Congratulations! You passed!

Grade received 97.50% Latest Submission Grade 97.50% To pass 80% or higher

Retake the assignment in 1h 40m

Go to next item

1. You are building a form using both Formik and Yup libraries, where one of the inputs is an email. Before the form gets submitted to the server, you would like to set up some client validation with Yup to make sure the field has an email that is valid, otherwise a message "Invalid email address" would be shown on the screen. This field is also required. Choose the correct validation code from the three code snippets.

1 / 1 point

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

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

```
Yup.email().string("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. 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 / 1 point

1. You write "Wash the car" in the first `ToDo` input

2. You write "Buy bumper stickers" in the second `ToDo` input

3. You click the button to reverse the order

What would happen on the screen after that?

```
const ToDo = props => (
  <tr>
    <td>
      <label>{props.id}</label>
    </td>
    <td>
      <input />
    </td>
    <td>
      <label>{props.createdAt}</label>
    </td>
  </tr>
);


function App() {
  const [todos, setTodos] = useState([
    {
      id: 'todo1',
      createdAt: '18:00',
    },
    {
      id: 'todo2',
      createdAt: '20:30',
    }
  ]);

  const reverseOrder = () => {
    // Reverse is a mutative operation, so we need to create a new array first
```

```
    // 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>
  );
```

- ○ todo1 Buy bumper stickers 18:00

  todo2 Wash the car 20:30
- ○ todo2 Buy bumper stickers 20:30

  todo1 Wash the car 18:00
- ◉ todo2 Wash the car 20:30

  todo1 Buy bumper stickers 18:00

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. True or false: There are at least two errors in the code below. **1 / 1 point**

```
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>
  );
};
```

- ◉ True
- ○ False

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. Select all the statements that are true for React elements: **0.75 / 1 point**

- ☑ A tree of elements can mix and match both components and DOM elements as the type property.

Correct
Correct, they can be mixed and matched.

- ☐ The type of an element can be a function corresponding to a React component, like a `SubmitButton`.
- ☐ Each element object should have at least two properties: type and children
- ☑ The type of an element can be a DOM node, like a HTML button.

Correct
Correct, the type can be a DOM node.

You didn't select all the correct answers

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 / 1 point**

```
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>;
}
```

- ○ "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. **1 / 1 point**

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 `toHaveBeenCalledWith` matcher.

- ○ False.
- ◉ True.

Correct
Correct, this is the proper matcher to check the arguments of the function call.

7. Is the following piece of code a valid implementation of the render props pattern? **1 / 1 point**

```
<MealProvider render={data => (
  <p>Ingredients: {data.ingredients}</p>
)}/>
```

- ◉ Yes
- ○ No

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

8. Inspect the given code snippet. **1 / 1 point**

```
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?

- ◉ As a second argument of 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 of the arrow function passed to the `useEffect()` call.

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 / 1 point**

```
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>
  );
};
```

- True
- False

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

10. Is this valid code? **1 / 1 point**

```
if (data !== '') {
  useEffect(() => {
    setData('test data');
  });
}
```

- No
- Yes

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