Final graded quiz: Advanced React

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: **1 / 1 ponto**

- 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.

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

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

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

Correto
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 ponto**

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?

```
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.
    setTodos([...todos].reverse());
  };

  return (
    <div>
      <button onClick={reverseOrder}>Reverse</button>
      {todos.map((todo, index) => (
        <ToDo key={index} id={todo.id} createdAt={todo.createdAt} />
      ))}
    </div>
  );
```

- (●) todo2 Wash dishes 20:30

  todo1 Buy groceries 18:00

- ( ) todo1 Buy groceries 18:00

  todo2 Wash dishes 20:30

- ( ) todo2 Buy groceries 20:30

  todo1 Wash dishes 18:00

Correto
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. A team is tasked with implementing a `ThemeProvider` for an application that will inject into the tree a light/dark theme, as well as a `toggler` function to be able to change it. For that, the following solution code has been incorporated. However, unwittingly, some errors have been introduced that prevent it from working correctly. What are the errors in the code solution? Select all that apply. **0.5 / 1 ponto**

```
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>
  );
};
```

- [x] There is no need to use local state for the context.

Não deve ser selecionado
Not quite, please review Module 1, Lesson 4, the lesson item titled *What is context and why it's used.*

- [x] The default value of `createContext` should be `"light"` instead of `undefined`.

Não deve ser selecionado
Not quite, please review Module 1, Lesson 4, the lesson item titled *What is context and why it's used.*

- [x] The children are not passed through

Correto
Correct, `ThemeProvider` should use the `children` prop and pass it as a direct child of `ThemeContext.Provider`.

- [x] The `toggleTheme` implementation is incorrect.

Correto
Correct, it should be instead `toggleTheme: () =>setTheme(theme === "light" ? "dark" : "light")`.

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

- [x] Each element object should have at least two properties: type and children

Não deve ser selecionado
Not quite, please review the lesson item titled JSX, Components and Elements in lesson 1 of module 3.

- [x] The type of an element can be a DOM node, like a HTML button.

Correto
Correct, the type can be a DOM node.

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

Correto
Correct, they can be mixed and matched.

☑ The type of an element can be a function corresponding to a React component, like a `SubmitButton`.

Correto
Correct, the type can be a React component.

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 ponto**

```
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”

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

6. **1 / 1 ponto**

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.

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

7. **1 / 1 ponto**

Among the following code examples, what are valid implementations of the render props pattern?

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

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

☑
```
<Row renderIcon={() => <Icon name=”add” />} />
```

Correto
Correct, it uses a render type prop that is a function that returns JSX. Even though the name is not exactly “render”, explicit variations are also valid.

```
<LoginUser renderUser={<p>Mark</p>} />
```

8. You need the below code snippet to run only after the initial render. What updates (if any) do you need to make to the code? **1 / 1 ponto**

```
React.useEffect(()=> {
 console.log('The value of the toggle variable is', toggle)
})
```

- (●) Add an empty dependency array.
- ( ) You shouldn't make any updates.
- ( ) You should remove the toggle variable .

Correto
Correct! To run the effect only on the initial render, you need an empty dependency array.

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 ponto**

```
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

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

10. The below code is not valid, because: **1 / 1 ponto**

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

- (●) You're breaking the rules of hooks.
- ( ) You're using the if statement wrong.
- ( ) You're using the arrow function wrong.
- ( ) There's a typo in the arrow function.

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