Congratulations! You passed!

Grade received 100% **To pass** 80% or higher

Go to next item

1. What of the next input types doesn't have a controlled version when they are used in React? 1 / 1 point

- (●) `<input type="file" />`
- ( ) `<input type="text" />`
- ( ) `<textarea />`

Correct
That's correct, because its value is read-only, it is an uncontrolled component in React.

2. What are some of the features of controlled components? Select all that apply 1 / 1 point

- [x] Conditionally disabling the submit button.

Correct
That's correct, you can specify a condition based on the React state from the different inputs to disable the submit button.

- [x] Enforcing a specific input format.

Correct
That's correct, you can use regular expressions and match the React local state against them to provide instant feedback to users about a specific pattern, like credit cards or phone numbers. Alternatively, you can also control what characters are valid for a specific input and pass them through and ignore invalid ones

- [x] Validating all values in the client side when a submission occurs in the form, before calling the server endpoint.

Correct
That's correct, you can access the different input states via React state and perform validation in the submit handler.

3. How do you get the value of an input when its state is handled by the DOM (Uncontrolled)? Select all that apply. 1 / 1 point

- ( ) Using a combination of `useEffect` and `useRef` hooks, where a ref is used on the uncontrolled input and then its value can be read on `useEffect` after a re-render cycle happens.
- ( ) Using local state and initializing it to an empty string. Then, reading the input value from the event object when the submission happens and finally setting the local state with that value.
- (●) Using a ref via `useRef` hook, assigning it to the input and then reading the input value when the submission happens via `ref.current.value`.

Correct
That's correct, that's the proper way to access the value from an uncontrolled input.

4. What happens when you click on the submit button in the below code snippet? 1 / 1 point

```
1  <form onSubmit={() => alert("Submitting")}>
2    <input type="text" value={text} onChange={e => setText(e.target.value)} />
3    <input type="button" value="Submit" />
4  </form>
```

- ○ The `onSubmit` callback is executed and an alert is shown with the text "Submitting".
- ○ An error is thrown.
- ◉ Nothing happens when the button is clicked.

Correct
That's correct, the input should be of type submit, otherwise `onSubmit` callback from the form tag won't fire.

5. What is missing in the below code for the select component to work properly? **1 / 1 point**

```
<select onChange={handleChange}>
  <option value="grapefruit">Grapefruit</option>
  <option value="lime">Lime</option>
  <option value="coconut">Coconut</option>
  <option value="mango">Mango</option>
</select>
```

- ○ Each `option` tag should be accompanied by a `label` tag.
- ○ Each `option` tag should have an `onChange` handler.
- ◉ The `select` tag is missing a `value` prop.

Correct
That's correct, the `select` tag is missing the current selection via the `value` prop.