ThePower business school

🇪🇸 UseEffect ReactJS[2]

Introducción

En este ejemplo vamos a ver cómo liberar recursos cuando desmontamos un componente del DOM.

Code

Para ello vamos a crear un componente llamado `CodeEffectUnmount.jsx` :

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

Ahora vamos a crear un state para definir si queremos tener nuestro componente visible o invisible.

```
const [visible, setVisible] = useState(false);
```

Y en nuestro `return`, preguntaremos si es visible para mostrar el contenido.

```
return <>{visible && <h4>I'm Iron Man</h4>}</>;
```

Vamos a dotarle de un poquito de funcionalidad, creamos un botón que cambie el estado para visualizar el contenido.

```
return (
    <>
      {visible && <h4>I'm Iron Man</h4>}
      <button onClick={() => setVisible(!visible)}>I'm inevitable</button>
    </>
  );
```

Y si empezásemos a componetizar nuestra aplicación, haciendo el `h4` un componente.

```
export const MessageComponent = () => {
  return <h4>I'm Iron Man</h4>
}
```

Y lo usamos en el componente padre tendremos el siguiente código.

```
export const MiniCodeEffectUnmount = () => {
  const [visible, setVisible] = React.useState(false);

  return (
    <>
      {visible && <MessageComponent />}
      <button onClick={() => setVisible(!visible)}>I'm inevitable</button>
    </>
  );
};
```

Ahora tenemos un componente hijo que se monta cuando cambia el estado, pero cómo podemos desmontar ese componente una vez volvemos a clickar el botón del state, podemos añadir un useEffect dentro de `MessageComponent` para lanzar side effects al desmontarlo. Aunque primero vamos a comprobar que se monta en el DOM correctamente.

```
React.useEffect(() => {
  console.log("Me monto en el DOM");
}, []);
```

Pero, ¿cómo podremos hacer que nos muestre por consola un mensaje cada vez que este componente se desmonte?, pues useEffect espera que devuelvas una función que se ejecutará cuando se desmonte el componente.

```
React.useEffect(() => {
  console.log('Me monto en el DOM');

  return () => {
    console.log('Me desmonto del DOM');
  };
}, []);
```

Si comprobamos nuestro código.

```
export const MessageComponent = () => {
  React.useEffect(() => {
    console.log('Me monto en el DOM');

    return () => {
      console.log('Me desmonto del DOM');
    };
  }, []);
  return <h4>I'm Iron Man</h4>
}
```

Y en el navegador observaremos cuando se monta y desmonta el componente en el DOM.

Utilidad

¿Para qué nos puede servir esto? Cuando abres una conexión a un webSocket y quieres ocultarla cuando el usuario oculte el componente, de tal modo que cuando el componente se monta, abre el socket, y cuando se desmonta lo libera.

Otro ejemplo es el de los “event listeners", como la escucha de un scroll, el movimiento del ratón, o cualquier otro evento de JavaScript. Para evitar múltiples adiciones del mismo listener al montar/desmontar el componente, tendrás que devolver en el useEffect el limpiado de estos listeners.