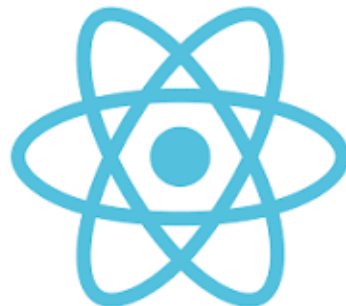



Agencia de  
Aprendizaje  
a lo largo  
de la vida

# ESPECIALIZACIÓN REACT





Agencia de  
Aprendizaje  
a lo largo  
de la vida

# Clase 10. REACT JS

## *HOOKS*

# HOOKS

Los Hooks son una API de React que nos permite tener estado, y otras características de React, en los componentes creados con una function.

Esto, antes, no era posible y nos obligaba a crear un componente con class para poder acceder a todas las posibilidades de la librería.

Hooks es gancho y, precisamente, lo que hacen, es que te permiten enganchar tus componentes funcionales a todas las características que ofrece React.

Este mismo concepto se puede aplicar a la interfaz de usuario. Por ejemplo, el botón Me Gusta de Facebook tendría el estado de meGusta a true cuando el usuario le ha dado a Me Gusta y a false cuando no lo ha hecho.

No solo podemos tener en el estado valores booleanos, también podemos tener objetos, arrays, números, etc.

Por ejemplo, si tienes un componente Counter que muestra un contador, puedes usar el estado para controlar el valor del contador.

Para crear un estado en React usamos el hook useState:

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

export const Counter = () => {
  const [count, setCount] = useState(0);

  return (
    <>
      <button onClick={() => setCount(count + 1)}>+</button>
      <p>Numero:{count}</p>
      <button onClick={() => setCount(count - 1)}>-</button>
    </>
  );
};
```

Al usar el hook useState este devolverá un array de dos posiciones:

El valor del estado.

La función para cambiar el estado.

Suele usarse desestructuración para facilitar la lectura y ahorrarnos algunas líneas de código. Por otro lado, al pasarle un dato como parámetro al useState le estamos indicando su estado inicial.

El hook useState es utilizado para crear variables de estado, quiere decir que su valor es dinámico, que este puede cambiar en el tiempo y eso requiere una re-renderización del componente donde se utiliza

Recibe un parámetro:

El valor inicial de nuestra variable de estado.

Devuelve un array con dos variables:

En primer lugar tenemos la variable que contiene el valor

La siguiente variable es una función set, requiere el nuevo valor del estado, y este modifica el valor de la variable que anteriormente mencionamos

Cabe destacar que la función proporciona cómo parametro el valor actual del propio estado.

## ¿Qué hace el hook useEffect?

El hook useEffect se usa para ejecutar código cuando se renderiza el componente o cuando cambian las dependencias del efecto.

Recibe dos parámetros:

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

export const Counter = () => {
  const [count, setCount] = useState(0);

  useEffect(() => {
    alert("Cambio el contador");
  }, [count]);

  return (
    <>
    <button onClick={() => setCount(count + 1)}>+</button>
    <p>Numero: {count}</p>
    <button onClick={() => setCount(count - 1)}>-</button>
    </>
  );
};
```

La función que se ejecutará al cambiar las dependencias o al renderizar el componente.

Un array de dependencias.

Si cambia el valor de alguna dependencia, ejecutará la función.

En este ejemplo mostramos un mensaje en consola cuando carga el componente y cada vez que cambia el valor de count:



Podemos usar el hook useEffect de diferentes formas, tales como:

Ejecutar código cuando se renderiza el componente, cuando cambian las dependencias del efecto o cuando se desmonta el componente.

Por eso puede ser útil para hacer llamadas a APIs, ya que sea nada más montar el componente o cuando cambian las dependencias.

Realizar tracking de eventos, como Google Analytics, para saber qué páginas visitan los usuarios. Podemos validar un formulario para que cada vez que cambie el estado, podamos actualizar la UI y mostrar dónde están los errores.

Podemos suscribirnos a eventos del navegador, como por ejemplo el evento resize para saber cuando el usuario cambia el tamaño de la ventana.