Hooks en React

Los Hooks en React son funciones especiales que te permiten "enganchar" o "conectar" funcionalidades adicionales sin tener que modificar la estructura de tu componente. Esto hace que el código sea más legible, reutilizable y fácil de mantener. Para usarlos debemos seguir unas reglas básicas:

1. Los Hooks solo se pueden usar en componentes de función. No se pueden usar en componentes de clase.
2. Los Hooks deben invocarse de manera top-level en el cuerpo del componente de función. No pueden invocarse dentro de bucles, condiciones o funciones anidadas.
3. Los Hooks deben ser invocados únicamente dentro de componentes de función de React o de otros Hooks personalizados. No se pueden invocar fuera de estos contextos.
4. Los Hooks, como useState o useEffect, deben comenzar con "use" en su nombre para que React los pueda reconocer y hacer uso de ellos.

Hook de UseState o Estado de los Componentes

A continuación, haremos que nuestra App cambie dependiendo de si el usuario "inicia sesión" o no y para esto, usaremos el Hook de Estado de los componentes.

El estado es un concepto fundamental en React y se utiliza para almacenar y gestionar datos que pueden cambiar durante la vida útil de un componente. Cada componente en React puede tener su propio estado. Cuando creas un componente en React, puedes definir un estado inicial utilizando `useState`.

El estado en React es inmutable, lo que significa que debes utilizar funciones para actualizar el estado en lugar de modificarlo directamente.

Hook de UseState o Estado de los Componentes

Por ejemplo, supongamos que estás haciendo una lista de tareas. Cada tarea es un bloque en tu interfaz y necesita almacenar su propio estado, como si está completada o no. Usando `useState`, puedes crear una cajita para cada tarea en la lista. Dentro de esa cajita, guardas el estado de la tarea, como si está completada o no. Inicialmente, la tarea puede estar sin completar, así que el estado sería "false".

Cuando alguien marca la tarea como completada, puedes abrir la cajita y actualizar el estado a "true". Ahora la cajita guarda la información de que la tarea está completada. De esta manera, `useState` te permite tener una caja para cada elemento en tu interfaz, donde puedes guardar y actualizar su propio estado. Así puedes manejar y visualizar el estado de cada elemento de forma individual.

En resumen, `useState` en React es como una caja especial para cada elemento en tu interfaz. Puedes guardar y actualizar el estado de cada elemento dentro de su propia caja, de manera individual.

Hook de UseState o Estado de los Componentes

Para comenzar a usar el estado de los componentes necesitamos importar este hook y lo haremos así dentro de nuestro archivo index, en la misma línea dónde importamos React:

```
1   import React, {useState} from 'react';
```

Luego borraremos la constante "sesión" y dentro de nuestro componente, creamos el estado con la palabra const, seguida de unos corchetes dentro de los que colocaremos el nombre de nuestro estado que será "sesion" y el nombre de la función que usaremos para modificar el mismo que será "cambiarEstadoSesion", seguidamente colocamos un símbolo de igual y al otro lado del igual, la función "useState()" colocando entre los paréntesis el valor que asignaremos por defecto a ese estado, que en este caso será un valor booleano:

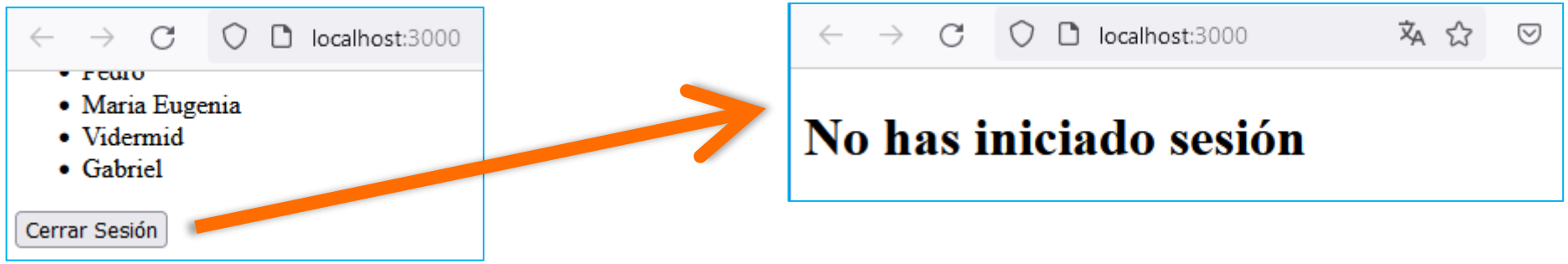
```
5   const App = () =>{
6     const [sesion, cambiarEstadoSesion] = useState(true);
```

Hook de UseState o Estado de los Componentes

Ahora dentro de nuestro componente App, en la condición verdadera de la sesión, crearemos un botón, con el cuál simularemos el cierre de la sesión y en el mismo, dentro de la etiqueta de apertura, colocaremos el evento onClick para llamar a la función de Cambio del estado de la sesión y cambiarle el valor a falso:

```
9       {sesion === true ?
10      <>
11        <Aula/>
12        <button onClick={() => cambiarEstadoSesion(false)}>Cerrar Sesión</button>
13      </>
```

Llegados a este punto, cuándo demos clic en el botón de "Cerrar Sesión" se cambiará el estado de nuestro componente y al hacerlo, se recargará el mismo y mostrará la información correspondiente a su estado falso, es decir, mostrará que no se ha iniciado sesión.



Hook de UseState o Estado de los Componentes

Para concluir esta secuencia, crearemos un botón, con el cuál simularemos el inicio de la sesión y en el mismo, dentro de la etiqueta de apertura, colocaremos el evento onClick para llamar a la función de Cambio del estado de la sesión y cambiarle el valor a verdadero:

```
14        :
15        <>
16          <h1>No has iniciado sesión</h1>
17          <button onClick={() => cambiarEstadoSesion(true)}>Iniciar Sesión</button>
18        </>
19        }
```

Llegados a este punto, cuándo demos clic en el botón de "Iniciar Sesión" se cambiará el estado de nuestro componente y al hacerlo, se recargará el mismo y mostrará la información correspondiente a su estado verdadero, es decir, mostrará el contenido de nuestra App.