React: State and Events

Phase 2 // Week 1, Lesson 2

Sakib Rasul | Updated July 5, 2023 | Created June 12, 2023

First, a question.

Today's Objectives

Today, we'll answer the following questions:

1. *What is **state** in React?*
    A. *How do we create it?*
    B. *How do we update it?*
2. *How do we handle **events** in React?*

What is state?

- A component's **state** is its *memory between renders*.

1. State is dynamic. It is data that changes over a component's lifetime.
2. State is independent. Its value cannot be derived from existing props or state.

State notices a change in a component and invokes a re-rendering of a component with the changes

State: An Example

Let’s say you’re building a website for a museum.

It consists of an App that renders multiple Gallery components.

A Gallery is a navigable slideshow for a given artist and their works.

What might make sense for App to pass with props?

What might make sense for Gallery to hold in state?

App = parent

Gallery = components
Q. What props would we pass into the "app"

A. More granular components such as artist name, size, year made etc.
-not "state" because these components are never going to change

Which of these are(n't) state?

1. The `address` a user's typed into a `Form`, before they've clicked submit.
   This is State because there is an initial value (an empty string) then when they type the address, the value is changed (re-rendered) and would be a different state & value
2. An `App`'s `name`.
   Static, thus is not state. Not gonna change
3. A user's `fullName`, given that a `Form` holds `firstName` and `lastName` in state.
   Any calculation or two components that are combined does not create a new state, it is derived from the broader state.
4. Whether or not a user has added a `Product` to their cart.
5. A `Car`'s `name`, passed down as a prop from the `Garage` it belongs to.
6. How many times you've clicked "Generate!" in a `DALLEImageGenerator` component.
   Is state because the value changes with each passing click

Which of these are(n't) state?

1. The `address` a user's typed into a `Form`, before they've clicked submit.
2. An `App`'s `name`.
3. A user's `fullName`, given that a `Form` holds `firstName` and `lastName` in state.
4. Whether or not a user has added a `Product` to their cart.
5. A `Car`'s `name`, passed down as a prop from the `Garage` it belongs to.
6. How many times you've clicked "Generate!" in a `DALLEImageGenerator` component.

How do we hold memory in state?

To hold something in state, we invoke `useState`.

```
const [ isInCart, setIsInCart ] = useState(false);
```

How do we hold memory in state?

To hold something in state, we invoke `useState`.

```
const [ something, setSomething ] = useState(initialValue);
```

What not to hold in state

1. **Don’t mirror props in state.** If you want to rename a prop for the purposes of a component, assign it to a new constant.

```
function NutritionFacts({ foodCalories }) {
    const calories = foodCalories;
}
```

2. **Don’t hold things in state that you can derive.** Declare a new constant if you want a name to refer to what you’re deriving.

```
const [ firstName, setFirstName ] = useState(“”);

const [ lastName, setLastName ] = useState(“”);

const fullName = firstName + “ “ + lastName;
```

How do we update state?

To update state, we use call its setter.

```
const [ isInCart, setIsInCart ] = useState(false);

...

function addToCart() { setIsInCart(true) };
```

How do we update state?

To update state, we use call its setter.

```
const [ something, setSomething ] = useState(initialValue);

                                    …

   function updateSomething() { setSomething(newValue) };
```

What happens when we update state?

Thinking in React means we follow two principles:

1. Any app can be broken into independent, reusable components.
2. We shouldn't have to re-render an entire DOM whenever something happens.

State is how React ensures its second principle. **Updating state tells React to queue up a re-render of the component it belongs to with fresh data**.

Handling Events

We handle **events** in React similarly to how we added event listeners in JS:

1. We declare a function that will handle an event of our choice.

```
function sayHi() { alert(“Hi!”) };
```

2. We pass that function as a prop to the event’s target.

```
<button onClick={sayHi}>Hey there.</button>
```

Handling Events with State

Since state is dynamic and independent, we'll often want to update it inside event handlers:

1. We declare a function that will handle an event of our choice.

```
const [ isOn, setIsOn ] = useState(false);

function toggleLamp() { setIsOn(!isOn) };
```

2. We pass that function as a prop to the event's target.

```
<button onClick={toggleLamp}>Turn Lamp On/Off</button>
```

Time to sell a TV.
And a spatula.

Questions? // Thanks!