ThePower business school

🇪🇸 UseState ReactJS[2]

Introducción

Vamos a continuar con useState, y ahora vamos a crear un componente que almacene en el **state** un dato más complejo como puede ser un objeto. Hay que tener en cuenta que cuando modificamos un atributo de un objeto debemos respetar el principio de inmutabilidad y no modificar el objeto original.

Gestión de estados con useState

Vamos a crear un nuevo archivo `ObjectState.jsx`. Y creamos nuestro state cuyo valor inicial será un objeto.

```
export const ObjectState = () => {
  const [avengerInfo, setAvengerInfo] = useState({
    name: "Thor",
    lastName: "Odinson",
   });
   return (
    <>
      <h4>ObjectState</h4>
    </>
  );
};
```

Por último necesitaremos un input que nos ayude a modificar nuestro estado y respetar el principio de inmutabilidad, esto quiere decir que no asignamos directamente el valor sino que lo haremos a través de nuestro ‘setter’, además tendremos que hacer uso de los `spread operators` para recuperar todas las propiedades del objeto y modificar solamente la que deseamos. Vamos a por ello.

```
return (
    <>
```

```
      <h4>
        {avengerInfo.name} | {avengerInfo.lastName}
      </h4>

      <input
        type="text"
        value={avengerInfo.name}
        onChange={(e) =>
          setAvengerInfo({
            ...avengerInfo,
            name: e.target.value,
          })
        }
      />

      <input
        type="text"
        value={avengerInfo.lastName}
        onChange={(e) =>
          setAvengerInfo({
            ...avengerInfo,
            lastName: e.target.value,
          })
        }
      />
    </>
  );
```

De este modo ya tenemos nuestro componente funcionando, y queda así.

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

export const MiniCodeObjectState = () => {
  const [avengerInfo, setAvengerInfo] = useState({
    name: "Thor",
    lastName: "Odinson",
  });

  return (
    <>
      <h4>
        {avengerInfo.name} | {avengerInfo.lastName}
      </h4>

      <input
        type="text"
        value={avengerInfo.name}
        onChange={(e) =>
          setAvengerInfo({
```

```
            ...avengerInfo,
            name: e.target.value,
          })
        }
      />
      <input
        type="text"
        value={avengerInfo.lastName}
        onChange={(e) =>
          setAvengerInfo({
            ...avengerInfo,
            lastName: e.target.value,
          })
        }
      />
    </>
  );
};
```

Ahora lo importamos en el `App.jsx`, y lo probamos 😝.

```
import "./App.css";
import { MiniCodeObjectState } from "./components/MiniCodeObjectState";

const App = () => {
  return (
    <div className="App">
      <MiniCodeObjectState />
    </div>
  );
};

export default App;
```

Y el resultado obtenido es el siguiente.

¡Ya tenemos varios valores controlados con un solo estado! 🔥 Tenemos que tener en cuenta que a ser posible, nuestros state deben ser lo más simples posibles dentro nuestras capacidades de atomización del contenido de la app. Así evitaremos estructuras de datos complejas y difíciles de controlar en el futuro.