useEffect Hook:

UseEffect Hook le permite realizar efectos secundarios en tus componentes.

Algunos ejemplos de efectos secundarios son: obtención de datos, actualización directa del DOM y temporizadores.

useEffect acepta dos argumentos. El segundo argumento es opcional.

useEffect(<función>, <dependencia>)

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

export default function App() {

   const [count, setCount] = useState(0);
    useEffect(() => {
     setTimeout(() => {
       setCount((count) => count + 1);
     }, 1000);
   });
    return <h1>I've rendered {count} times!</h1>;
}
```

¡¡Pero espera!! ¡Sigue contando aunque solo debería contar una vez!

useEffect se ejecuta en cada renderizado. Eso significa que cuando cambia el conteo, ocurre un renderizado, que luego desencadena otro efecto.

Esto no es lo que queremos. Hay varias formas de controlar cuándo se presentan los efectos secundarios.

Siempre debemos incluir el segundo parámetro que acepta una matriz. Opcionalmente, podemos pasar dependencias a useEffect en este array.

```
useEffect(() => {
  //Corre cada vez que  renderiza

});
```

```
useEffect(() => {

  //Corre solo la primera vez que renderiza

}, []);
```

```
useEffect(() => {

  //Corre la primera vez que renderiza

  //Y cada vez que haya un cambio en la dependencia

}, [prop, state]);
```

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



export default function App() {



   const [count, setCount] = useState(0);

    useEffect(() => {

     setTimeout(() => {

       setCount((count) => count + 1);

     }, 1000);

   }, []);

    return <h1>I've rendered {count} times!</h1>;

}
```

Renderiza 2 veces por el React.StrictMode.
Aquí hay un ejemplo de un useEffect Hook que depende de una variable. Si la variable de conteo se actualiza, el efecto se ejecutará nuevamente:

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

export default function App() {
 const [count, setCount] = useState(0);
 const [calculation, setCalculation] = useState(0);

 useEffect(() => {
   setCalculation(() => count * 2);
 }, [count]);

 return (
```

```
    <>
      <p>Count: {count}</p>
      <button onClick={() => setCount((c) => c + 1)}>+</button>
      <p>Calculation: {calculation}</p>
    </>
  );
}
```