Mentorow
Talent Accelerator

REACT INTERVIEW QUESTIONS

Beginner to Advanced level practical React.js Questions

WHAT WILL BE THE OUTPUT OF THE BELOW CODE IF THE BUTTON IS CLICKED

sujith Kumar R

```
function App() {
  const [count, setCount] = useState(0);
 useEffect(() => {
        console.log("Component rendered successfully");
  }, []);
   return (
        <div>
        <button onClick={() => setCount(count + 1)}>Click
me</button>
        <p>You clicked {count} times</p>
        </div>
  );
}
```

FIND THE ISSUE IN THE BELOW CODE SNIPPET AFTER RENDERING THE LIST OF NAMES

sujith Kumar R

```
 import React from "react";
 function App() {
  const names = ["Brian", "Paul", "Krug", "Halley"];
   const listItems = names.map((name) => <li>{name}</li>);
   return <ul>{listItems}</ul>;
}
export default App;
Question 3
Analyze the below code and advise what is wrong with using
setState() inside the render() method:
import React, { Component } from "react";
 class App extends Component {
  state = {
        counter: 0,
  };

  render() {
        this.setState({ counter: this.state.counter + 1 });
        return <div>Counter: {this.state.counter}</div>;
  }
}
export default App;
```

WHAT ISSUE EXISTS IN THE BELOW CODE REGARDING STATE VARIABLE

sujith Kumar R

```
import React, { useState } from "react";
 function App() {
  const [counter, setCounter] = useState(0);
  function incrementCounter() {
        setCounter(counter + 1);
  }
   return (
        <div>
        <button onClick={incrementCounter}>Increment</button>
        <p>Counter: 0</p>
        </div>
  );
}

export default App;
```

SEE THE BELOW CODE SNIPPET AND ADVISE, WILL THERE BE ANY ISSUE MAKING A REST API CALL IN A COMPONENT'S USEEFFECT HOOK?

sujith Kumar R

```
import { useState } from "react";
import axios from "axios";
function MyComponent() {
  const [data, setData] = useState([]);
 useEffect(() => {
    axios.get("/api/data").then((response) => {
      setData(response.data);
    });
  }, []);

  return <div>{data.map((d) => <p>{d.text}</p>)}</div>;
}
```

WHAT ARE HIGHER-ORDER COMPONENTS (HOCS)?

sujith Kumar R

Higher-Order Components (HOCs) are a pattern in React for reusing component logic. HOCs are functions that take a component as an argument and return a new component with additional props or behavior.

Example:

```
function withLogging(WrappedComponent) {
  return function LoggingComponent(props) {
    console.log('Props:', props);
    return <WrappedComponent {...props} />;
  };
}
```

In this example, the withLogging function takes a WrappedComponent as an argument and returns a new LoggingComponent that logs the props and renders the WrappedComponent with the same props.

To use the HOC, you can wrap your component like this

```
import React from 'react';
import withLogging from './withLogging';

function MyComponent(props) {
  return <p>Hello, {props.name}!</p>;
}

export default withLogging(MyComponent);
```

This will create a new component that includes the logging behavior, and you can use it just like the original component.

WHAT IS USEMEMO, AND WHEN SHOULD YOU USE IT?

sujith Kumar R

useMemo is a hook in React that allows you to memoize the result of a function so that it doesn't get recomputed on every render of the component. This can be useful when you want to optimize the performance of your application by preventing unnecessary calculations or data transformations

The useMemo hook takes two arguments: a function that computes the memoized value and an array of dependencies. The hook returns the memoized value, which will only be recomputed if any of the dependencies change.

```
import React, { useMemo, useState } from 'react';

function App() {
  const [count, setCount] = useState(0);

  const expensiveComputation = (value) => {
    console.log('Expensive computation');
    return value * 10;
  };

  const memoizedResult = useMemo(() => expensiveComputation(count), [count]);

  const handleClick = () => {
    setCount((prevCount) => prevCount + 1);
  };

  return (
    <div>
      <button onClick={handleClick}>Increase count</button>
      <p>Count: {count}</p>
      <p>Memoized result: {memoizedResult}</p>
    </div>
  );
}

export default App;
```

BUILD A USER REGISTRATION FORM

sujith Kumar R

Create a user registration form with fields for username, email, and password. Validate inputs and display appropriate error messages

Tasks

- Implement the user registration form UI
- Validate the form inputs and display error messages
- Integrate form submission with a mock API call (simulated delay).
- Add a success message upon successful registration

Hints

- Create a controlled form component with state to manage input values.
- Use regular expressions or a library like Yup for input validation.

CREATE A RESPONSIVE NAVBAR

Design a responsive navigation bar with options for home, about, services, and contact. Ensure it adapts well to different screen sizes.

Tasks

- Implement the responsive navigation bar UI.
- Ensure it collapses into a hamburger menu on smaller screens.
- Add functionality to navigate between pages.

Hints

- Use CSS flexbox or grid for layout.
- Utilize React Router for page navigation.

TODO LIST WITH LOCAL STORAGE

Build a to-do list that allows users to add, edit, and remove tasks. Persist the tasks using local storage

Tasks

- Create the todo list UI with options to add, edit, and delete tasks.
- Implement local storage to persist tasks across page refreshes.
- Add functionality to mark tasks as completed.

Hints

- Use React state to manage the list of tasks.
- Utilize the local Storage API to store and retrieve tasks.

E-COMMERCE PRODUCT FILTER

sujith Kumar R

Build a product filtering component that allows users to filter products by categories, price range, and ratings

Tasks

- Implement the product filter UI.
- Add functionality to filter products based on selected criteria.
- Update the product list dynamically.

Hints

- Use React state to manage the selected filter criteria.
- Implement filtering logic based on the selected criteria.

.QUIZ APP WITH TIMER

Create a quiz application with a timer that counts down as users answer questions. Display results at the end

Tasks

- Implement the quiz application UI with questions and multiple-choice options.
- Add a countdown timer that starts when the quiz begins.
- Calculate and display results at the end

Hints

- Use React state to manage the quiz state and timer.
- Use JavaScript's setInterval for timer functionality.