ASSIGNMENT NO: 01

Name: Rimsha Aqeel

Roll no: Sp22-Bse-040

Subject: Mobile Application Development

Submitted to: Sir Kamran

Submission Date: September 26, 2024

Overview

The shopping cart is implemented as a class called `ShoppingCart`, which encapsulates all the functionality needed to manage a shopping cart in a mobile application. The class uses ES6 features such as arrow functions, array methods (`map`, `filter`, `reduce`), and object manipulation to provide a clean and efficient way to handle products.

Key Components

1. Class Definition:

-The `ShoppingCart` class is defined to hold an array called `cart`, which will store product objects.

2. Adding Items:

- **Function**: `addItem(productId, productName, quantity, price)`

- **Description**: This function creates a new product object with properties: `productId`, `productName`, `quantity`, and `price`. It then adds this object to the `cart` array using the `push` method.

- **Usage**: When called, it logs a message indicating that the product has been added to the cart.

3. Removing Items:

- **Function**: `removeItem(productId)`

- **Description**: This function removes an item from the cart based on its `productId`. It uses the `filter` method to create a new array that excludes the item with the specified ID.

- **Usage**: After filtering, it checks if any items were removed and logs an appropriate message.

4. Updating Quantity:

- **Function**: `updateQuantity(productId, newQuantity)`

- **Description**: This function searches for a product in the cart by its ID using the `find` method. If found, it updates the product's quantity property.

- **Usage**: It logs a message confirming that the quantity has been updated or indicating that the product was not found.

5. Calculating Total Cost:

- **Function**: `calculateTotalCost()`

- **Description**: This function calculates the total cost of all items in the cart using the `reduce` method. It multiplies each item's price by its quantity and accumulates these values into a total.

- **Usage**: It returns the total cost as a number.

6. Displaying Cart Summary:

- **Function**: `displayCartSummary()`

- **Description**: This function generates a summary of all items in the cart using the `map` method. It creates an array of objects containing each item's name, quantity, and total price (price multiplied by quantity).

- **Usage**: It logs each item's details and also displays the overall total cost by calling `calculateTotalCost()`.

7. Filtering Zero Quantity Items:

- **Function**: `filterZeroQuantityItems()`

- **Description**: This function removes any items from the cart that have a quantity of zero using the `filter` method.

- **Usage**: It updates the cart and logs a message indicating that zero-quantity items have been filtered out.

8. Applying Discount Codes (Bonus Feature):

- **Function**: `applyDiscountCode(code)`

- **Description**: This optional function allows users to apply discount codes (e.g., "SAVE10" for 10% off). It checks if the provided code is valid and calculates the discounted total based on the current total cost.

- **Usage**: If valid, it logs the discounted total; otherwise, it indicates an invalid code.

How It All Works Together

- The class provides a structured way to manage products in a shopping cart through various methods.

- Users can add products, remove them, update quantities, view summaries, filter out unwanted items, and even apply discounts.

- Each method utilizes modern JavaScript features for clarity and efficiency:

- Arrow functions make it easy to define methods without losing context (`this`).

- Array methods (`map`, `filter`, `reduce`) allow for concise manipulation of arrays without needing explicit loops.

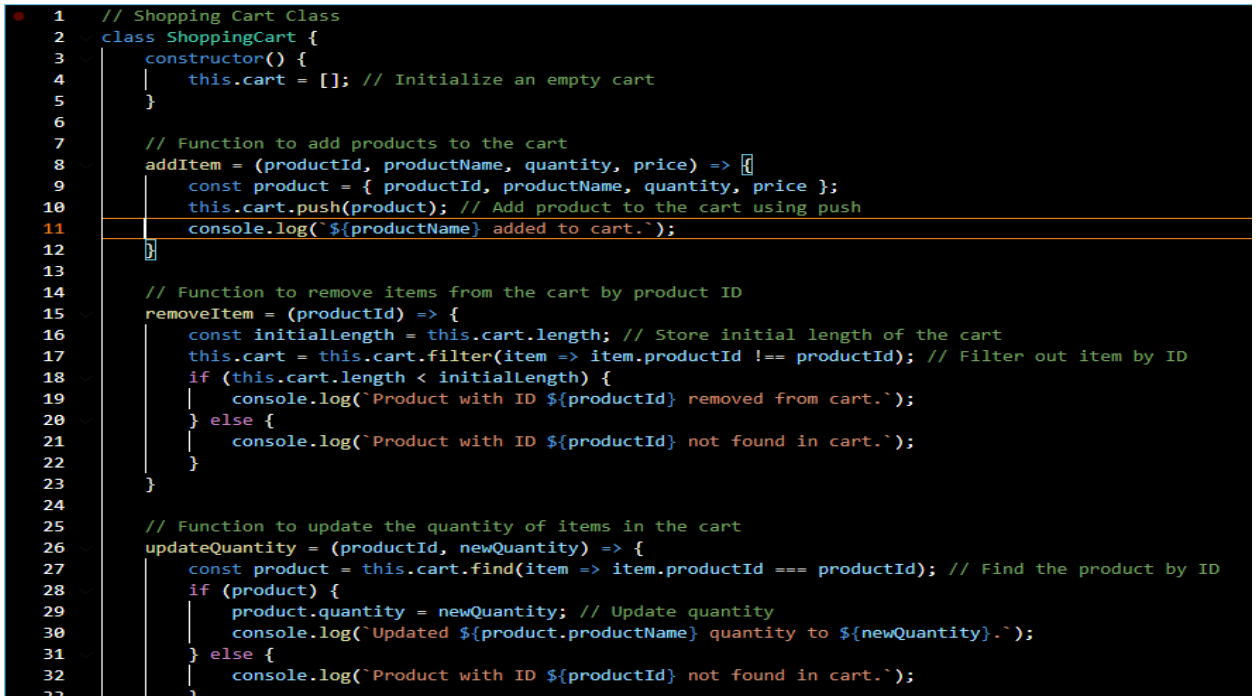
- The implementation ensures that all actions are logged for user feedback, enhancing user experience in managing their shopping cart.

Example Usage

At the end of the code snippet, there is an example usage section where an instance of `ShoppingCart` is created (`myCart`). Various methods are called on this instance to demonstrate adding items, updating quantities, removing items, displaying summaries, filtering zero quantities, and applying discount codes. This showcases how all functionalities can be utilized seamlessly in practice.

Overall, this implementation provides a robust foundation for a mobile shopping cart feature while adhering to modern JavaScript best practices.

CODE

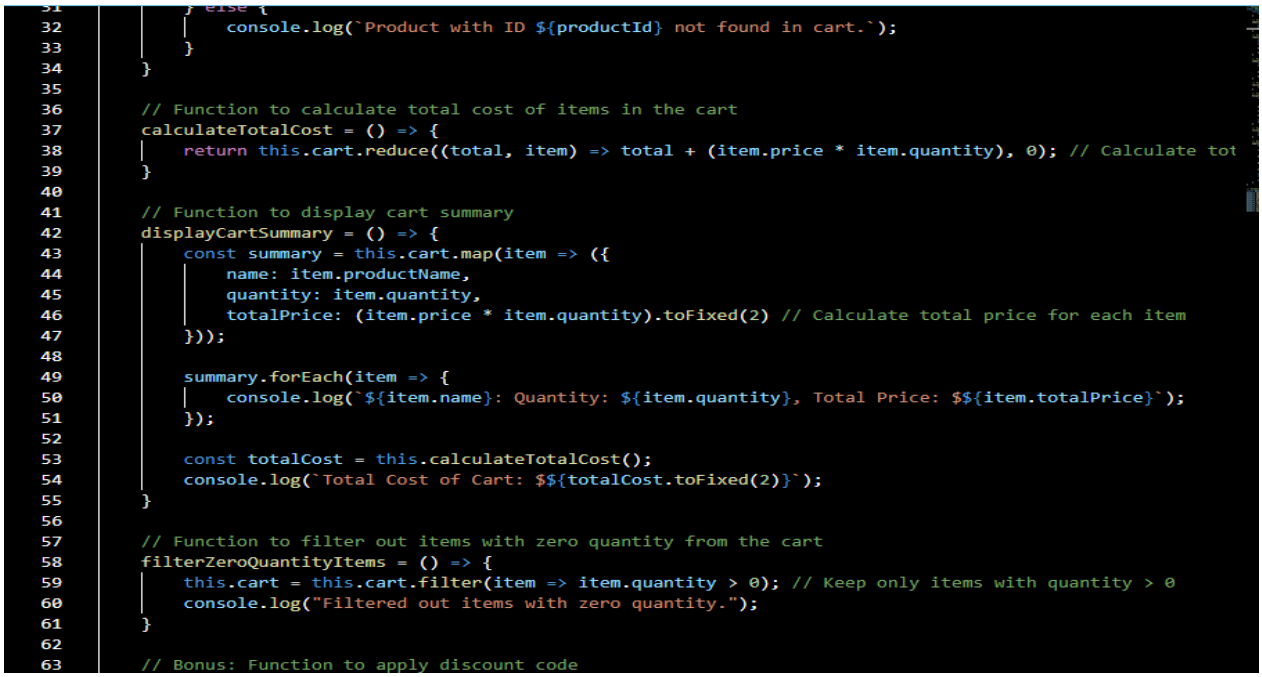


```
// Shopping Cart Class
class ShoppingCart {
    constructor() {
        this.cart = []; // Initialize an empty cart
    }

    // Function to add products to the cart
    addItem = (productId, productName, quantity, price) => {
        const product = { productId, productName, quantity, price };
        this.cart.push(product); // Add product to the cart using push
        console.log(`${productName} added to cart.`);
    }

    // Function to remove items from the cart by product ID
    removeItem = (productId) => {
        const initialLength = this.cart.length; // Store initial length of the cart
        this.cart = this.cart.filter(item => item.productId !== productId); // Filter out item by ID
        if (this.cart.length < initialLength) {
            console.log(`Product with ID ${productId} removed from cart.`);
        } else {
            console.log(`Product with ID ${productId} not found in cart.`);
        }
    }

    // Function to update the quantity of items in the cart
    updateQuantity = (productId, newQuantity) => {
        const product = this.cart.find(item => item.productId === productId); // Find the product by ID
        if (product) {
            product.quantity = newQuantity; // Update quantity
            console.log(`Updated ${product.productName} quantity to ${newQuantity}.`);
        } else {
            console.log(`Product with ID ${productId} not found in cart.`);
```



```
            console.log(`Product with ID ${productId} not found in cart.`);
        }
    }

    // Function to calculate total cost of items in the cart
    calculateTotalCost = () => {
        return this.cart.reduce((total, item) => total + (item.price * item.quantity), 0); // Calculate tot
    }

    // Function to display cart summary
    displayCartSummary = () => {
        const summary = this.cart.map(item => ({
            name: item.productName,
            quantity: item.quantity,
            totalPrice: (item.price * item.quantity).toFixed(2) // Calculate total price for each item
        }));

        summary.forEach(item => {
            console.log(`${item.name}: Quantity: ${item.quantity}, Total Price: $${item.totalPrice}`);
        });

        const totalCost = this.calculateTotalCost();
        console.log(`Total Cost of Cart: $${totalCost.toFixed(2)}`);
    }

    // Function to filter out items with zero quantity from the cart
    filterZeroQuantityItems = () => {
        this.cart = this.cart.filter(item => item.quantity > 0); // Keep only items with quantity > 0
        console.log("Filtered out items with zero quantity.");
    }

    // Bonus: Function to apply discount code
```

```
64      applyDiscountCode = (code) => {
65          let discount = 0;
66
67          if (code === "SAVE10") {
68              discount = 0.10; // 10% discount
69              console.log("10% discount applied.");
70          } else if (code === "SAVE20") {
71              discount = 0.20; // 20% discount
72              console.log("20% discount applied.");
73          } else {
74              console.log("Invalid discount code.");
75              return;
76          }
77
78          const totalCost = this.calculateTotalCost();
79          const discountedTotal = totalCost - (totalCost * discount);
80
81          console.log(`Total Cost after discount: $${discountedTotal.toFixed(2)}`);
82      }
83  }
84
85  // Example Usage:
86  const myCart = new ShoppingCart();
87  myCart.addItem(1, "Apple", 3, 0.50);
88  myCart.addItem(2, "Banana", 2, 0.30);
89  myCart.addItem(3, "Orange", 1, 0.80);
90  myCart.displayCartSummary();
91  myCart.updateQuantity(1, 5);
92  myCart.removeItem(2);
93  myCart.filterZeroQuantityItems();
94  myCart.displayCartSummary();
95  myCart.applyDiscountCode("SAVE10");
```

OUTPUT

```
"C:\Program Files\nodejs\node.exe" ".\extensions\MAD assignment1.js"
Apple added to cart.
Banana added to cart.
Orange added to cart.
Apple: Quantity: 3, Total Price: $1.50
Banana: Quantity: 2, Total Price: $0.60
Orange: Quantity: 1, Total Price: $0.80
Total Cost of Cart: $2.90
Updated Apple quantity to 5.
Product with ID 2 removed from cart.
Filtered out items with zero quantity.
Apple: Quantity: 5, Total Price: $2.50
Orange: Quantity: 1, Total Price: $0.80
Total Cost of Cart: $3.30
10% discount applied.
Total Cost after discount: $2.97
```