Multiple useState calls

Let's learn how to manage multiple useState calls and use them in a React component.

WE'LL COVER THE FOLLOWING

With class components, we all got used to setting state values in an object whether it contained a single property or more.

```
// single property
state = {
  count: 0
}
// multiple properties
state = {
 count: 0,
 time: '07:00'
}
```

With `useState` you may have noticed a subtle difference.

In the last lesson, we only called `useState` with the actual initial value. Not an object to hold the value.

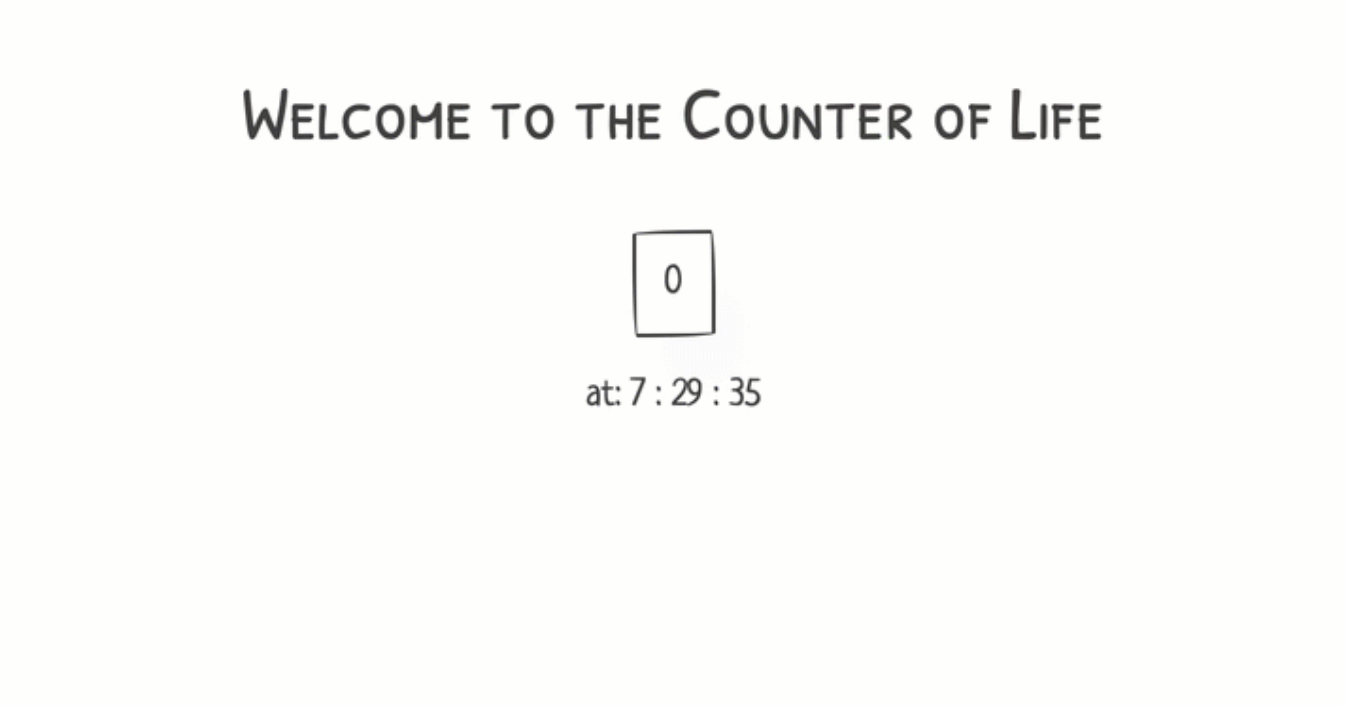
```
useState(0)
```

What if we wanted to keep track of another state value?

Can multiple `useState` calls be used?

The Counter App 2.0

Consider the component below. It's the same counter application with a twist. This time the counter keeps track of the time at which it was clicked.




```
function CounterHooks() {
  const [count, setCount] = useState(0);
  const [time, setTime] = useState(new Date())
  const handleClick = () => {
    setCount(count + 1);
    setTime(new Date())
  }
  return (
      <div>
        <h3>Welcome to the Counter of Life </h3>
        <button onClick={handleClick}>{count}</button>
        <p>
          at: {`${time.getHours()} : ${time.getMinutes()} :${time.getSeconds()}`}
                          ...
        </p>
       </div>
    );
}
```

As you can see, the hook's usage is quite the same, except for having a new `useState` call.

```
const [time, setTime] = useState(new Date())
```

Now, the `time` state variable is used in the rendered markup to display the hour, minute, and second of the click.

```
<p>
    at: {`${time.getHours()} : ${time.getMinutes()} :${time.getSeconds()}`}
  </p>
```

Great!

In this next lesson, we'll discuss the difference between setState and useState.