REACT CONTEXT

React Context es una forma de administrar el estado globalmente en nuestras aplicaciones.

Se puede usar junto con useState Hook para compartir el estado entre componentes profundamente anidados más fácilmente que con useState solo.

El problema es que el estado debe estar en manos del componente principal más alto en la pila que requiere acceso al estado.

Para ilustrar, tenemos muchos componentes anidados. El componente en la parte superior e inferior de la pila necesita acceso al estado.

Para hacer esto sin contexto, necesitaremos pasar el estado como "props" a través de cada componente anidado. Esto se llama "prop drilling".

```
import { useState } from "react";
import ReactDOM from "react-dom/client";

export default function App() {
 const [user, setUser] = useState("Jesse Hall");

 return (
   <>
     <h1>{`Hello ${user}!`}</h1>
     <Component2 user={user} />
   </>
 );
}

function Component2({ user }) {
 return (
   <>
     <h1>Component 2</h1>
     <Component3 user={user} />
   </>
 );
}

function Component3({ user }) {
 return (
   <>
     <h1>Component 3</h1>
     <Component4 user={user} />
   </>
```

```
  );
}

function Component4({ user }) {
  return (
    <>
      <h1>Component 4</h1>
      <Component5 user={user} />
    </>
  );
}

function Component5({ user }) {
  return (
    <>
      <h1>Component 5</h1>
      <h2>{`Hello ${user} again!`}</h2>
    </>
  );
}
```

Aunque los componentes 2-4 no necesitaban el estado, tenían que pasar el estado para que pudiera llegar al componente 5.

La solución
La solución es crear contexto.

Crear contexto
Para crear un contexto, debe importar createContext e inicializarlo:

A continuación, usaremos el proveedor de contexto para envolver el árbol de componentes que necesitan el contexto de estado.

Proveedor de contexto
Envuelva los componentes secundarios en el proveedor de contexto y proporcione el valor del estado.

Ahora, todos los componentes de este árbol tendrán acceso al contexto del usuario.

Use el gancho useContext
Para usar el Contexto en un componente secundario, necesitamos acceder a él usando el UseContext Hook.

Ahora, todos los componentes de este árbol tendrán acceso al contexto del usuario.

Use el gancho useContext
Para usar el Contexto en un componente secundario, necesitamos acceder a él usando el UseContext Hook.

Primero, incluya useContext en la declaración de importación:

```
import { useState, createContext, useContext } from "react";

const UserContext = createContext();

export default function App() {
 const [user, setUser] = useState("Jesse Hall");

 return (
   <UserContext.Provider value={user}>
     <h1>{`Hello ${user}!`}</h1>
     <Component2 />
   </UserContext.Provider>
 );
}

function Component2() {
 return (
   <>
     <h1>Component 2</h1>
     <Component3 />
   </>
 );
}

function Component3() {
 return (
   <>
     <h1>Component 3</h1>
     <Component4 />
   </>
 );
}

function Component4() {
 return (
   <>
     <h1>Component 4</h1>
     <Component5 />
```

```
    </>
  );
}

function Component5() {
  const user = useContext(UserContext);

  return (
    <>
      <h1>Component 5</h1>
      <h2>{`Hello ${user} again!`}</h2>
    </>
  );
}
```