Quiz

This quiz will test your knowledge on creating strongly-typed function component state.

Creating strongly-typed function component state

1

What is the inferred type for the `name` state from the `useState` hook?

```
const [name, setName] = React.useState("");
```

2

How can we define a state variable called `email` with the `useState` hook that can be a `string` or `null`. Its initial value should be `null`.

3

We are implementing state using the `useReducer` hook. One of the pieces of state, called `data`, is an array of strings. An action object like the one below will start the process of updating the `data` state:

```
{
  type: "loaded",
  data: ["Bob", "Tom", "Jane"]
}
```

What type could represent this action object? We want the object to be immutable.

We have the following reducer function that we want to use in the `useReducer` hook:

```
function reducer (state: State, action: Actions): State {
  switch (action.type) {
    case 'loading':
      return { ...state, loading: true };
    case 'loaded':
      return { ...state, loading: false, data: action.data };
  }
};
```

How can we improve the implementation so that the type checking process ensures all the types in the `Actions` union are handled in the `switch` statement?

5

Below is a `useReducer` hook:

```
const [state, dispatch] = React.useReducer(
  (state: State, action: Actions) => {
    switch (action.type) {
      case "add":
        return { total: state.total + action.amount };
      case "subtract":
        return { total: state.total + action.amount };
    }
  },
  initialState
);
```

The inferred type for `state` isn't `State` - it is `{total: number}` instead.

What could we change in the implementation so that the type of `state` is `State`? (There is more than one correct answer).

Check Answers

Now that we have got to grips with strongly-typing function components, in the next chapter, we will learn how to strongly-type class components.