import { createContext, useEffect, useState } from "react";

import PropTypes from "prop-types";

import axios from "axios";

export const ShopContext = createContext();

// Function to set the default cart state based on products

const getDefaultCart = (products) => {

    let cart = {};

    // Initialize cart with all items set to 0

    products.forEach((product, index) => {

        cart[product.id] = 0;

    });

    return cart;

};

const ShopContextProvider = ({ children }) => {

    const [cartItems, setCartItems] = useState({});

    const [products, setProducts] = useState([]);

    const [loading, setLoading] = useState(true);

    const [error, setError] = useState(null);

    useEffect(() => {

        fetchData();

    }, []);  // Fetch products when the component mounts

    // Function to fetch data from API

    async function fetchData() {

        try {

            const response = await axios.get("https://fakestoreapi.com/products");

            setProducts(response.data);

            setCartItems(getDefaultCart(response.data));  // Initialize cart items based on fetched products

            setLoading(false);

        } catch (err) {

            setError("Failed to fetch data");

            setLoading(false);

        }

    }

    if (loading) {

        return <h1>Loading...</h1>;

    }

    if (error) {

        return <h1>{error}</h1>;

    }

    // Function to add an item to the cart

    const addToCart = (itemId) => {

        setCartItems((prev) => ({ ...prev, [itemId]: prev[itemId] + 1 }));

        console.log(cartItems);

    };

    // Function to remove an item from the cart

    const removeFromCart = (itemId) => {

        setCartItems((prev) => ({ ...prev, [itemId]: prev[itemId] - 1 }));

    };

    // Function to get total cart amount

    const getTotalCartAmmount = () => {

        let totalAmount = 0;

        // Iterate through the cart items to calculate the total amount

        for (const item in cartItems) {

            if (cartItems[item] > 0) {

                // Find the product matching the item ID

                let itemInfo = products.find((product) => product.id === Number(item));

                totalAmount += itemInfo.price \* cartItems[item];  // Calculate total

            }

        }

        return totalAmount;

    };

    // Function to get the total number of items in the cart

    const getTotalCartItems = () => {

        let totalItems = 0;

        // Sum all cart item counts

        for (const item in cartItems) {

            if (cartItems[item] > 0) {

                totalItems += cartItems[item];

            }

        }

        return totalItems;

    };

    // Context value that will be shared with consumers

    const contextValue = {

        getTotalCartItems,

        getTotalCartAmmount,

        products,

        cartItems,

        addToCart,

        removeFromCart,

    };

    return (

        <ShopContext.Provider value={contextValue}>

            {children}

        </ShopContext.Provider>

    );

};

// Prop type validation

ShopContextProvider.propTypes = {

    children: PropTypes.node.isRequired,

};

export default ShopContextProvider;

with local storage

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

import PropTypes from "prop-types";

import axios from "axios";

// Create the context

export const ShopContext = createContext();

// Function to get the default cart state from localStorage

const getDefaultCart = () => {

    const storedCart = JSON.parse(localStorage.getItem("cartItems"));

    if (storedCart) {

        return storedCart;

    }

    return {}; // Return empty cart if nothing is stored

};

// ShopContextProvider component to manage cart and products

const ShopContextProvider = ({ children }) => {

    const [cartItems, setCartItems] = useState(getDefaultCart());

    const [products, setProducts] = useState([]);

    const [loading, setLoading] = useState(true);

    const [error, setError] = useState(null);

    // Fetch products from API

    useEffect(() => {

        axios.get("https://fakestoreapi.com/products")

            .then(response => {

                setProducts(response.data);

                setLoading(false);

            })

            .catch(err => {

                console.error("Failed to fetch products", err);

                setError("Failed to fetch data");

                setLoading(false);

            });

    }, []);

    // Always call the useEffect hook, but do not call conditionally.

    useEffect(() => {

        // Save cart items to localStorage whenever cartItems state changes

        if (Object.keys(cartItems).length > 0) {

            localStorage.setItem("cartItems", JSON.stringify(cartItems));

        }

    }, [cartItems]);

    // Loading and error state handling

    if (loading) {

        return <h1>Loading...</h1>;

    }

    if (error) {

        return <h1>{error}</h1>;

    }

    // Add item to the cart

    const addToCart = (itemId) => {

        setCartItems(prev => ({

            ...prev,

            [itemId]: (prev[itemId] || 0) + 1

        }));

    };

    // Remove item from the cart

    const removeFromCart = (itemId) => {

        setCartItems(prev => {

            const newCart = { ...prev };

            if (newCart[itemId] > 0) {

                newCart[itemId] -= 1;

            }

            return newCart;

        });

    };

    // Optimized getTotalCartAmmount function

    const getTotalCartAmmount = () => {

        let totalAmount = 0;

        // Create a map of products for fast lookup

        const productMap = products.reduce((acc, product) => {

            acc[product.id] = product;

            return acc;

        }, {});

        // Iterate through the cart items and calculate total

        for (const item in cartItems) {

            if (cartItems[item] > 0) {

                const itemInfo = productMap[item];

                if (itemInfo) {

                    totalAmount += itemInfo.price \* cartItems[item];

                }

            }

        }

        return totalAmount;

    };

    // Function to get the total number of items in the cart

    const getTotalCartItems = () => {

        let totalItems = 0;

        for (const item in cartItems) {

            if (cartItems[item] > 0) {

                totalItems += cartItems[item];

            }

        }

        return totalItems;

    };

    // The context value to be provided

    const contextValue = {

        cartItems,

        products,

        addToCart,

        removeFromCart,

        getTotalCartItems,

        getTotalCartAmmount

    };

    return (

        <ShopContext.Provider value={contextValue}>

            {children}

        </ShopContext.Provider>

    );

};

// Prop type validation

ShopContextProvider.propTypes = {

    children: PropTypes.node.isRequired,

};

export default ShopContextProvider;