State y setState

DigitalHouse >
Coding School

Índice

“

El **setState()** programa una actualización al **objeto state** de un componente. Cuando el **state** cambia, el componente responde volviendo a **renderizar**.

”

1 State

State

En su momento, dijimos que una de las razones por las cuales React nos beneficia es la **actualización por componentes del DOM**.

También, dijimos que los componentes con estado eran **reactivos a las interacciones con el usuario** y que, en base a eso, se actualizaban o no.

Ahora, llegó el momento de fusionar estas dos características de React y empezar a usarlas.

Vamos a ver cómo usamos **setState** para cambiar el estado de nuestros componentes, ya sea ante eventos de usuario, cambios en el servidor o cambios en los props.

Veamos cómo se ve esto en el código con un breve ejemplo.

2 | Constructor

Constructor
{código}

{}

```
class contador extends Component{
 constructor(){
  super();
  this.state = {
   valor:1,
  }
 }
}
```

El método **constructor** es necesario para poder definir la estructura de un componente.

Constructor
{código}

{}

```
class contador extends Component{
 constructor(){
  super();
  this.state = {
   valor:1,
  }
 }
}
```

La función **super** en el constructor es necesaria en React ya que de esa forma podemos utilizar las props que hereda de su componente padre.

Constructor

{código}

{}

```
class contador extends Component{
 constructor(){
  super();
  this.state = {
   valor:1,
  }
 }
}
```

El constructor es el único lugar donde debemos asignar **this.state** directamente. Este va a ser un objeto literal.

3 | Props

Props
{código}

{}

```
class contador extends Component{
 constructor(props){
  super(props);
  this.state = {
   valor:props.value,
  }
 }
}
```

Podemos recibir las **props** en el **constructor**. Es buena práctica utilizarlas al llamar al **super**.

4 | setState

setState
{código}

{}

```
class contador extends Component{
 constructor(){
  super();
  this.state = {
   valor:1,
  }
 }
 incrementar(){
  this.setState({
   valor: this.state.valor + 1
  });
 }
}
```

En todos los métodos que **no** sean el **constructor** debemos utilizar **this.setState()**.

setState
{código}

{}

```
class contador extends Component{
// Aqui va el constructor
incrementar(){
 this.setState({
 valor: this.state.valor + 1
 });
 }
 render(){
   return (
     <button
       onClick={this.incrementar}>
     </button>
   );
 }
}
```

Con el evento **onClick** vamos a estar modificando a través del método incrementar, el estado de nuestro componente.

Documentación

Para saber más sobre setState en los componentes stateful podemos acceder a la documentación oficial de React haciendo clic en el siguiente link.


DigitalHouse>
Coding School