**Refactor từng bước** từ useState → useReducer, nhưng **component không cần thay đổi**

**Giỏ hàng cũ**

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

// 1. Tạo context

const CartContext = createContext();

// 2. Provider bao bọc toàn app

export const CartProvider = ({ children }) => {

  const [isCartOpen, setIsCartOpen] = useState(false);

  const [cartItems, setCartItems] = useState([]); // ✅ thêm dòng này

  const openCart = () => setIsCartOpen(true);

  const closeCart = () => setIsCartOpen(false);

  // ✅ Hàm thêm sản phẩm

  const addToCart = (product) => {

    setCartItems((prevItems) => {

      const existingItem = prevItems.find((item) => item.id === product.id);

      if (existingItem) {

        // Nếu đã có sản phẩm → tăng số lượng

        return prevItems.map((item) =>

          item.id === product.id

            ? { ...item, quantity: item.quantity + 1 }

            : item

        );

      } else {

        // Nếu chưa có → thêm mới

        return [...prevItems, { ...product, quantity: 1 }];

      }

    });

  };

  // Xóa sản phẩm khỏi giỏ hàng

  const removeFromCart = (id) => {

    setCartItems((prevItems) => prevItems.filter((item) => item.id !== id));

  };

  // Cập nhật sản phẩm vào giỏ hàng

  const updateQuantity = (id, newQuantity) => {

    setCartItems((prevItems) =>

      prevItems.map((item) =>

        item.id === id ? { ...item, quantity: newQuantity } : item

      )

    );

  };

  return (

    <CartContext.Provider

      value={{

        isCartOpen,

        openCart,

        closeCart,

        cartItems,

        addToCart,

        removeFromCart,

        updateQuantity, // 👈 THÊM DÒNG NÀY

      }} // ✅ truyền thêm

    >

      {children}

    </CartContext.Provider>

  );

};

// 3. Custom hook cho tiện

export const useCart = () => useContext(CartContext);

**Giỏ hàng mới**

import React, { createContext, useContext, useReducer, useState } from "react";

// 1. Tạo context

const CartContext = createContext();

// 🎯 Reducer quản lý logic thêm / sửa / xoá sản phẩm

const cartReducer = (state, action) => {

switch (action.type) {

case "ADD\_TO\_CART": {

const product = action.payload;

const existingItem = state.find((item) => item.id === product.id);

if (existingItem) {

// Nếu đã có sản phẩm → tăng số lượng

return state.map((item) =>

item.id === product.id

? { ...item, quantity: item.quantity + 1 }

: item

);

} else {

// Nếu chưa có → thêm mới

return [...state, { ...product, quantity: 1 }];

}

}

case "REMOVE\_FROM\_CART": {

// Xóa sản phẩm khỏi giỏ hàng

return state.filter((item) => item.id !== action.payload);

}

case "UPDATE\_QUANTITY": {

const { id, quantity } = action.payload;

// Cập nhật số lượng sản phẩm

return state.map((item) =>

item.id === id ? { ...item, quantity } : item

);

}

default:

return state;

}

};

// 2. Provider bao bọc toàn app

export const CartProvider = ({ children }) => {

const [isCartOpen, setIsCartOpen] = useState(false);

const [cartItems, dispatch] = useReducer(cartReducer, []); // ✅ dùng reducer thay vì useState

const openCart = () => setIsCartOpen(true);

const closeCart = () => setIsCartOpen(false);

// ✅ Hàm thêm sản phẩm

const addToCart = (product) => {

dispatch({ type: "ADD\_TO\_CART", payload: product });

};

// Xóa sản phẩm khỏi giỏ hàng

const removeFromCart = (id) => {

dispatch({ type: "REMOVE\_FROM\_CART", payload: id });

};

// Cập nhật sản phẩm vào giỏ hàng

const updateQuantity = (id, newQuantity) => {

dispatch({

type: "UPDATE\_QUANTITY",

payload: { id, quantity: newQuantity },

});

};

return (

<CartContext.Provider

value={{

isCartOpen,

openCart,

closeCart,

cartItems,

addToCart,

removeFromCart,

updateQuantity, // 👈 THÊM DÒNG NÀY

}}

>

{children}

</CartContext.Provider>

);

};

// 3. Custom hook cho tiện

export const useCart = () => useContext(CartContext);