React.JS

Introducción

Módulo 02

State Reglas

Usando el estado correctamente

Hay cuatro cosas que se debe saber sobre el *estado:*

- El estado no se modifica directamente.
- Las actualizaciones de estado pueden ser asincrónicas.
- Las actualizaciones de estado pueden ser unidas.
- La información fluye en una sola dirección, “hacia abajo”.

El estado no se modifica directamente

En el ejemplo anterior, realizar esa modificación no activará el auto render del componente. Usamos el método ***setState()***

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

Las actualizaciones de estado pueden ser asincrónicas

```
// Incorrecto
this.setState({
    contador: this.state.contador + 1
});

// Correcto
this.setState((prevState, props) => ({
    contador: prevState + 1
}));
// O sino...
this.setState(function(prevState, props){
    return {
        contador: prevState + 1
    }
})
```

React puede agrupar varias llamadas ***setState()*** en una sola actualización para el rendimiento.

Como ***this.props*** y ***this.state*** se pueden actualizar de forma asincrónica, no debe confiar en sus valores para calcular el siguiente estado.

Para solucionarlo, use una segunda forma de ***setState()*** que acepte una función en lugar de un objeto. Esa función recibirá el estado anterior como primer argumento y los puntales en el momento en que se aplica la actualización como segundo argumento.

Las actualizaciones de estado pueden ser unidas

Cuando llama a setState (), React fusiona el objeto que proporciona en el estado actual. Por ejemplo, si su estado contiene varias partes independientes, luego puede actualizarlos de manera independiente con llamadas setState () separadas: La fusión es superficial, por lo que una llamada a this.setState que actualice una sola parte del estado, deja intacta el resto de las partes.

```
componentDidMount() {
    fetch("/posts")
        .then((r) => r.json())
        .then((posts) => this.setState({ posts: posts }));

    fetch("/comments")
        .then((r) => r.json())
        .then((comments) => this.setState({ comments: comments }));
}
```

Las actualizaciones de estado pueden ser unidas

Cuando llama a ***setState()***, React fusiona el objeto que proporciona en el estado actual. Por ejemplo, si su estado contiene varias partes independientes, luego puede actualizarlos de manera independiente con llamadas ***setState() separadas***: La fusión es superficial, por lo que una llamada a this.setState que actualice una sola parte del estado, deja intacta el resto de las partes.

```
componentDidMount() {
    fetch("/posts")
        .then((r) => r.json())
        .then((posts) => this.setState({ posts: posts }));

    fetch("/comments")
        .then((r) => r.json())
        .then((comments) => this.setState({ comments: comments }));
}
```

La información fluye hacia abajo

Ni los componentes primarios ni secundarios pueden saber si un componente determinado tiene estado o no, y no debería importar si se define como una función o una clase.

Esta es la razón por la cual el estado a menudo se llama *local* o *encapsulado.* No es accesible desde ningún componente que no sea el que posee y lo establece.

Un componente puede elegir pasar su estado como props a sus componentes secundarios.

Esto se conoce comúnmente como **flujo de datos *"descendente"* o *"unidireccional"*.**

```jsx
import React from "react";
import Texto from './Texto';

export default class Contador 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>
                <Texto contenido={this.state.contador} />
            </div>
        );
    }
}
```

¡Muchas gracias!

¡Sigamos trabajando!