Final Quiz

Question 1

You are building a form using both Formik and Yup libraries, where one of the inputs is an email. Here are this input’s client validation rules:

* It has to be a valid email address.
* If the email input is invalid, a message “Invalid email address” will be displayed.
* If the email input is blank, a message “Required” will be shown.

Based on the above requirements, choose the correct Yup validation code from the provided options.

1 / 1 point



1

Yup.email().string("Invalid email address").required("Required")







1

Yup.email("Invalid email address").required("Required").







1

Yup.string().email("Invalid email address").required("Required")





Correct

Correct, first Yup needs to know the type of value (string) and then chain the different validation rules with their associated error message to show.

**2.**

Question 2

You have the following React application where you have a **ToDo** component that has two text labels and an uncontrolled text input and the entry point App component that renders a list of two ToDos and a button to reverse the order of the ToDos. To avoid a React keys warning, a key is provided to each **ToDo** component, with the index as its value. Suppose that the next sequence of events happen in the application:

1. You write “Wash dishes” in the first **ToDo** input

2. You write “Buy groceries” in the second **ToDo** input

3. You click the button to reverse the order

What would happen on the screen after that?

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

  const reverseOrder = () => {

    // Reverse is a mutative operation, so we need to create a new array first.

    setTodos([...todos].reverse());

  };

  return (

    <div>

      <button onClick={reverseOrder}>Reverse</button>

      {todos.map((todo, index) => (

        <ToDo key={index} id={todo.id} createdAt={todo.createdAt} />

      ))}

    </div>

  );

}





1 / 1 point



todo2 Buy groceries 20:30

todo1 Wash dishes 18:00



todo2 Wash dishes 20:30

todo1 Buy groceries 18:00



todo1 Buy groceries 18:00

todo2 Wash dishes 20:30

Correct

Correct, when reversing the order React understands they are still the same nodes with key=1 and key=2, so it will preserve their internal state (input value). Since the props are different though, it will just update the node with the new prop values.

**3.**

Question 3

Consider the code below, and choose the correct sentence about this code.

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

import{ createContext, useContext, useState} from"react";

const ThemeContext = createContext(undefined);

export const ThemeProvider= () => {

  const[theme, setTheme] = useState("light");

  return(

    <ThemeContext.Provider

      value={{

        theme,

        toggleTheme: () => setTheme(!theme),

      }}

    >

    </ThemeContext.Provider>

  );

};





1 / 1 point



This code has one or more errors in it, and thus needs to be fixed.



This code doesn’t have an error and can be ran as is, without errors.

Correct

Correct, there are two errors in this code. First, the **toggleTheme** implementation is incorrect and should be: **toggleTheme: () =>setTheme(theme === "light" ? "dark" : "light")**. Second, **ThemeProvider** should use the **children** prop and pass it as a direct child of **ThemeContext.Provider**.

**4.**

Question 4

True or False: The type of a React element can be a DOM node, such as, for example, an HTML button.

1 / 1 point



True

Correct

Correct, the type can be a DOM node.



False.

**5.**

Question 5

Assuming you have the following set of components, what would be logged into the console when clicking the Submit button that gets rendered on the screen?

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

const Button = ({ children, ...rest }) => (

  <button onClick={() => console.log("ButtonClick")} {...rest}>

    {children}

  </button>

);

const withClick = (Component) => {

  const handleClick = () => {

    console.log("WithClick");

  };

  return(props) => {

    return<Component {...props} onClick={handleClick} />;

  };

};

const MyButton = withClick(Button);

export default function App() {

  return <MyButton onClick={() => console.log("AppClick")}>Submit</MyButton>;

}





1 / 1 point



“ButtonClick”.



“WithClick”



“AppClick”

Correct

Correct, due to the order of the spread operator in the different components, the **withClick** higher Order Component (HOC) takes precedence.

**6.**

Question 6

True or False: Using jest and react-testing-library, to assert that a function has been called with some specific arguments, you would need to use the **toHaveAttribute** matcher.

1 / 1 point



True.



False.

Correct

Correct, the **toHaveAttribute** is not the proper matcher to check the arguments of the function call. Instead the **toHaveBeenCalledWith** should be used.

**7.**

Question 7

Is the following piece of code a valid implementation of the render props pattern?

1

2

3

<MealProvider render={data => (

  <p>Ingredients: {data.ingredients}</p>

)}/>





1 / 1 point



Yes



No

Correct

Correct, it uses a render type prop that is a function that returns JSX.

**8.**

Question 8

Inspect the given code snippet.

1

2

3

React.useEffect(()=> {

 console.log('The initial value of the price variable is', price)

})





Where should you add an empty array to have the effect ran only on initial render?

1 / 1 point



As a second argument after the arrow function passed to the useEffect() call.



You need to add an empty array in a separate arrow function.



You can't add an empty array in this code snippet.

Correct

Correct. You need to add it as a second argument after the arrow function passed to the **useEffect()** call.

**9.**

Question 9

True or false? In the following component, the **setRestaurantName** variable’s value will not be reset between re-renders of the App component.

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

import {useState} from "react";

export default function App() {

  const [restaurantName, setRestaurantName] = useState("Lemon");

  function updateRestaurantName() {

    setRestaurantName("Little Lemon");

  };

  return (

    <div>

      <h1>{restaurantName}</h1>

      <button onClick={updateRestaurantName}>

        Update restaurant name

      </button>

    </div>

  );

};





1 / 1 point



True



False

Correct

Correct. The **setRestaurantName** variable's value will not be reset between re-renders of the App component.

**10.**

Question 10

Is the following code snippet valid? Why?

1

2

3

4

5

if (data !== '') {

  useEffect(() => {

    setData('test data');

  });

}





1 / 1 point



It's not valid, because it's breaking the rules of hooks.



It's valid, because it's not breaking the rules of hooks.



It's valid, because you can use the useEffect() call in an if statement.

Correct

Correct. If you use a hook in a condition, you're breaking rules! Thus, the below code is invalid.