🇪🇸 UseState ReactJS[1]

Introducción

useState() es una función que crea internamente una variable donde podremos almacenar el estado de nuestro componente. Acepta un valor inicial para esa variable y devuelve un array con dos elementos, el valor de la variable y la función para modificarla.

Gestión de estados con useState

Antes de comenzar para este ejemplo vamos a crear un proyecto de React con Vite .

```
npm create @vite-latest

name - useStateOne
react
javascript
cd useStateOne
npm i
```

Para entender el Hook de useState vamos a realizar un breve ejemplo de un `h1` que va cambiando en función de un `input`.

Una vez hemos terminado de inicializar el proyecto, vamos a crear en nuestro `src`, un componente llamado `MyState.jsx` , es un componente de tipo función que devuelve elementos de tipo React Element.

```
export const MyState = () => {
  return <>MyState</>;
};
```

Una vez tenemos nuestro componente vamos a crear un **estado** para nuestro componente que contenga un '**getter'** y un '**setter'** para pintar o modificar el valor de nuestro **estado.**

```
const [myName, setMyName] = useState("Ziggy Stardust");
```

Como ya mencionamos previamente, la constante `myName` sería nuestro 'getter' y la `setMyName` nuestro 'setter', y el valor por defecto o inicial de nuestro estado es Ziggy Stardust. Ahora ya tenemos un estado con el que poder jugar e interactuar 🙈.

Ahora vamos a definir un `input` que pueda modificar el valor de nuestro state cuando escribamos en éste.

```
return (
    <>
      <h1>{myName}</h1>
      <input
        type="text"
        value={myName}
        onChange={(e) => setMyName(e.target.value)}
      />
    </>
  );
```

Vemos que como particularidad en el `onChange` invocamos al 'setter' con el valor actual del `input`, gracias al `onChange` cambiará a tiempo real y nuestro estado será totalmente dinámico.

Recuerda que cambiar el estado le pide a React un repintado o renderizado de nuestra app cuando sea posible, por lo que veremos el cambio de `myName` a tiempo real. De tal modo que nuestro componente queda.

```
import { useState } from "react";

export const MyState = () => {
  const [myName, setMyName] = useState("Ziggy Stardust");
  return (
    <>
      <h4>{myName}</h4>
      <input
        type="text"
        value={myName}
        onChange={(e) => setMyName(e.target.value)}
      />
    </>
  );
};
```

Por último vamos a importarlo en el `App.jsx`, y arrancar el proyecto para comprobar si funciona nuestro state.

```
import { MyState } from "./components/MyState";

const App = () => {
  return <MyState />;
};

export default App;
```

Arrancamos !!!

```
npm run dev
```

Y el resultado que obtenemos.

¿Qué es lo que hemos conseguido? Pues que cada vez que se lanza el evento `onChange` realizamos el ‘set’ del valor del state y React está repintando los cambios que estamos enviando al input 👋.