State in React

If props represent outside data for a component, its state represents the internal data.

WE'LL COVER THE FOLLOWING

What is `state`

Each component can encapsulate a `state` object which is accessible inside all the functions but is not exposed to the outside world. Similar to `props`, you can add primitives as well as complex structures to a `state` object.

How `setState()` works

The state should not be mutated directly but through the `setState()` function!

In order to change any value inside the `state` object, we do the following call:

```
this.setState({key: newValue});
```

With this, the `key` is either updated in the state or created if it didn't exist before. All other keys are preserved, so you don't need to specify them each time you call `setState`.

`setState` also accepts a function as a parameter:

```
this.setState((oldState) => ({ value: oldState.value + 1 }));
```

In this example, we are increasing the `value` with one, but we need the old value in order to compute the new one. Even if you're not using the `oldState`, this is the recommended way of working with `setState`.

this is the recommended way of working with `setState`.

A component that handles its state would look like this:

```
import React from 'react';

export default class Message extends React.Component {
    constructor() {
        super();
        this.state = {
            showMessage: true,
        };

        this.toggle = this.toggle.bind(this);
    }

    toggle() {
        this.setState(state => ({
            showMessage: !state.showMessage,
        }));
    }

    render() {
        return (
            <div>
                {this.state.showMessage && "Message is visible"}
                <button onClick={this.toggle}>Click me!</button>
            </div>
        );
    }
}
```

The constructor creates the state with the key `showMessage` and value `true` on **lines 6-8**. Then the `toggle()` function inverts the value of `showMessage` by setting the state on **lines 14-16**. The `render` function creates a div on **lines 22-24** that displays a message if `showMessage` is `true`. Curly braces inside JSX, as on **line 22**, indicate that the code inside them is to be treated as JavaScript code. So, the string “Message is visible” is only displayed if `this.state.showMessage` is `true`. Then, a button is created on **line 23** that calls the `toggle()` function when clicked which causes the message to disappear/reappear.

> **REMINDER**: `setState` does not perform an instant action and the state object will not be updated on the next line. Also, behind the curtains, `React` tries to optimize the number of re-renders by batching up multiple calls to `setState`.

Quick quiz on `state`

1 Which of the following is correct?

COMPLETED 0%

1 of 5

Now that we have revised some foundational elements and state in React, let’s study lifecycle hooks in the next lesson.