameworks, but React has several advantages over Vue.js. React has a larger and more active community, which means there are more resources and support available. React also performs better with large data sets and has a more flexible component structure. However, Vue.js has advantages in other areas, such as ease of use and a simpler learning curve.

[Reference: https://www.codingdojo.com/blog/react-vs-vue-which-is-the-best-choice-for-2020]

Question: Why React tab is not showing up in DevTools?

Answer Option 1: The React DevTools extension is not installed

Answer Option 2: The React app is not running in development mode

Answer Option 3: There is a conflict with other browser extensions

Answer Option 4: The React app is not using React version 16 or higher

Correct Response: 4

Explanation: If the React tab is not showing up in DevTools, it may be because the React app is not using React version 16 or higher. The React tab was introduced in version 16, so if you are using an earlier version of React, it will not be available in DevTools. You can check the React version in your app by looking at the package.json file or by running the command 'npm ls react'.

[Reference: https://reactjs.org/blog/2017/09/26/react-v16.0.html#new-features]

Question: What are styled components?

Answer Option 1: A way to define styles in a separate CSS file

Answer Option 2: A way to define styles using inline styles in JavaScript

Answer Option 3: A way to define styles using a preprocessor like Sass or Less

Answer Option 4: A library for creating React components with embedded styles

Correct Response: 4

Explanation: Styled components are a library for creating React components with embedded styles. They allow you to define styles directly in your JavaScript code using tagged template literals, which makes it easy to style your components based on props and state. Styled components also provide a number of other features, such as server-side rendering, theming, and global styles.

[Reference: https://styled-components.com/]

Question: What is Relay?

Answer Option 1: A type checking tool for JavaScript

Answer Option 2: A library for managing forms in React applications

Answer Option 3: A library for managing network requests in React applications

Answer Option 4: A state management library for React

Correct Response: 3

Explanation: Relay is a library for managing network requests in React applications. It is often used in conjunction with GraphQL APIs and provides a number of features for managing data fetching and caching, such as automatic batching, optimistic updates, and pagination. Relay also provides a way to declare data dependencies for your components, which can help to reduce unnecessary re-renders.

[Reference: https://relay.dev/docs/en/introduction-to-relay]

Question: How to use TypeScript in create-react-app application?

Answer Option 1: Use the --typescript flag when creating the app

Answer Option 2: Add the typescript dependency and update the configuration files

Answer Option 3: Use the create-react-app-ts tool instead of create-react-app

Answer Option 4:

Correct Response: 2

Explanation: To use TypeScript in a create-react-app application, you need to add the typescript dependency and update the configuration files. You can do this manually or by using a tool like react-scripts-ts. After installing the typescript dependency, you need to rename some files and update the configuration files to use TypeScript instead of JavaScript.

[Reference: https://create-react-app.dev/docs/adding-typescript/]

Question: What are the main features of reselect library?

Answer Option 1: Memoization of selectors

Answer Option 2: Composable selectors

Answer Option 3: Easy integration with Redux

Answer Option 4: Support for asynchronous selectors

Correct Response: 1, 2, 3

Explanation: The main features of the reselect library are memoization of selectors, composability of selectors, and easy integration with Redux. Reselect provides a way to create memoized selectors that only recompute when their input selectors have changed, which can help to improve performance and reduce unnecessary re-renders in React applications. Reselect also allows you to compose selectors together to create more complex selectors, and it works seamlessly with Redux to provide a way to select data from the Redux store.

[Reference: https://github.com/reduxjs/reselect]

Question: Give an example of reselect usage?

Answer Option 1: A selector that returns the length of an array

Answer Option 2: A selector that filters and sorts data from the Redux store

Answer Option 3: A selector that computes the sum of two numbers

Answer Option 4:

Correct Response: 2

Explanation: An example of reselect usage is a selector that filters and sorts data from the Redux store. This selector can take other selectors as input and use them to filter and sort the data before returning it. For example, you could create a selector that filters a list of items by a certain category and then sorts the results by price. This selector would only recompute when the input selectors (category and items) have changed, which can help to improve performance.

[Reference: https://github.com/reduxjs/reselect#example]

Question: Does the statics object work with ES6 classes in React?

Answer Option 1: Yes, the statics object works with ES6 classes

Answer Option 2: No, the statics object only works with React.createClass()

Answer Option 3:

Answer Option 4:

Correct Response: 1

Explanation: Yes, the statics object works with ES6 classes in React. The statics object is used to define static properties for a React component, such as defaultProps or propTypes. In ES6 classes, you can define static properties using the static keyword, like so: 'static defaultProps = {...}'. This is equivalent to using the statics object in React.createClass().

[Reference: https://reactjs.org/docs/reusable-components.html#es6-classes]

Question: Can Redux only be used with React?

Answer Option 1: Yes, Redux can only be used with React

Answer Option 2: No, Redux can be used with any JavaScript framework or library

Answer Option 3:

Answer Option 4:

Correct Response: 2

Explanation: No, Redux can be used with any JavaScript framework or library, not just React. Redux is a standalone state management library that can be used with any UI framework or library, as well as with vanilla JavaScript applications. While it is commonly used with React, it can also be used with other popular frameworks such as Angular, Vue.js, and Ember.

[Reference: https://redux.js.org/introduction/ecosystem]

Question: Do you need to have a particular build tool to use Redux?

Answer Option 1: Yes, you need to use Webpack to use Redux

Answer Option 2: No, you can use any build tool with Redux

Answer Option 3:

Answer Option 4:

Correct Response: 2

Explanation: No, you do not need to use a particular build tool to use Redux. Redux is a standalone library that can be used with any JavaScript build tool, including Webpack, Rollup, and Browserify. However, some build tools may have specific plugins or configurations that make it easier to work with Redux, so it's worth checking the documentation for your build tool to see if there are any recommended setups.

[Reference: https://redux.js.org/faq/general#do-i-have-to-use-redux-with-react]

Question: How Redux Form initialValues get updated from state?

Answer Option 1: initialValues are automatically synced with the Redux store

Answer Option 2: You need to manually update initialValues in the form component

Answer Option 3: initialValues are passed in as a prop to the form component

Answer Option 4:

Correct Response: 3

Explanation: The initialValues prop in Redux Form is used to set the initial values for the form fields. You can pass in an object of initial values as the initialValues prop, which will be used to populate the form fields when the form is first rendered. The initialValues can be sourced from the Redux store, and can be updated by dispatching an action that updates the relevant values in the store. The form component will automatically update when the initialValues prop changes.

[Reference: https://redux-form.com/8.3.0/docs/api/props.md/#-code-initialvalues-object-code-optional]

Question: How React PropTypes allow different type for one prop?

Answer Option 1: PropTypes does not allow different types for one prop

Answer Option 2: You need to define separate PropTypes for each type

Answer Option 3: You can use PropTypes.oneOfType()

Answer Option 4:

Correct Response: 3

Explanation: You can use PropTypes.oneOfType() to allow a prop to have multiple possible types. For example, you can define a prop that can be either a string or a number like this: 'myProp: PropTypes.oneOfType([PropTypes.string, PropTypes.number])'. This will allow the prop to accept values of either type, and will throw a warning if a value of a different type is passed in.

[Reference: https://reactjs.org/docs/typechecking-with-proptypes.html#oneof-type-checking]

Question: Can I import an SVG file as react component?

Answer Option 1: Yes, you can import an SVG file as a React component

Answer Option 2: No, SVG files cannot be used as React components

Answer Option 3:

Answer Option 4:

Correct Response: 1

Explanation: Yes, you can import an SVG file as a React component using the '@svgr/webpack' loader or the 'svg-react-loader' package. These tools allow you to import an SVG file as a React component, which you can then use in your React application just like any other component. This can be useful for creating reusable SVG icons or graphics.

[Reference: https://react-svgr.com/docs/webpack/]

Question: Why are inline ref callbacks or functions not recommended?

Answer Option 1: They can cause memory leaks

Answer Option 2: They can introduce performance issues

Answer Option 3: They are not compatible with all browsers

Answer Option 4: They can interfere with the component lifecycle

Correct Response: 1

Explanation: Inline ref callbacks or functions are not recommended because they can cause memory leaks in your application. When using an inline ref callback, the function is recreated on each render, which can result in a buildup of references that are not properly cleaned up by the garbage collector. This can lead to memory leaks over time. It is recommended to use the createRef() API or a callback ref defined outside of the component to avoid these issues.

[Reference: https://reactjs.org/docs/refs-and-the-dom.html#caveats-with-callback-refs]

Question: What is render hijacking in React?

Answer Option 1: When a component's render method is called multiple times

Answer Option 2: When a component modifies the rendering behavior of its children

Answer Option 3:

Answer Option 4:

Correct Response: 2

Explanation: Render hijacking is a technique used in React where a component modifies the rendering behavior of its children by wrapping them in higher-order components (HOCs). This can be used to add or modify props, add event handlers, or manipulate the rendering of the children in other ways. While this can be a powerful technique, it can also lead to code that is difficult to reason about and maintain.

[Reference: https://reactpatterns.com/#render-hijacking]

sQuestion: What are HOC factory implementations?

Answer Option 1: Higher-order components that return a function

Answer Option 2: Higher-order components that return a component

Answer Option 3: Higher-order components that are used to create other higher-order components

Answer Option 4:

Correct Response: 1

Explanation: HOC factory implementations are higher-order components that return a function. This function can then be used to create a new higher-order component that has a specific set of props or behavior. This can be useful for creating reusable HOCs that can be customized for specific use cases. For example, you could create a withData HOC factory that takes a data source as an argument and returns a HOC that fetches and passes data to the wrapped component.

[Reference: https://reactpatterns.com/#higher-order-component-factories]

Question: How to pass numbers to React component?

Answer Option 1: Wrap the number in quotes as a string

Answer Option 2: Use the Number() constructor

Answer Option 3: Use the parseInt() or parseFloat() functions

Answer Option 4: Pass the number directly as a prop

Correct Response: 4

Explanation: To pass a number to a React component, you can simply pass the number directly as a prop. React will automatically handle the conversion of the number to a string when rendering the component. For example, you could pass a number like this: ' <MyComponent myProp= {42} />'. If you need to pass a number as a string, you can wrap it in quotes like any other string.

[Reference: https://reactjs.org/docs/jsx-in-depth.html#javascript-expressions]

Question: Do I need to keep all my state into Redux? Should I ever use react internal state?

Answer Option 1: Yes, you should always use Redux to manage all of your application state

Answer Option 2: No, you should never use Redux and always use React internal state

Answer Option 3: It depends on the complexity and size of your application

Answer Option 4: It depends on whether you are using class or functional components

Correct Response: 3

Explanation: Whether or not to use Redux to manage state in a React application depends on the complexity and size of the application. For small and simple applications, it may be sufficient to use React's internal state management. However, for larger and more complex applications, Redux can be a helpful tool for managing application state.

[Reference: https://redux.js.org/faq/general#when-should-i-use-redux]

Question: What is the purpose of registerServiceWorker in React?

Answer Option 1: To register a new service worker with the browser

Answer Option 2: To enable client-side caching of static assets

Answer Option 3: To enable offline access to the application

Answer Option 4: All of the above

Correct Response: 4

Explanation: registerServiceWorker is a utility function provided by the Create React App tool that registers a new service worker with the browser. This service worker enables client-side caching of static assets and enables offline access to the application.

[Reference: https://create-react-app.dev/docs/making-a-progressive-web-app/]

Question: What is React memo function?

Answer Option 1: A function that creates a memoized version of a component

Answer Option 2: A function that returns a higher-order component

Answer Option 3: A function that creates a new instance of a component

Answer Option 4: A function that creates a new element in the DOM

Correct Response: 1

Explanation: React memo is a higher-order function that creates a memoized version of a component. This memoized version can help improve performance by avoiding unnecessary re-renders of the component. The memo function works by caching the result of the component's render method and comparing it to the previous result. If the result is the same, the component is not re-rendered.

[Reference: https://reactjs.org/docs/react-api.html#reactmemo]

Question: What is React lazy function?

Answer Option 1: A function that creates a lazy version of a component

Answer Option 2: A function that returns a higher-order component

Answer Option 3: A function that creates a new instance of a component

Answer Option 4: A function that creates a new element in the DOM

Correct Response: 1

Explanation: React lazy is a function that creates a lazy version of a component. This lazy version is loaded only when it is actually needed, such as when a user navigates to a page that requires the component. This can help improve performance by reducing the initial load time of the application.

[Reference: https://reactjs.org/docs/code-splitting.html#reactlazy]

Question: How to prevent unnecessary updates using setState?

Answer Option 1: Use shouldComponentUpdate lifecycle method

Answer Option 2: Use componentDidUpdate lifecycle method

Answer Option 3: Use componentWillUnmount lifecycle method

Answer Option 4: Use React.PureComponent

Correct Response: 1

Explanation: The shouldComponentUpdate lifecycle method can be used to prevent unnecessary updates to a component when its props or state have not changed. This method should return a boolean value indicating whether the component should update or not. By default, the method returns true, but it can be overridden to provide custom logic for determining whether an update is necessary.

[Reference: https://reactjs.org/docs/react-component.html#shouldcomponentupdate]

Question: How do you render Array, Strings and Numbers in React 16 Version?

Answer Option 1: Use createElement method

Answer Option 2: Use ReactDOM.render method

Answer Option 3: Use JSX syntax

Answer Option 4: Use React.createClass method

Correct Response: 3

Explanation: In React 16 and later versions, Array, Strings, and Numbers can be rendered using JSX syntax. For example, an array can be rendered using the map method, a string can be rendered using curly braces, and a number can be rendered directly.

[Reference: https://reactjs.org/docs/introducing-jsx.html#embedding-expressions-in-jsx]

Question: How to use class field declarations syntax in React classes?

Answer Option 1: Use the constructor method

Answer Option 2: Use the componentWillMount lifecycle method

Answer Option 3: Use the componentDidMount lifecycle method

Answer Option 4: Use the class fields syntax

Correct Response: 4

Explanation: The class fields syntax can be used to declare class properties directly on the class without the need for a constructor method. This syntax can be used in React classes to declare state and other properties.

[Reference: https://reactjs.org/docs/state-and-lifecycle.html#adding-local-state-to-a-class]

Question: What are hooks?

Answer Option 1: Functions that let you "hook into" React state and lifecycle features from functional components

Answer Option 2: Functions that let you modify the DOM directly from React components

Answer Option 3: Functions that let you render custom elements in React

Answer Option 4: Functions that let you create reusable UI components in React

Correct Response: 1

Explanation: Hooks are functions that allow developers to "hook into" React state and lifecycle features from functional components. This means that functional components can now have state and lifecycle methods, making them more powerful and flexible. Hooks were introduced in React 16.8 and have since become a popular way of managing state and adding behavior to functional components.

[Reference: https://reactjs.org/docs/hooks-intro.html]

Question: What rules need to be followed for hooks?

Answer Option 1: Hooks can only be used in functional components

Answer Option 2: Hooks must be called in the same order on every render

Answer Option 3: Hooks cannot be used with class components

Answer Option 4: Hooks must be called at the beginning of a component's render method

Correct Response: 2

Explanation: The rules for using hooks in React include calling hooks in the same order on every render and not calling hooks inside loops, conditions, or nested functions. Additionally, hooks can only be used in functional components or custom hooks, not in class components.

[Reference: https://reactjs.org/docs/hooks-rules.html]

Question: How to ensure hooks followed the rules in your project?

Answer Option 1: Use a linter plugin such as eslint-plugin-react-hooks

Answer Option 2: Use a testing framework to check for rule violations

Answer Option 3: Use a code review process to check for rule violations

Answer Option 4: All of the above

Correct Response: 4

Explanation: To ensure that hooks are being used correctly in a project, developers can use a linter plugin such as eslint-plugin-react-hooks to check for rule violations. They can also use a testing framework to check that the application behaves as expected when hooks are used. Finally, a code review process can help catch any rule violations before they are merged into the codebase.

[Reference: https://reactjs.org/docs/hooks-rules.html#eslint-plugin]

Question: What are the differences between Flux and Redux?

Answer Option 1: Flux is a pattern, while Redux is a library

Answer Option 2: Flux has a single store, while Redux has multiple stores

Answer Option 3: Flux has actions, stores, and views, while Redux has actions, reducers, and a store

Answer Option 4: Flux is a Facebook-created technology, while Redux is an open-source technology

Correct Response: 1, 3

Explanation: Flux is a pattern for managing state in a React application, while Redux is a library that implements the Flux pattern. Both Flux and Redux have actions, but Flux has stores and views, while Redux has reducers and a store. Additionally, Flux was created by Facebook, while Redux is an open-source library.

[Reference: https://redux.js.org/faq/general#what-are-the-differences-between-redux-and-flux]

Question: What are the benefits of React Router V4?

Answer Option 1: Dynamic routing

Answer Option 2: Lazy loading

Answer Option 3: Server-side rendering

Answer Option 4: Simpler API

Correct Response: 1, 2, 4

Explanation: React Router V4 introduced several new features and benefits over previous versions, including dynamic routing, lazy loading of components, and a simpler and more intuitive API. Additionally, React Router V4 allows for better performance and easier debugging of routing issues. Server-side rendering is not a new feature introduced in React Router V4, but it can be achieved with the library.

[Reference: https://reacttraining.com/react-router/web/guides/philosophy]

Question: Can you describe the componentDidCatch lifecycle method signature?

Answer Option 1: componentDidCatch(error: Error, errorInfo: object)

Answer Option 2: componentDidCatch(error: string, errorInfo: object)

Answer Option 3: componentDidCatch(error: Error, errorInfo: string)

Answer Option 4: componentDidCatch(error: string, errorInfo: string)

Correct Response: 1

Explanation: The componentDidCatch lifecycle method is called whenever an error is thrown in a component's child tree. The method has a signature of componentDidCatch(error: Error, errorInfo: object). The error parameter is the actual error object that was thrown, while the errorInfo parameter is an object that contains additional information about the error, such as the component stack trace.

[Reference: https://reactjs.org/docs/react-component.html#componentdidcatch]

Question: In which scenarios error boundaries do not catch errors?

Answer Option 1: Inside event handlers

Answer Option 2: During server-side rendering

Answer Option 3: Errors thrown in the error boundary itself

Answer Option 4: Errors thrown in lifecycle methods

Correct Response: 1, 2, 3

Explanation: Error boundaries in React are designed to catch errors that occur during rendering, but there are certain scenarios where they may not catch errors. For example, errors thrown inside event handlers are not caught by error boundaries, and errors thrown during server-side rendering may not be caught until runtime. Additionally, errors thrown within the error boundary itself or within lifecycle methods of the component are not caught by the error boundary.

[Reference: https://reactjs.org/docs/error-boundaries.html#how-about-event-handlers]

Question: Why do you not need error boundaries for event handlers?

Answer Option 1: Because event handlers are not executed within the component tree

Answer Option 2: Because event handlers are not prone to errors

Answer Option 3: Because event handlers are executed asynchronously

Answer Option 4: Because event handlers do not cause the component to re-render

Correct Response: 1

Explanation: Error boundaries are not necessary for event handlers in React because event handlers are not executed within the component tree. Instead, they are executed outside of the component hierarchy, and any errors that occur within an event handler will be caught by the global error handling mechanism provided by the browser.

[Reference: https://reactjs.org/docs/error-boundaries.html#how-about-event-handlers]

Question: What is the difference between try-catch block and error boundaries?

Answer Option 1: try-catch block catches all errors, while error boundaries only catch errors in React components

Answer Option 2: Error boundaries can catch errors in child components, while try-catch block only catches errors in the current scope

Answer Option 3: try-catch block can handle both synchronous and asynchronous errors, while error boundaries can only handle synchronous errors

Answer Option 4: Error boundaries are React-specific, while try-catch block is a language feature

Correct Response: 2, 4

Explanation: The try-catch block is a JavaScript language feature that can be used to catch errors in synchronous code within a given scope. In contrast, error boundaries are a React-specific mechanism for catching errors that occur during rendering or in lifecycle methods of a component, and can also catch errors in child components. Additionally, error boundaries can only handle synchronous errors, while try-catch blocks can handle both synchronous and asynchronous errors.

[Reference: https://reactjs.org/docs/error-boundaries.html#introducing-error-boundaries]

Question: What is the behavior of uncaught errors in React 16?

Answer Option 1: They are ignored

Answer Option 2: They are logged to the console

Answer Option 3: They trigger a fatal error and crash the application

Answer Option 4: They trigger an error boundary to catch the error

Correct Response: 3

Explanation: Uncaught errors in React 16 and later versions trigger a fatal error and crash the application. This behavior was introduced in order to prevent subtle bugs and inconsistencies that could result from errors being silently ignored or logged to the console.

[Reference: https://reactjs.org/blog/2017/09/26/react-v16.0.html#breaking-changes]

Question: What is the proper placement for error boundaries?

Answer Option 1: They should be placed at the top level of the component tree

Answer Option 2: They should be placed in every component that renders other components

Answer Option 3: They should be placed in the root element of the application

Answer Option 4: They can be placed anywhere in the component tree

Correct Response: 1

Explanation: Error boundaries in React should be placed at the top level of the component tree, as close to the root of the application as possible. This ensures that errors are caught and handled as early as possible in the rendering process, and prevents them from affecting other components or causing cascading errors.

[Reference: https://reactjs.org/docs/error-boundaries.html#how-to-use-error-boundaries]

Question: What is the benefit of component stack trace from error boundary?

Answer Option 1: It helps identify the component that caused the error

Answer Option 2: It provides a detailed explanation of the error

Answer Option 3: It suggests possible solutions to the error

Answer Option 4: It prevents the error from being thrown again

Correct Response: 1

Explanation: The component stack trace provided by an error boundary in React can be useful in identifying the component that caused the error. This can be helpful in debugging and resolving issues with the application. The stack trace includes information about the component hierarchy and the order in which components were rendered, making it easier to trace the source of the error.

[Reference: https://reactjs.org/docs/error-boundaries.html#component-stack-traces]

Question: What is the required method to be defined for a class component?

Answer Option 1: render()

Answer Option 2: constructor()

Answer Option 3: componentDidMount()

Answer Option 4: setState()

Correct Response: 1

Explanation: The render() method is the required method to be defined for a class component in React. This method is responsible for rendering the component and returning the resulting element tree. All other methods, such as constructor() and componentDidMount(), are optional and serve specific purposes in the component lifecycle.

[Reference: https://reactjs.org/docs/react-component.html#render]

Question: What are the possible return types of render method?

Answer Option 1: React elements

Answer Option 2: Strings

Answer Option 3: Numbers

Answer Option 4: Arrays

Correct Response: 1, 4

Explanation: The render() method in React can return React elements, such as a div or a span, or an array of React elements. Other data types such as strings or numbers cannot be returned by the render() method.

[Reference: https://reactjs.org/docs/rendering-elements.html]

Question: What is the main purpose of constructor?

Answer Option 1: To initialize the component's state and bind methods to the component

Answer Option 2: To define the component's markup and styling

Answer Option 3: To render the component's children

Answer Option 4: To handle events and update the component's state

Correct Response: 1

Explanation: The main purpose of the constructor in a React component is to initialize the component's state and bind methods to the component. The constructor is called before the component is mounted and can be used to set the initial state of the component or to bind methods to the component.

[Reference: https://reactjs.org/docs/react-component.html#constructor]

Question: Is it mandatory to define constructor for React component?

Answer Option 1: Yes, it is mandatory for all React components

Answer Option 2: No, it is only necessary if the component needs to set its initial state or bind methods to the component

Answer Option 3: No, it is only necessary if the component has props

Answer Option 4:

Correct Response: 2

Explanation: It is not mandatory to define a constructor for a React component. If the component does not need to set its initial state or bind methods to the component, the constructor can be omitted. Additionally, if the component does not have any props, a constructor is not necessary.

[Reference: https://reactjs.org/docs/react-component.html#constructor]

Question: What are default props?

Answer Option 1: Props that are set by the parent component

Answer Option 2: Props that are passed to the component through the URL

Answer Option 3: Props that are assigned default values

Answer Option 4: Props that are passed to the component through a global store

Correct Response: 3

Explanation: Default props are props that are assigned default values in a React component. These default values are used if the prop is not passed to the component from the parent or if it is passed as undefined. Default props are defined using the defaultProps property on the component class.

[Reference: https://reactjs.org/docs/react-component.html#defaultprops]

Question: Why should not call setState in componentWillUnmount?

Answer Option 1: Because it will cause a memory leak

Answer Option 2: Because it will trigger a re-render after the component is unmounted

Answer Option 3: Because the component's state is no longer available after unmounting

Answer Option 4: Because it is unnecessary and can cause performance issues

Correct Response: 3

Explanation: The componentWillUnmount() method is called immediately before a component is unmounted and destroyed. It is not safe to call setState() within this method, as the component's state is no longer available after unmounting. Attempting to update the state after unmounting can cause errors or unexpected behavior in the application.

[Reference: https://reactjs.org/docs/react-component.html#componentwillunmount]

Question: What is the purpose of getDerivedStateFromError?

Answer Option 1: To handle errors that occur during rendering

Answer Option 2: To update the component's state based on the error that occurred

Answer Option 3: To provide additional information about the error to the user

Answer Option 4: To log the error to the console

Correct Response: 1

Explanation: The getDerivedStateFromError() method is a lifecycle method in React that is called whenever an error is thrown during rendering. Its main purpose is to update the component's state with information about the error that occurred, which can then be used to render an error message or to trigger some other action in the application.

[Reference: https://reactjs.org/docs/react-component.html#static-getderivedstatefromerror]

Question: What is the methods order when component re-rendered?

Answer Option 1: constructor(), render(), componentDidMount()

Answer Option 2: render(), componentDidUpdate(), componentWillUnmount()

Answer Option 3: render(), componentDidUpdate(), componentDidMount()

Answer Option 4: componentDidMount(), render(), componentDidUpdate()

Correct Response: 3

Explanation: When a component is re-rendered in React, the order of methods that are called is as follows: render(), componentDidUpdate(), and then componentDidMount(). The render() method is called to update the component's UI based on any changes in props or state, followed by componentDidUpdate() to handle any side effects that may have occurred as a result of the update. Finally, componentDidMount() is called to perform any additional setup that may be necessary for the updated component.

[Reference: https://reactjs.org/docs/react-component.html#updating]

Question: What are the methods invoked during error handling?

Answer Option 1: static getDerivedStateFromError()

Answer Option 2: componentDidCatch()

Answer Option 3: render()

Answer Option 4: componentWillUnmount()

Correct Response: 1, 2

Explanation: When an error is thrown in a component's child tree, React invokes the componentDidCatch() method on the nearest error boundary. This method is responsible for handling the error and updating the component's state with information about the error. In addition, the static getDerivedStateFromError() method can be used to update the component's state with information about the error. The render() method is not directly involved in error handling, but it may be used to render an error message or to conditionally render different UI components based on whether an error has occurred. The componentWillUnmount() method is not invoked during error handling, as it is only called when the component is being unmounted and destroyed.

[Reference: https://reactjs.org/docs/error-boundaries.html#introducing-error-boundaries]

Question: What is the purpose of displayName class property?

Answer Option 1: To set the name of the component's file

Answer Option 2: To set the name of the component for debugging purposes

Answer Option 3: To specify the component's DOM display style

Answer Option 4: To set the component's initial state

Correct Response: 2

Explanation: The displayName class property in a React component is used to set the name of the component for debugging purposes. This name is used in error messages and in the React Developer Tools to help identify the component in the component hierarchy. If the displayName property is not set explicitly, React will attempt to infer the name of the component from the name of the component class or function.

[Reference: https://reactjs.org/docs/react-component.html#displayname]

Question: What is the purpose of unmountComponentAtNode method?

Answer Option 1: To render a component to a specified DOM node

Answer Option 2: To update the state of a mounted component

Answer Option 3: To unmount a component from a specified DOM node

Answer Option 4: To add a new child component to an existing component

Correct Response: 3

Explanation: The unmountComponentAtNode() method in React is used to unmount a component from a specified DOM node. This method removes the component from the DOM and cleans up any associated event listeners or timers. It can be useful in cases where a component needs to be removed from the page dynamically or conditionally.

[Reference: https://reactjs.org/docs/react-dom.html#unmountcomponentatnode]

Question: What is code-splitting?

Answer Option 1: A technique for optimizing React component performance

Answer Option 2: A way to split code into smaller, more manageable chunks

Answer Option 3: A method for separating HTML and CSS in a web application

Answer Option 4: A feature of React that enables components to be reused in multiple contexts

Correct Response: 2

Explanation: Code-splitting is a technique for splitting a large JavaScript bundle into smaller, more manageable chunks. This can improve the performance and load time of a web application by reducing the amount of code that needs to be downloaded and parsed by the browser. In React, code-splitting can be achieved using the dynamic import() method, which allows components to be loaded asynchronously at runtime.

[Reference: https://reactjs.org/docs/code-splitting.html]

Question: What is the benefit of strict mode?

Answer Option 1: It enforces stricter type checking for props and state

Answer Option 2: It improves the performance of React applications

Answer Option 3: It helps identify potential problems in code early on

Answer Option 4: It enables advanced debugging features in React Developer Tools

Correct Response: 3

Explanation: The strict mode feature in React is used to identify potential problems in code early on in the development process. It activates additional checks and warnings for common mistakes and unsafe operations, such as using deprecated lifecycle methods or modifying props directly. This can help prevent bugs and improve the overall quality of the code.

[Reference: https://reactjs.org/docs/strict-mode.html]

Question: What are Keyed Fragments?

Answer Option 1: A way to group elements without adding extra nodes to the DOM

Answer Option 2: A type of encryption used in React components

Answer Option 3: A method for optimizing the rendering of large lists in React

Answer Option 4: A component that can be used to render fragments in React

Correct Response: 1

Explanation: Keyed Fragments are a feature in React that allows elements to be grouped together without adding extra nodes to the DOM. This can improve performance and reduce the complexity of the component hierarchy. Keyed Fragments are created by using the <React.Fragment> element and adding a unique key prop to each child element.

[Reference: https://reactjs.org/docs/fragments.html#keyed-fragments]

Question: Does React support all HTML attributes?

Answer Option 1: Yes, React supports all HTML attributes

Answer Option 2: No, React only supports a subset of HTML attributes

Answer Option 3:

Answer Option 4:

Correct Response: 2

Explanation: React supports a subset of HTML attributes, as not all attributes are applicable or relevant to React components. Some HTML attributes, such as "class" and "for", are reserved words in JavaScript and cannot be used directly in JSX. Instead, React uses the "className" and "htmlFor" attributes, respectively. Additionally, some HTML attributes may have different names or syntax in React, such as "tabindex" being spelled "tabIndex" in React.

[Reference: https://reactjs.org/docs/dom-elements.html]

Question: What are the limitations with HOCs?

Answer Option 1: They can make it difficult to trace the flow of props and state

Answer Option 2: They can cause performance issues if used excessively

Answer Option 3: They can lead to code duplication and complexity

Answer Option 4: They can be difficult to understand and use correctly

Correct Response: 1, 3, 4

Explanation: Higher Order Components (HOCs) are a common pattern in React for enhancing the functionality and reuse of components. However, they also have some limitations and potential drawbacks. HOCs can make it difficult to trace the flow of props and state through a component hierarchy, as multiple HOCs may be applied to a single component. They can also lead to code duplication and complexity if used excessively, and can be difficult to understand and use correctly.

[Reference: https://reactjs.org/docs/higher-order-components.html#drawbacks-of-hocs]

Question: How to debug forwardRefs in DevTools?

Answer Option 1: Use the React Developer Tools to inspect the component hierarchy

Answer Option 2: Add console.log statements to the component code

Answer Option 3: Use the Chrome DevTools to debug the component code

Answer Option 4: Use the React Profiler to analyze component performance

Correct Response: 1

Explanation: Debugging forwardRefs in React can be challenging, as the ref may not be available in the component code itself. One approach is to use the React Developer Tools to inspect the component hierarchy and check the props and state of each component in the tree. This can help identify any issues with the forwardRef and determine if it is being passed correctly to child components.

[Reference: https://reactjs.org/docs/forwarding-refs.html#debugging-tips]

Question: When component props defaults to true?

Answer Option 1: When the prop is undefined

Answer Option 2: When the prop is null

Answer Option 3: When the prop is an empty string

Answer Option 4: When the prop is zero

Correct Response: 1

Explanation: In React, a component's props will default to true if the prop value is undefined. This can happen if the prop is not passed explicitly in the component declaration or if it is explicitly set to undefined in the parent component. To avoid this behavior, default values can be set for props using the defaultProps property in the component class.

[Reference: https://reactjs.org/docs/components-and-props.html#default-prop-values]

Question: Is it good to use arrow functions in render methods?

Answer Option 1: Yes, it improves performance

Answer Option 2: No, it can cause unnecessary re-renders

Answer Option 3: It depends on the use case

Answer Option 4: It has no effect on performance

Correct Response: 2

Explanation: It is not recommended to use arrow functions in render methods because it can cause unnecessary re-renders. Arrow functions are created each time the component renders, which can cause a new reference to be passed down to child components, triggering unnecessary re-renders.

[Reference: https://reactjs.org/docs/optimizing-performance.html#avoid-reconciliation]

Question: How to prevent a function from being called multiple times?

Answer Option 1: Use memoization

Answer Option 2: Use shouldComponentUpdate lifecycle method

Answer Option 3: Use componentDidUpdate lifecycle method

Answer Option 4: Use a callback ref

Correct Response: 1

Explanation: Memoization is a technique used to optimize functions by caching the results of expensive function calls and returning the cached result when the same inputs occur again. By using memoization, a function can be prevented from being called multiple times with the same inputs.

[Reference: https://reactjs.org/docs/optimizing-performance.html#use-memoization]

Question: How JSX prevents Injection Attacks?

Answer Option 1: By using string concatenation

Answer Option 2: By using a virtual DOM

Answer Option 3: By escaping special characters

Answer Option 4: By using JavaScript eval() function

Correct Response: 3

Explanation: JSX prevents injection attacks by escaping special characters. When JSX is compiled, special characters are automatically escaped, preventing them from being interpreted as code. This helps to prevent XSS (cross-site scripting) attacks.

[Reference: https://reactjs.org/docs/introducing-jsx.html#jsx-prevents-injection-attacks]

Question: How do you update rendered elements?

Answer Option 1: By calling setState()

Answer Option 2: By calling forceUpdate()

Answer Option 3: By manipulating the DOM directly

Answer Option 4: By re-rendering the entire application

Correct Response: 1

Explanation: In React, rendered elements are updated by calling the setState() method. When setState() is called, React re-renders the component and updates the UI accordingly. Manipulating the DOM directly is not recommended in React, as it can cause bugs and performance issues.

[Reference: https://reactjs.org/docs/state-and-lifecycle.html#using-state-correctly]

Question: How do you say that props are read-only?

Answer Option 1: By using Object.freeze()

Answer Option 2: By using Object.preventExtensions()

Answer Option 3: By using Object.seal()

Answer Option 4: By not modifying them directly

Correct Response: 4

Explanation: In React, props are read-only and should not be modified directly. Attempting to modify props directly can cause bugs and make components unpredictable. Instead, components should create and maintain their own state to manage changes to data.

[Reference: https://reactjs.org/docs/components-and-props.html#props-are-read-only]

Question: How do you say that state updates are merged?

Answer Option 1: By using setState()

Answer Option 2: By using forceUpdate()

Answer Option 3: By using Object.assign()

Answer Option 4: By using the spread operator

Correct Response: 4

Explanation: In React, state updates are merged with the existing state using the spread operator. When setState() is called, React updates the state object by merging the new state object with the existing state object. This allows components to update specific properties of the state object without overwriting the entire object.

[Reference: https://reactjs.org/docs/state-and-lifecycle.html#state-updates-are-merged]

Question: How do you pass arguments to an event handler?

Answer Option 1: By using the event object

Answer Option 2: By using arrow functions

Answer Option 3: By using the bind() method

Answer Option 4: By using the apply() method

Correct Response: 2

Explanation: In React, arguments can be passed to an event handler by using arrow functions. Arrow functions allow parameters to be passed to the function when it is called, rather than when it is defined. This allows event handlers to receive arguments without causing unnecessary re-renders.

[Reference: https://reactjs.org/docs/faq-functions.html#how-do-i-pass-a-parameter-to-an-event-handler-or-callback]

Question: How to prevent component from rendering?

Answer Option 1: By using the shouldComponentUpdate lifecycle method

Answer Option 2: By using the componentDidUpdate lifecycle method

Answer Option 3: By using the componentWillMount lifecycle method

Answer Option 4: By using the componentWillReceiveProps lifecycle method

Correct Response: 1

Explanation: In React, components can be prevented from rendering unnecessarily by using the shouldComponentUpdate() lifecycle method. By implementing this method, components can determine whether an update is necessary before rendering. This can help improve performance by minimizing the number of unnecessary re-renders.

[Reference: https://reactjs.org/docs/react-component.html#shouldcomponentupdate]

Question: What are the conditions to safely use the index as a key?

Answer Option 1: The list is static and will not change

Answer Option 2: The list is never reordered or filtered

Answer Option 3: The items in the list do not have unique IDs

Answer Option 4: The items in the list are of similar content and structure

Correct Response: 1, 2, 4

Explanation: Using the index as a key is only safe under certain conditions. If the list is static and will not change, the index can be used as a key. If the list is never reordered or filtered, the index can be used as a key. Additionally, if the items in the list are of similar content and structure, the index can be used as a key. If any of these conditions are not met, using the index as a key can cause errors and unpredictable behavior.

[Reference: https://reactjs.org/docs/lists-and-keys.html#keys]

Question: Is it keys should be globally unique?

Answer Option 1: Yes, always

Answer Option 2: No, never

Answer Option 3: It depends on the use case

Answer Option 4:

Correct Response: 1

Explanation: In React, keys should be globally unique whenever possible. This helps React identify which items have changed, added, or removed from a list, and update the UI accordingly. While keys do not have to be globally unique in all cases, it is generally a best practice to use unique keys whenever possible.

[Reference: https://reactjs.org/docs/lists-and-keys.html#keys]

Question: What is the popular choice for form handling?

Answer Option 1: React Forms

Answer Option 2: Redux Form

Answer Option 3: Formik

Answer Option 4: Unform

Correct Response: 3

Explanation: Formik is a popular library for handling forms in React. Formik provides a simple and flexible API for managing form state and validation, as well as handling submission and error messages. Formik is widely used in the React community and is considered a best practice for form handling.

[Reference: https://formik.org/docs/overview]

Question: What are the advantages of formik over redux form library?

Answer Option 1: Simpler API

Answer Option 2: Better performance

Answer Option 3: Easier integration with React

Answer Option 4: More flexible validation

Correct Response: 1, 3, 4

Explanation: Formik has several advantages over the Redux Form library, including a simpler API, easier integration with React, and more flexible validation options. Formik's API is designed to be simple and intuitive, making it easier to use and learn than Redux Form. Formik also integrates more seamlessly with React, with built-in support for React components and hooks. Finally, Formik provides a more flexible and extensible validation system, allowing developers to create custom validation rules and error messages.

[Reference: https://formik.org/docs/comparison]

Question: Why do you not require to use inheritance?

Answer Option 1: React components use composition

Answer Option 2: Inheritance is not supported in React

Answer Option 3: React components use mixins instead

Answer Option 4: Inheritance is deprecated in React

Correct Response: 1

Explanation: React components use composition rather than inheritance to build complex UI components. Composition allows components to be composed of multiple smaller components, each with their own logic and behavior. This makes components more reusable and easier to manage than inheritance-based approaches.

[Reference: https://reactjs.org/docs/composition-vs-inheritance.html]

Question: Can I use web components in React application?

Answer Option 1: Yes, but it requires additional configuration

Answer Option 2: No, React does not support web components

Answer Option 3: Yes, web components can be used directly in React

Answer Option 4: Yes, but it requires using a separate library

Correct Response: 1

Explanation: Web components can be used in a React application, but it requires additional configuration to enable support for custom elements and shadow DOM. React components can also be wrapped in web components, allowing them to be used in non-React contexts.

[Reference: https://reactjs.org/docs/web-components.html]

Question: What is dynamic import?

Answer Option 1: Importing code asynchronously at runtime

Answer Option 2: Importing code synchronously at compile time

Answer Option 3: Importing code using a CDN

Answer Option 4: Importing code from a different domain

Correct Response: 1

Explanation: Dynamic import is a feature of JavaScript that allows code to be imported asynchronously at runtime. This can help improve performance by allowing the application to load only the necessary code when it is needed, rather than loading everything upfront. Dynamic import is supported natively in modern browsers and can also be used with tools like Webpack.

[Reference: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/import#dynamic_imports]

Question: What are loadable components?

Answer Option 1: A component that loads data asynchronously

Answer Option 2: A component that loads other components asynchronously

Answer Option 3: A component that loads itself asynchronously

Answer Option 4: A component that loads CSS stylesheets asynchronously

Correct Response: 2

Explanation: Loadable components are a way to load other components asynchronously in React. Loadable components allow components to be split into smaller chunks and loaded on-demand, improving performance and reducing initial load times. Loadable components are typically used with dynamic imports to enable asynchronous loading of code.

[Reference: https://loadable-components.com/docs/getting-started/]

Question: What is suspense component?

Answer Option 1: A component for handling errors in React

Answer Option 2: A component for delaying rendering in React

Answer Option 3: A component for handling lazy loading in React

Answer Option 4: A component for handling forms in React

Correct Response: 2

Explanation: The suspense component is a component for delaying rendering in React. The suspense component allows components to wait for asynchronous data to load before rendering, improving the user experience and reducing loading times. The suspense component is typically used with code splitting and lazy loading to enable on-demand loading of code.

[Reference: https://reactjs.org/docs/react-api.html#reactsuspense]

Question: What is route based code splitting?

Answer Option 1: Splitting code based on component hierarchy

Answer Option 2: Splitting code based on component state

Answer Option 3: Splitting code based on component location

Answer Option 4: Splitting code based on component size

Correct Response: 3

Explanation: Route-based code splitting is a technique for splitting code based on the location of the component in the application. Route-based code splitting allows components to be loaded on-demand based on the user's navigation, reducing the initial load time of the application. Route-based code splitting is typically used with libraries like React Router to enable on-demand loading of code.

[Reference: https://reactjs.org/docs/code-splitting.html#route-based-code-splitting]

Question: What is the purpose of default value in context?

Answer Option 1: To provide a fallback value when a context value is not available

Answer Option 2: To override the context value in child components

Answer Option 3: To provide a default value for the context provider

Answer Option 4: To prevent child components from accessing the context value

Correct Response: 1

Explanation: The default value in context is used to provide a fallback value when a context value is not available. When a component consumes a context value, it looks for the context value in its ancestors. If no ancestor provides a value, the default value is used instead. The default value is typically used as a fallback or to provide a default value for the context.

[Reference: https://reactjs.org/docs/context.html#reactcreatecontext]

Question: How do you use contextType?

Answer Option 1: By passing the context value as a prop

Answer Option 2: By using the useContext() hook

Answer Option 3: By assigning the contextType property in the class definition

Answer Option 4: By creating a context consumer

Correct Response: 3

Explanation: The contextType property is used to consume a context value in a class component in React. To use contextType, you assign the context object to the contextType property in the class definition. This allows the component to access the context value using the this.context property. ContextType can only be used with a single context object and can only be used in class components.

[Reference: https://reactjs.org/docs/context.html#classcontexttype]

Question: What is a consumer?

Answer Option 1: A component that provides a context value

Answer Option 2: A component that consumes a context value

Answer Option 3: A component that renders children conditionally

Answer Option 4: A component that handles user events

Correct Response: 2

Explanation: In React context, a consumer is a component that consumes a context value. The consumer component allows child components to access the context value without the need to pass props explicitly. Consumers can be implemented using the useContext() hook or the Consumer component, which provides a render prop that can be used to consume the context value.

[Reference: https://reactjs.org/docs/context.html#contextconsumer]

Question: How do you solve performance corner cases while using context?

Answer Option 1: Memoize context values

Answer Option 2: Limit the scope of the context provider

Answer Option 3: Use shouldComponentUpdate to optimize updates

Answer Option 4: Use useContext() instead of Consumer

Correct Response: 1, 2, 3

Explanation: While using context in React, there are some performance corner cases that can arise, such as unnecessary re-renders and component updates. To solve these issues, context values can be memoized using the useMemo() hook, which can improve performance by preventing unnecessary re-computation. The scope of the context provider can also be limited to reduce the number of components that need to re-render. Additionally, shouldComponentUpdate can be used to optimize updates and prevent unnecessary renders.

[Reference: https://reactjs.org/docs/context.html#performance-optimization]

Question: What is the purpose of forward ref in HOCs?

Answer Option 1: To forward props to child components

Answer Option 2: To forward the ref to child components

Answer Option 3: To wrap child components with additional functionality

Answer Option 4: To wrap child components with a higher-order component

Correct Response: 2

Explanation: Forwarding refs in Higher-Order Components (HOCs) is a technique for passing a ref from a parent component to its child components. This allows the child components to access the DOM node or React element that the ref is attached to. To forward a ref in an HOC, the HOC component should use the forwardRef() method to create a new component that can receive a ref. The new component can then be used to wrap the child components and pass the ref through to the child components.

[Reference: https://reactjs.org/docs/forwarding-refs.html#forwarding-refs-in-higher-order-components]

Question: Is the ref argument available for all functions or class components?

Answer Option 1: Yes, for all components

Answer Option 2: No, only for class components

Answer Option 3: No, only for function components

Answer Option 4:

Correct Response: 2

Explanation: The ref argument is only available for class components in React. Function components do not have an instance, so refs cannot be attached to them. Refs can be attached to DOM elements, class components, and functional components that are created using the forwardRef() method.

[Reference: https://reactjs.org/docs/refs-and-the-dom.html#adding-a-ref-to-a-class-component]

Question: Why do you need additional care for component libraries while using forward refs?

Answer Option 1: Refs can break encapsulation

Answer Option 2: Refs can cause memory leaks

Answer Option 3: Refs can cause conflicts with other libraries

Answer Option 4: Refs can cause compatibility issues with future React versions

Correct Response: 1, 3

Explanation: When using forward refs with component libraries in React, additional care is needed to ensure that the refs do not break encapsulation and cause conflicts with other libraries. Refs can expose the internal implementation details of a component, which can lead to issues with library compatibility and maintainability. Additionally, when using third-party libraries, it is important to ensure that the library is compatible with the latest version of React and that any potential issues are addressed.

[Reference: https://reactjs.org/docs/forwarding-refs.html#forwarding-refs-to-dom-components]

Question: How to create React class components without ES6?

Answer Option 1: By using create-react-class method

Answer Option 2: By using functional components

Answer Option 3: By using the React.Component class directly

Answer Option 4:

Correct Response: 1

Explanation: The create-react-class method is a way to create React class components without using ES6 syntax. This method allows components to be created using a simple object-based syntax, rather than defining a class and extending the React.Component class. The create-react-class method is useful for creating components that do not require complex logic or state management.

[Reference: https://reactjs.org/docs/create-a-new-react-app.html#create-react-app]

Question: Is it possible to use React without JSX?

Answer Option 1: Yes, JSX is optional in React

Answer Option 2: No, JSX is required in React

Answer Option 3:

Answer Option 4:

Correct Response: 1

Explanation: JSX is a syntax extension for JavaScript that allows developers to write HTML-like syntax in their JavaScript code. While JSX is the preferred way to write React components, it is not strictly required. React components can also be written using plain JavaScript syntax, although this can be more verbose and difficult to read. However, JSX is widely used in the React community and is considered a best practice for writing React components.

[Reference: https://reactjs.org/docs/react-without-jsx.html]

Question: What is the diffing algorithm?

Answer Option 1: A process for comparing two React elements

Answer Option 2: A process for reconciling changes in the React component tree

Answer Option 3: A process for optimizing React component rendering

Answer Option 4: A process for testing React components

Correct Response: 2

Explanation: The diffing algorithm is the process used by React to reconcile changes in the component tree and update the DOM. The diffing algorithm compares the new tree of React elements with the previous tree and identifies the minimum set of changes needed to update the DOM. This process is also known as reconciliation and is a key part of React's performance optimizations.

[Reference: https://reactjs.org/docs/reconciliation.html]

Question: What are the rules covered by the diffing algorithm?

Answer Option 1: Components of different types should produce different trees

Answer Option 2: The same component should behave the same way for the same props

Answer Option 3: Only update the changed attributes

Answer Option 4: Keys should be unique among siblings

Correct Response: 1, 2, 4

Explanation: The diffing algorithm in React is governed by several rules, including that components of different types should produce different trees, the same component should behave the same way for the same props, and keys should be unique among siblings. These rules help to ensure that the minimum set of changes are made to the component tree and that performance is optimized.

[Reference: https://reactjs.org/docs/reconciliation.html#recursing-on-children]

Question: When do you need to use refs?

Answer Option 1: To access the DOM node of a component

Answer Option 2: To pass data between sibling components

Answer Option 3: To manage component state

Answer Option 4: To update component props

Correct Response: 1

Explanation: Refs in React are used to access the DOM node of a component. Refs can be attached to any component, including functional components, and allow developers to access the underlying DOM node or React element. Refs are typically used to access the value of an input or textarea element, or to attach event listeners to a DOM node.

[Reference: https://reactjs.org/docs/refs-and-the-dom.html#accessing-refs]

Question: Is it a must that the prop must be named as "render" for render props?

Answer Option 1: Yes, it is a strict requirement

Answer Option 2: No, it can be named anything

Answer Option 3:

Answer Option 4:

Correct Response: 2

Explanation: Render props in React are a technique for sharing code between components. A render prop is a function that a component uses to render its children, allowing the children to access the component's state or other data. The render prop can be named anything, and the name of the prop does not affect its functionality. The convention of naming the prop "render" is simply a convention and is not required by React.

[Reference: https://reactjs.org/docs/render-props.html]

Question: What are the problems of using render props with pure components?

Answer Option 1: The component can re-render unnecessarily

Answer Option 2: The component can lose performance benefits

Answer Option 3: The component can lose the ability to shallowly compare props

Answer Option 4: The component can become difficult to maintain

Correct Response: 1, 2, 3

Explanation: Render props are a popular technique in React for sharing code between components. However, when used with pure components, render props can cause issues with unnecessary re-renders, lost performance benefits, and lost ability to shallowly compare props. This is because pure components use a shallow comparison of their props and state to determine whether to re-render, and the use of a render prop can cause the pure component to re-render unnecessarily.

[Reference: https://kentcdodds.com/blog/dont-use-render-props-with-pure-component]

Question: How do you create HOC using render props?

Answer Option 1: By passing a function as a prop to the higher-order component

Answer Option 2: By returning a new component from the higher-order component

Answer Option 3: By using the withRenderProp() method

Answer Option 4: By using the createRenderProp() method

Correct Response: 2

Explanation: Higher-Order Components (HOCs) are a powerful pattern in React that allow developers to reuse code between components. HOCs can also be implemented using render props, which involves returning a new component from the higher-order component that renders the children using a function prop. This pattern is known as "render prop HOCs" and is a flexible and powerful way to share code between components.

[Reference: https://reactjs.org/docs/higher-order-components.html#convention-wrap-the-display-name-for-easy-debugging]

Question: What is windowing technique?

Answer Option 1: A technique for manipulating the browser window

Answer Option 2: A technique for optimizing performance by rendering only a subset of a large data set

Answer Option 3: A technique for managing React state

Answer Option 4: A technique for handling asynchronous data fetching

Correct Response: 2

Explanation: Windowing is a technique used in React to optimize performance when rendering a large data set. Windowing involves rendering only a subset of the data that is currently visible on the screen, and dynamically rendering additional data as the user scrolls or interacts with the UI. This technique can significantly improve the performance of applications that need to render large amounts of data, such as tables, lists, and grids.

[Reference: https://react-window.vercel.app/]

Question: What is the typical use case of portals?

Answer Option 1: For rendering large lists

Answer Option 2: For creating modals and tooltips

Answer Option 3: For building layouts and grids

Answer Option 4: For handling state in complex components

Correct Response: 2

Explanation: Portals in React are a way to render a child component outside of its parent component. This is useful in cases where you need to render a child component in a different part of the DOM hierarchy, such as when creating modals or tooltips.

[Reference: https://reactjs.org/docs/portals.html]

Question: How do you set a default value for an uncontrolled component?

Answer Option 1: Use the value attribute

Answer Option 2: Use the defaultValue attribute

Answer Option 3: Use the setState() method

Answer Option 4: Use the props attribute

Correct Response: 2

Explanation: In React, uncontrolled components are form components that store their own state internally. To set a default value for an uncontrolled component, you can use the defaultValue attribute. This will set the initial value of the component when it first renders.

[Reference: https://reactjs.org/docs/uncontrolled-components.html#default-values]

Question: What is your favorite React stack?

Answer Option 1: MERN (MongoDB, Express, React, Node)

Answer Option 2: MEAN (MongoDB, Express, Angular, Node)

Answer Option 3: MEVN (MongoDB, Express, Vue, Node)

Answer Option 4: Other

Correct Response: 4

Explanation: This is a subjective question and the correct answer will vary depending on the individual's experience and preferences. There is no one "correct" answer.

Question: What is the difference between Real DOM and Virtual DOM?

Answer Option 1: Real DOM is faster than Virtual DOM

Answer Option 2: Virtual DOM is faster than Real DOM

Answer Option 3: Real DOM is a physical representation of the web page, while Virtual DOM is a virtual representation

Answer Option 4: Virtual DOM is a physical representation of the web page, while Real DOM is a virtual representation

Correct Response: 3

Explanation: The Real DOM is a physical representation of the web page, consisting of all the HTML elements, their attributes, and their relationships with each other. The Virtual DOM, on the other hand, is a lightweight JavaScript representation of the Real DOM. The Virtual DOM allows React to update the UI efficiently by minimizing the number of updates needed to the Real DOM.

[Reference: https://reactjs.org/docs/faq-internals.html#what-is-the-virtual-dom]

Question: How to add Bootstrap to a React application?

Answer Option 1: Install the Bootstrap npm package and import it into your components

Answer Option 2: Link to the Bootstrap CSS and JavaScript files in your HTML file

Answer Option 3: Use a third-party React Bootstrap library

Answer Option 4: All of the above

Correct Response: 4

Explanation: There are several ways to add Bootstrap to a React application, including installing the Bootstrap npm package and importing it into your components, linking to the Bootstrap CSS and JavaScript files in your HTML file, and using a third-party React Bootstrap library.

[Reference: https://getbootstrap.com/docs/5.1/getting-started/introduction/]

Question: Can you list down top websites or applications using React as front end framework?

Answer Option 1: Airbnb, Dropbox, and Salesforce

Answer Option 2: Facebook, Instagram, and Netflix

Answer Option 3: Google, Amazon, and Microsoft

Answer Option 4: Twitter, LinkedIn, and PayPal

Correct Response: 2

Explanation: React is widely used by many popular websites and applications, including Facebook, Instagram, and Netflix. Other companies that use React include Airbnb, Dropbox, and Salesforce.

[Reference: https://reactjs.org/community/companies.html]

Question: Is it recommended to use CSS In JS technique in React?

Answer Option 1: Yes, it is a best practice

Answer Option 2: No, it is not recommended

Answer Option 3: It depends on the project requirements

Answer Option 4:

Correct Response: 3

Explanation: The decision to use CSS in JS is largely dependent on the specific project requirements and the preferences of the development team. CSS in JS can offer certain benefits, such as better modularity and encapsulation of styles, but it may not be the best choice for every project.

[Reference: https://reactjs.org/docs/faq-styling.html#what-is-css-in-js]

Question: Do I need to rewrite all my class components with hooks?

Answer Option 1: Yes, it is recommended to use hooks over class components

Answer Option 2: No, you can continue to use class components if they are working for your project

Answer Option 3: It depends on the project requirements

Answer Option 4:

Correct Response: 2

Explanation: React Hooks were introduced to provide an alternative way of managing state and lifecycle methods in functional components. However, class components are still fully supported in React, and you can continue to use them if they are working for your project.

[Reference: https://reactjs.org/docs/hooks-faq.html#should-i-use-hooks-migrating-from-classes]

Question: How to fetch data with React Hooks?

Answer Option 1: Use the componentDidMount() method

Answer Option 2: Use the fetch() function in a useEffect() hook

Answer Option 3: Use the componentWillMount() method

Answer Option 4: Use the AJAX library

Correct Response: 2

Explanation: In React, you can use the useEffect() hook to fetch data from an API. Inside the useEffect() hook, you can use the fetch() function to make a request to the API and update the component state with the response data.

[Reference: https://reactjs.org/docs/hooks-effect.html]

Question: Is Hooks cover all use cases for classes?

Answer Option 1: Yes, Hooks cover all use cases for classes

Answer Option 2: No, there are some use cases where classes may be more appropriate

Answer Option 3:

Answer Option 4:

Correct Response: 2

Explanation: While Hooks provide an alternative way of managing state and lifecycle methods in functional components, there may be some use cases where classes are more appropriate. For example, classes can be used to implement certain patterns, such as higher-order components (HOCs) and render props, which may not be as straightforward to implement with Hooks.

[Reference: https://reactjs.org/docs/hooks-faq.html#do-hooks-cover-all-use-cases-for-classes]

Question: What is the stable release for hooks support?

Answer Option 1: React 15

Answer Option 2: React 16

Answer Option 3: React 17

Answer Option 4:

Correct Response: 3

Explanation: React Hooks were first introduced in React 16.8, and have been a stable feature of React since React 16.8.0. React 17 is the most recent stable release of React as of September 2021.

[Reference: https://reactjs.org/docs/hooks-intro.html]

Question: Why do we use array destructuring (square brackets notation) in useState?

Answer Option 1: It's a personal preference of the developer

Answer Option 2: It's required by the React API

Answer Option 3: It's a cleaner way to write the code

Answer Option 4: It allows us to name the state variables

Correct Response: 4

Explanation: When using useState in React, the function returns an array with two elements: the state value and a function to update the state value. By using array destructuring (square brackets notation), we can name the state variables to make our code more readable and easier to maintain.

[Reference: https://reactjs.org/docs/hooks-state.html#declaring-a-state-variable]

Question: What are the sources used for introducing hooks?

Answer Option 1: Community proposals and discussions

Answer Option 2: Official React documentation and blog posts

Answer Option 3: React conferences and meetups

Answer Option 4: All of the above

Correct Response: 4

Explanation: React Hooks were introduced based on community proposals and discussions, as well as official React documentation and blog posts. They were also presented and discussed at React conferences and meetups.

[Reference: https://reactjs.org/docs/hooks-intro.html#motivation]

Question: How do you access the imperative API of web components?

Answer Option 1: Use the refs API in React

Answer Option 2: Use the useEffect() hook in React

Answer Option 3: Use the setState() method in React

Answer Option 4: Use the createContext() API in React

Correct Response: 1

Explanation: In React, you can use the refs API to access the imperative API of web components. By using refs, you can reference a web component and then access its imperative API methods and properties.

[Reference: https://reactjs.org/docs/integrating-with-other-libraries.html#using-third-party-dom-libraries]

Question: What is Formik?

Answer Option 1: A React library for working with forms

Answer Option 2: A testing library for React

Answer Option 3: A utility library for working with arrays

Answer Option 4: An animation library for React

Correct Response: 1

Explanation: Formik is a utility library for working with forms in React. It provides a simple and flexible way to handle form validation, input masking, and submission handling. Formik also integrates with other popular form libraries, such as Yup and React-Select.

[Reference: https://formik.org/docs/overview]

Question: Do browsers understand JSX code?

Answer Option 1: Yes

Answer Option 2: No

Answer Option 3:

Answer Option 4:

Correct Response: 2

Explanation: Browsers do not understand JSX code, as it is not valid JavaScript syntax. JSX must be transpiled into regular JavaScript code using a tool such as Babel before it can be understood by browsers.

[Reference: https://reactjs.org/docs/introducing-jsx.html]

Question: Describe data flow in React?

Answer Option 1: Unidirectional data flow

Answer Option 2: Bidirectional data flow

Answer Option 3: Multidirectional data flow

Answer Option 4:

Correct Response: 1

Explanation: Data flow in React is unidirectional, meaning that data flows in a single direction from the parent component to the child component. This helps ensure that the application state is predictable and easier to manage.

[Reference: https://reactjs.org/docs/state-and-lifecycle.html#the-data-flows-down]

Question: What is react-scripts?

Answer Option 1: A build tool for React applications

Answer Option 2: A testing framework for React applications

Answer Option 3: A code analysis tool for React applications

Answer Option 4: A runtime environment for React applications

Correct Response: 1

Explanation: react-scripts is a build tool that comes bundled with Create React App, a popular tool for creating React applications. react-scripts provides a set of preconfigured build scripts and development tools, such as Webpack and Babel, to simplify the process of setting up and running a React application.

[Reference: https://create-react-app.dev/docs/available-scripts/]

Question: of renderToNodeStream method?

Answer Option 1: To render React components on the client-side

Answer Option 2: To render React components on the server-side

Answer Option 3: To render React components in a web worker

Answer Option 4: To render React components in an iframe

Correct Response: 2

Explanation: The renderToNodeStream method is used to render React components on the server-side. It returns a Node.js stream, which allows for efficient server-side rendering of large amounts of data. This method is useful for improving the performance of server-side rendering in large applications.

[Reference: https://reactjs.org/docs/react-dom-server.html#rendertonodestream]

Question: What is MobX?

Answer Option 1: A state management library for React

Answer Option 2: A testing framework for React

Answer Option 3: A build tool for React

Answer Option 4: A UI component library for React

Correct Response: 1

Explanation: MobX is a state management library for React. It allows developers to manage application state in a simple and efficient way, using observables and actions. Observables are objects that can be observed for changes, while actions are functions that modify observables. MobX is often used in combination with React, but it can also be used with other UI libraries.

[Reference: https://mobx.js.org/README.html]

Question: Should I learn ES6 before learning ReactJS?

Answer Option 1: Yes, because ES6 is a prerequisite for ReactJS

Answer Option 2: Yes, because ES6 features are commonly used in ReactJS

Answer Option 3: No, because ES6 is not required for ReactJS

Answer Option 4: No, because ES6 is not relevant to ReactJS

Correct Response: 2

Explanation: While it's not strictly necessary to learn ES6 before learning ReactJS, it is recommended because many ES6 features are commonly used in ReactJS development. Some of these features include arrow functions, template literals, and destructuring assignments. Additionally, ReactJS documentation often assumes knowledge of ES6 syntax.

[Reference: https://reactjs.org/docs/javascript-environment-requirements.html]

Question: What is Concurrent Rendering?

Answer Option 1: A rendering technique for low-end devices

Answer Option 2: A rendering technique for high-end devices

Answer Option 3: A new feature in React 18

Answer Option 4: A way to render multiple parts of a component tree at the same time

Correct Response: 4

Explanation: Concurrent Rendering is a new rendering strategy introduced in React 18 that allows multiple parts of a component tree to be rendered at the same time, without blocking the UI thread. This means that the user interface can remain responsive while React is rendering updates. Concurrent Rendering is particularly useful for large and complex applications that need to maintain high performance.

[Reference: https://reactjs.org/docs/concurrent-mode-intro.html]

Question: What is the difference between async mode and concurrent mode?

Answer Option 1: Async mode is used for long-running tasks, while concurrent mode is used for rendering

Answer Option 2: Async mode is blocking, while concurrent mode is non-blocking

Answer Option 3: Async mode can lead to slow UI updates, while concurrent mode can lead to faster UI updates

Answer Option 4: Async mode can cause jank, while concurrent mode can prevent jank

Correct Response: 3, 4

Explanation: Async mode and concurrent mode are both ways to improve the performance of React applications, but they have different characteristics. Async mode is used for long-running tasks, such as data fetching or image processing, that can block the UI thread and slow down the application. Concurrent mode, on the other hand, is a rendering mode that allows React to update parts of the UI without blocking the UI thread, which can prevent jank and improve the perceived performance of the application.

[Reference: https://reactjs.org/docs/concurrent-mode-intro.html]

Question: Can I use javascript urls in react16.9?

Answer Option 1: Yes

Answer Option 2: No

Answer Option 3:

Answer Option 4:

Correct Response: 2

Explanation: JavaScript URLs are not allowed in React16.9 or any modern web development framework due to security concerns. JavaScript URLs are URLs that begin with "javascript:", and they allow for the execution of arbitrary JavaScript code when clicked. This can be used for malicious purposes, such as stealing user data or injecting malware. Instead of using JavaScript URLs, developers should use event handlers and other safe mechanisms to handle user interactions.

[Reference: https://stackoverflow.com/questions/63375722/is-it-possible-to-use-javascript-url-in-href-tag-in-reactjs]

Question: What is the purpose of eslint plugin for hooks?

Answer Option 1: To enforce best practices for React component development

Answer Option 2: To check for errors in React hooks usage

Answer Option 3: To improve the performance of React hooks

Answer Option 4: To add support for new React features

Correct Response: 2

Explanation: The eslint-plugin-react-hooks is a plugin for the eslint linter that helps developers avoid common issues with React hooks, such as missing dependencies or incorrect usage. It checks for errors and enforces best practices for using React hooks, which can improve code quality and reduce bugs.

[Reference: https://www.npmjs.com/package/eslint-plugin-react-hooks]

Question: What is the difference between Imperative and Declarative in React?

Answer Option 1: Imperative is more performant than declarative

Answer Option 2: Declarative is more concise than imperative

Answer Option 3: Imperative is easier to read than declarative

Answer Option 4: Declarative is easier to reason about than imperative

Correct Response: 4

Explanation: In React, Imperative and Declarative are two different approaches to building user interfaces. Imperative programming involves giving explicit instructions on how to accomplish a task, while declarative programming involves describing what the outcome should be without specifying how to achieve it. React uses a declarative approach, which means that developers describe what the user interface should look like, and React takes care of the details of how to render it. This makes it easier to reason about and maintain complex UIs.

[Reference: https://reactjs.org/docs/faq-programming.html#what-is-the-difference-between-imperative-and-declarative-in-react]

Question: What are the benefits of using typescript with reactjs?

Answer Option 1: Better type checking

Answer Option 2: Improved code readability

Answer Option 3: Enhanced editor support

Answer Option 4: Increased performance

Correct Response: 1, 2, 3

Explanation: TypeScript is a typed superset of JavaScript that adds optional static type checking to the language. When used with React, TypeScript offers several benefits, including better type checking, improved code readability, and enhanced editor support. Better type checking helps catch errors early in the development process, reducing bugs and improving code quality. Improved code readability makes it easier to understand and maintain the codebase, while enhanced editor support improves developer productivity. However, TypeScript does not provide any performance benefits over regular JavaScript.

[Reference: https://www.sitepoint.com/react-with-typescript-best-practices/]

Question: How do you make sure that user remains authenticated on page refresh while using Context API State Management?

Answer Option 1: Use cookies to store the authentication token

Answer Option 2: Store the authentication token in local storage

Answer Option 3: Use session storage to store the authentication token

Answer Option 4: Use a server-side session to store the authentication token

Correct Response: 2

Explanation: When using the Context API for state management in a React application, it is important to ensure that the user remains authenticated even if the page is refreshed. One way to do this is to store the authentication token in local storage, which persists even after the page is refreshed. This allows the application to retrieve the authentication token from local storage and use it to authenticate the user without requiring them to log in again.

[Reference: https://stackoverflow.com/questions/54175654/react-context-how-to-stay-authenticated-after-page-refresh]