MODULE 3: SHOPPING CART FUNCTIONALITY

LESSON 9: Persistent Cart with Local Storage

Learning Objective

By the end of this lesson, you will understand how to persist cart data across browser sessions using localStorage.

9.1 How localStorage Works

localStorage is a web API that lets you store data in the user's browser. Data persists even after the browser is closed.

Key Methods:

- `localStorage.setItem(key, value)` - Save data

- `localStorage.getItem(key)` - Retrieve data
- `localStorage.removeItem(key)` - Delete data
- `localStorage.clear()` - Delete all data

Important: localStorage can only store strings. Use `JSON.stringify()` and `JSON.parse()` for objects/arrays.

9.2 Updating CartContext with localStorage

We already implemented this in Lesson 7, but let's understand it better:

javascript

```javascript
// Load cart from localStorage on initial mount
useEffect(() => {
  const savedCart = localStorage.getItem('ceylonmart-cart');
  if (savedCart) {
    setCart(JSON.parse(savedCart));
  }
  setIsInitialized(true);
}, []);

// Save cart to localStorage whenever it changes
useEffect(() => {
  if (isInitialized) {
    localStorage.setItem('ceylonmart-cart', JSON.stringify(cart));
  }
}, [cart, isInitialized]);
```

Why this pattern?

1. First effect runs once when component mounts
2. It loads saved cart data

3. Sets `isInitialized` to true
4. Second effect only saves after initialization
5. This prevents overwriting saved data on first load

9.3 Testing Persistence

Let's test if our cart persists:

Step 1: Add items to cart
Step 2: Close browser completely
Step 3: Open browser and navigate back to site
Step 4: Cart should still have items

✅ Success! Cart data persists.

9.4 Adding Cart Expiry (Optional)

Sometimes you want cart to expire after a certain time. Let's add this feature.

Step 1: Update CartContext.js with expiry

javascript

```javascript
"use client";

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

const CartContext = createContext();

export function useCart() {
```

```
  return useContext(CartContext);
}

export function CartProvider({ children }) {
  const [cart, setCart] = useState([]);
  const [isInitialized, setIsInitialized] = useState(false);
  const CART_EXPIRY_DAYS = 7; // Cart expires after 7 days

  useEffect(() => {
    // Load cart with expiry check
    const savedCartData = localStorage.getItem('ceylonmart-cart');

    if (savedCartData) {
      const { cart: savedCart, timestamp } = JSON.parse(savedCartData);
      const now = new Date().getTime();
      const expiryTime = CART_EXPIRY_DAYS * 24 * 60 * 60 * 1000; // Convert days to
milliseconds

      // Check if cart has expired
      if (now - timestamp < expiryTime) {
        setCart(savedCart);
      } else {
        // Cart expired, clear it
        localStorage.removeItem('ceylonmart-cart');
      }
    }

    setIsInitialized(true);
  }, []);

  useEffect(() => {
    if (isInitialized) {
      // Save cart with timestamp
      const cartData = {
        cart: cart,
        timestamp: new Date().getTime()
      };
```

```
    localStorage.setItem('ceylonmart-cart', JSON.stringify(cartData));
  }
}, [cart, isInitialized]);

// ... rest of cart functions remain the same
}
```

9.5 Adding Multiple Carts for Different Users (Bonus)

If your site has user accounts, you might want separate carts per user.

javascript

```
// When user logs in
const saveCartForUser = (userId, cart) => {
  localStorage.setItem(`cart-${userId}`, JSON.stringify(cart));
};

// When user logs out
const clearCurrentCart = () => {
  setCart([]);
  localStorage.removeItem('ceylonmart-cart');
};
```

9.6 Debugging localStorage

You can inspect localStorage in browser DevTools:

Chrome:

1. Open DevTools (F12)
2. Go to "Application" tab
3. Look for "Local Storage" in left sidebar
4. Click on your site URL
5. See all stored data

Firefox:

1. Open DevTools (F12)
2. Go to "Storage" tab
3. Look for "Local Storage"

9.7 Common localStorage Mistakes

Mistake	Why It's Wrong	Correct Way
Storing objects directly	localStorage only accepts strings	Use `JSON.stringify()`
Forgetting to parse	Getting back a string, not object	Use `JSON.parse()`

Not checking if data exists	May get null and cause errors	Always check: `if (data) {...}`
Storing sensitive data	localStorage is not secure	Never store passwords, tokens
Infinite loops	Setting state in useEffect without dependency array	Use proper dependencies

---

Lesson 9 Summary

- localStorage persists data across browser sessions
- Always stringify before saving, parse after loading
- Add timestamps for expiry functionality
- Debug using browser DevTools
- Never store sensitive information

Practice Exercises

1. Add "Clear Cart" confirmation with a modal dialog
2. Implement cart versioning (in case cart structure changes in future updates)
3. Add multiple cart support for different stores/sessions
4. Challenge: Sync cart across tabs using `storage` event listener