React - Context

Tags	Context React
Última actualización	

¿Qué es un Context en React?

Un contexto en React es una forma nativa, es decir, no necesitamos de módulos extra, para compartir información a través del árbol de componentes de nuestra aplicación, pero **sin** la necesidad de pasar propiedades de padres a hijos.

Es decir, los Contextos de React vienen a darnos una nueva alternativa a utilizar Flux o Redux. Además, traen consigo varias ventajas:

1. No necesitamos instalar o mantener módulos extra.
2. Cada contexto tiene sus componentes proveedor y consumidor, que nos permiten decidir quién tiene acceso a cuál información y quién puede consumirla. Esto nos permite más modularidad en el manejo de estados. A diferencia de Redux cuyo estándar es envolver la aplicación completa en un solo proveedor y mezclar todos los reductores en un solo estado global.

Pasos para crear un contexto

Como vimos en la sección anterior, los contextos están compuestos de dos partes: un proveedor y un consumidor.

El proveedor del contexto es el componente que inicializa los valores y cómo *high order component* que es también inyecta dichos valores a sus hijos.

El consumidor es quien expone las propiedades inyectadas por el consumidor a los componentes, normalmente esta parte se usa con un hook llamado

useContext.

Para crear un únicamente necesitamos de la función de createContext de React y la llamamos con el valor por defecto que queremos que tenga nuestro contexto.

Si vamos a utilizar varios contextos en nuestra aplicación es mejor exponer nosotros mismos nuevos hooks que utilicen nuestro contexto específico, un ejemplo:

```
import axios from 'axios';
import React, {createContext, useContext, useEffect, useState} from 'react';

const TodosContext = createContext({
  todos: [],
  todoSeleccionado: null,
  setTodos: () => {},
  setTodoSeleccionado: () => {},
  fetchTodos: () => new Promise(() => {}),
});

export const TodosProvider = ({children}) => {
  const [todos, setTodos] = useState<ITodo[]>([]);
  const [todoSeleccionado, setTodoSeleccionado] = useState<ITodo | null>(null);

  const fetchTodos = async () => {
    try {
      const todos = await axios.get(
        'https://jsonplaceholder.typicode.com/todos',
      );

      setTodos(todos.data);
    } catch (error) {
      console.log(error);
    }
  };

  useEffect(() => {
    async function callFetchTodos() {
      await fetchTodos();
    }

    callFetchTodos();
  }, []);
```

```
  const val = {
    todos,
    todoSeleccionado,
    setTodos,
    setTodoSeleccionado,
    fetchTodos,
  };
  return <TodosContext.Provider value={val}>{children}</TodosContext.Provider>;
};

export const useTodos = () => useContext(TodosContext);
```