## Hook

Son una función que en general comienza con use, y nos permite reutilizar código:

Algunas son:

* useState()
* useEffect()

Básicamente los hooks son funciones que nos otorga react.

Reglas:

* No puede estar adentro de un condicional.
* Se tienen que llamar en el top-level del componente.
* Solo se pueden invocar en function components (o en otros hook).

## Método useEffect

Se pasa como parámetro un callback que contiene código imperativo. El callback se ejecutará después de que el renderizado es confirmado en la pantalla.

Sintaxis:

useEffect(() => {} , dependencias)

Se necesitan dos argumentos separados con una coma

El primer parámetro es una función que deseamos ejecutar cada vez que se ejecuta useEffect.

* Se usa para realizar efectos secundarios en una function component
* Se usa luego de los cambios del DOM
* Recibe dos argumentos: una función y un array de dependencias

Dato adicional si el elemento que declaramos en el arreglo de dependencias, cambia el useEffect cambia, un ejemplo de eso podemos verlo en el siguiente ejemplo;

useEffect(function(){

console.log (‘Cambio detectado)

}, [elemento]);

En este ejemplo vemos que si el elemento cambia, el useEffect se vuelve a ejecutar y renderiza el elemento.

Efectos secundarios en un componente de reacción.

El trabajo de react es renderizar, y volver a renderizar la interfaz de usuario cuando sea necesario (después de cambiar el estado, o accesorios). Aparte de eso, podemos enviar solicitudes http al servidor backend, como actualizar variables globales dentro del componente, guardar o actualizar el cache, se consideran efectos secundarios, como sugerencia no uses el efecto secundario dentro del componente principal, ya que nos puede crear errores como enviar demasiadas solicitudes http.