React.JS

Introducción

Módulo 02

State

State

Consideremos el ejemplo anterior donde queríamos ir sumando una cantidad dado el input de un usuario. Dicha cantidad debería estar almacenada en algún lugar de manera local por posibles usos futuros del mismo componente.

Para implementar esto, podemos usar el ***state*** o *"estado"* del componente. El estado es similar a los accesorios, pero es privado y está completamente controlado por el componente.

Anteriormente mencionamos que los componentes definidos como clases tienen algunas características adicionales. **El estado local es exactamente eso: una característica disponible solo para las clases.**

Estado en componentes de clase

Vamos a implementar un componente que nos permita ver el estado de nuestra cuenta:

```
import React from 'react';

class Componente extends React.Component {
    render() {
        let contador = 0;
        return (
            <div>
                { contador }
            </div>
        )
    }
}

export default Componente;
```

```
import React from 'react';

class Componente extends React.Component {
    constructor(props) {
        super(props);
        this.state = {
            contador: 0
        }
    }

    render() {
        return (
            <div>
                { this.state.contador }
            </div>
        )
    }
}

export default Componente;
```

Para hacerlo, necesitamos inicializar la variable ***this.state*** en el constructor del componente.

No es necesario que tenga los valores definitivos, pero sí debe tener todas las propiedades.

Una vez hecho eso, podemos leerla mediante ***this.state*** en cualquier parte de la clase.

```
import React from 'react';

class Componente extends React.Component {
    constructor(props) {
        super(props);
        this.state = {
            contador: 0
        }
    }

    render() {
        return (
            <div>
                <button onClick={
                    () => this.setState({contador: this.state.contador + 1})
                    }>Incrementar</button>
                { this.state.contador }
            </div>
        )
    }
}

export default Componente;
```

El estado es una variable especial:

cuando cambia su valor mediante la función *setState*, eso ocasiona que el componente se vuelva a dibujar, es decir, se vuelva a ejecutar el método *render()* con el nuevo valor del estado.

Un ejemplo muy claro es cuando cambiamos el estado en un evento.

Estado en componentes funcionales

¿Y qué hay de los componentes funcionales?
Los **componentes funcionales**, como vimos previamente, son aquellos que se definen mediante funciones en lugar de clases.

Con respecto al estado, los componentes funcionales no pueden usar la variable *state* al igual que las clases.

Sin embargo, pueden acceder al estado y cambiarlo a través de los ***Hooks*** o *"ganchos"*. **Los *Hooks* son la forma de añadir funcionalidad propia de los componentes basados en clases a los componentes funcionales.**

Más información en **https://es.reactjs.org/docs/hooks-intro.html**

```jsx
import React, { useState } from "react";

export default function Componente(props) {
    const contador = useState(0);
    const getContador = contador[0];
    const setContador = contador[1];
    return (
        <div>
            <button onClick={() => setContador(getContador + 1)}>
                Incrementar
            </button>
            {getContador}
        </div>
    );
}
```

Los ***hooks*** son funciones especiales, cada una con su propia definición.

useState permite obtener un array con un elemento para leer y otro para escribir el estado.

¡Muchas gracias!

¡Sigamos trabajando!