Material descargable

egg

Teoría React I

¿Qué son los estados de React?

Los componentes a menudo necesitan cambiar lo que se muestra en pantalla como resultado de una interacción. Escribir dentro de un formulario debería actualizar el campo de texto, hacer clic en «siguiente» en un carrusel de imágenes debería cambiar la imagen que es mostrada; hacer clic en un botón para comprar un producto debería actualizar el carrito de compras. En los ejemplos anteriores los componentes deben «recordar» cosas: el campo de texto, la imagen actual, el carrito de compras. En React, a este tipo de memoria de los componentes se le conoce como estado.

Cuando la interfaz ya fue renderizada y debido a la interactividad alguna variable cambia de valor, los cambios en las variables que venimos manejando no activarán renderizaciones ya que React no se da cuenta de que necesita renderizar el componente nuevamente con los nuevos datos.

Para actualizar un componente con datos nuevos, se deben conservar los datos entre renderizaciones y provocar que React re-renderice el componente con nuevos datos.

React nos ofrece una función para poder manejar estos cambios de estados. La función useState() ofrece: - Una variable de estado para mantener los datos entre renderizados. - Una función que actualiza el estado para actualizar la variable y provocar que React renderice el componente nuevamente. - Configurar el valor inicial del estado.

```
Unset
import { useState } from 'react';


export default function Contador() {

 const [contador, actualizarContador] = useState(0);

 const sumarUno = ()=> actualizarContador(contador + 1);
```

```
  return (<>
    <button onClick={sumarUno}>Sumar 1</button>
    <h3>{contador}</h3>
   </>);
}
```

En este ejemplo, la función devuelve un array con el estado y la función seteadora del estado y se inicializa con un numero. La convención es nombrar estas dos variables como `const [algo, setAlgo]`. Podemos nombrarlo como queramos, pero mantener las convenciones hacen que las cosas sean más fáciles de entender en todos los proyectos.

Es importante entender, que cada vez que se ejecute la función seteadora, se modificará el estado de la aplicación con su correspondiente re-renderización de la vista.

El estado es local para una instancia de un componente en la pantalla. En otras palabras, si renderizas el mismo componente dos veces, ¡cada copia tendrá un estado completamente aislado! Cambiar uno de ellos no afectará al otro.

¿Qué son los Hooks?

En React, useState(), así como cualquier otra función que empiece con "use", se le conoce como Hook.

Los Hooks son funciones especiales que sólo están disponibles mientras React está renderizando. Los Hooks permiten «engancharnos» a diferentes características de los componentes de React.

El hook useState() es una función que se engancha al estado del componente para modificarlo, pero el estado es solo una de esas características! Existen muchos más Hooks!

Los Hooks deben ser solo llamadas en el nivel superior de los componentes o en tus propios Hooks. No podemos usar Hooks dentro de condicionales, bucles, u otras funciones anidadas. Los Hooks son funciones, pero es útil pensar en ellos como declaraciones independientes de las necesidades de nuestro componente. Las funciones de React se «usan» en la parte superior del componente de manera

similar a cómo se «importan» módulos en la parte superior de un archivo.