Certified Tech Developer
The Ultimate Degree

Front End III

Casos de uso de useEffect

Ahora que conocés los casos de uso de useEffect, veámoslos en profunidad.

1. Establecer el título de una página de forma imperativa

El título de una página se encuentra fuera del nodo raíz de React (#root).

```
<!DOCTYPE html>
<html lang="es">
   <head>
      <title>Título</title>
   </head>
   <body>
      <div id="root"></div>
   </body>
</html>
```

```
import React from "react";
import ReactDOM from "react-dom";

import { App } from "./components/App";

ReactDOM.render(
   <React.StrictMode>
      <App />
   </React.StrictMode>,
   document.getElementById("root")
);
```

Sin embargo, es accesible a través de la interfaz document.[1]

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

const ChangeTitleExample = () => {
   const [title, setTitle] = useState(
      "Certified Tech Developer | Digital House"
   );

   useEffect(() => {
      document.title = title;
   }, [title]);

   return <div />;
};

export default ChangeTitleExample;
```

2. Trabajar con temporizadores: setInterval o setTimeout

Si trabajamos con temporizadores, aquí deben introducirse:

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

const Interval = () => {
   const [cantidad, setCantidad] = useState(2);

   useEffect(() => {
      const interval = setInterval(() => {
         setCantidad((prevState) => ++prevState);
      }, 1000);
   }, []);

   return <div>Quiero {cantidad} chocolates</div>;
};

export default Interval;
```

[1] La interfaz document representa cualquier página web cargada en el navegador y sirve como punto de entrada al contenido de la página (el árbol DOM). En https://developer.mozilla.org/es/docs/Web/API/Document.

Quiero 94 chocolates

3. Leer ancho, alto o posición de elementos del DOM

4. Registrar mensajes (console.log)

Aquí un ejemplo donde detectamos el ancho y alto de la ventana al redimensionarla y registramos un mensaje en consola.

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

const GetResizeFromWindow = () => {
   useEffect(() => {
      function handleResize() {
         console.log(
            "Redimensionar: ",
            window.innerWidth,
            "x",
            window.innerHeight
         );
      }

      window.addEventListener("resize", handleResize);
   });

   return <div />;
};

export default GetResizeFromWindow;
```

```
Redimensionar:  888 x 594
Redimensionar:  889 x 594
Redimensionar:  891 x 594
Redimensionar:  894 x 594
Redimensionar:  899 x 596
Redimensionar:  900 x 596
Redimensionar:  904 x 598
Redimensionar:  907 x 599
```

5. Establecer u obtener valores de almacenamiento local

Podemos usar el almacenamiento local del navegador para una gran cantidad de situaciones. Por ejemplo, en casos de corroborar si un usuario está logueado.

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

import Login from "./Login";
import Home from "./Home";

const UserLogged = () => {
   const [user, setUser] = useState();

   useEffect(() => {
      function checkUser() {
         const item = localStorage.getItem("User");
         if (item) setUser(item);
      }

      window.addEventListener("storage", checkUser);

      return () => {
         window.removeEventListener("storage", checkUser);
      };
   }, []);

   return <>{user ? <Home /> : <Login />}</>;
};

export default UserLogged;
```

6. Realizar peticiones a servicios

Cuando realizamos una petición a una API, en la función Fetch o Axios, debemos declarar async y await debido a que el callback de useEffect es sincrónico.

```
useEffect(() => {
   async function getCandy() {
      const resp = await fetch("https://alotofcandy.iumi");
      const data = await resp.json();
      setCandy(data);
   }
   getCandy();
}, []);
```