

**TITLE:**

**SHOPPING CART IMPLEMENTATION**

**ASSIGNMENT NO 01**

**MOBILE APPLICATION DEVELOPMENT**

**PROGRAM: BSSE-6**

---



**SUBMITTED TO:**

**SIR MUHAMMAD KAMRAN**

**SUBMITTED BY:**

**ZARNAB IMRAN (sp22-bse-058)**

**DATE: 26<sup>TH</sup>, SEP 2024**

## **INTRODUCTION:**

### **OBJECTIVE:**

The objective of this assignment is to develop a **JavaScript-based mobile shopping cart** using ES6 features, specifically arrow functions, array methods (map, filter, reduce), and object manipulation techniques. The cart will manage products by adding, removing, updating, and calculating the total price while also offering an optional discount feature.

### **OPERATIONS IMPLEMENTED:**

#### **1. Add Items to the Cart:**

- **Objective:** Add products to the shopping cart.
- **Implementation:** A function using the push method to add a product object (containing productId, productName, quantity, and price) into the cart array.

#### **2. Remove and Update Items:**

- **Objective:** Remove and update products in the cart.
- **Remove Function:** Uses splice to remove an item from the cart by finding its productId.
- **Update Quantity Function:** Utilizes map to update the quantity of an item by its productId and adjust the cart accordingly.

#### **3. Calculate Total Cost:**

- **Objective:** Calculate the total price of all products in the cart.
- **Implementation:** The reduce method is used to calculate the total cost by summing the product of price and quantity for each item.

#### **4. Display Cart Summary:**

- **Objective:** Display a summary of the cart contents.
- **Implementation:** The map method generates a summary of each product (name, quantity, total price per product). A filter function is also included to remove items with zero quantity.

#### **5. Bonus: Apply Discount Code:**

- **Objective:** Apply a discount to the total price based on a given code.
- **Implementation:** Uses object manipulation to validate discount codes (e.g., 'DISCOUNT10' for 10%) and applies the respective percentage discount to the total price.

Each function is implemented using **ES6 arrow functions** to ensure modern, concise, and readable code.

## **CODE EXPLANATION:**

### **1. addItemToCart(productId, productName, quantity, price)**

- **Purpose:** Adds a product to the cart.
- **Logic:**

1. A product object is created using the parameters `productId`, `productName`, `quantity`, and `price`.
2. This object is then pushed into the cart array using the `push` method.
3. A message confirming that the product has been added is logged to the console.

## 2. `removeItemFromCart(productId)`

- **Purpose:** Removes a product from the cart based on its `productId`.
- **Logic:**
  - ❖ The function first finds the index of the product in the cart using the `findIndex` method, which compares each item's `productId`.
  - ❖ If a product with the given `productId` exists, it is removed using the `splice` method, which removes one item at the found index.
  - ❖ The name of the removed item is logged to the console.
  - ❖ If the `productId` is not found, an error message is logged.

## 3. `updateItemQuantity(productId, newQuantity)`

- **Purpose:** Updates the quantity of a product in the cart.
- **Logic:**
  - ❖ The function uses `map` to iterate over the cart array.
  - ❖ If the `productId` of an item matches the input `productId`, a new product object is returned with the updated quantity.
  - ❖ If no match is found, the original item is returned unchanged.
  - ❖ If an item with the given `productId` is found, a success message is logged. Otherwise, an error message is shown.

## 4. `calculateTotalCost()`

- **Purpose:** Calculates the total cost of the items in the cart.
- **Logic:**
  - ❖ The function uses the `reduce` method to iterate over the cart array and accumulates the total price by multiplying each item's price by its quantity.
  - ❖ The final total cost is returned, and it is also logged to the console in a formatted string.

## 5. `displayCartSummary()`

- **Purpose:** Displays a summary of the cart's contents.
- **Logic:**
  - ❖ The function iterates over the cart array using `forEach` and logs each product's `productName`, `quantity`, and the total price (`price * quantity`) to the console.

- ❖ The toFixed(2) method is used to format the total price to two decimal places.

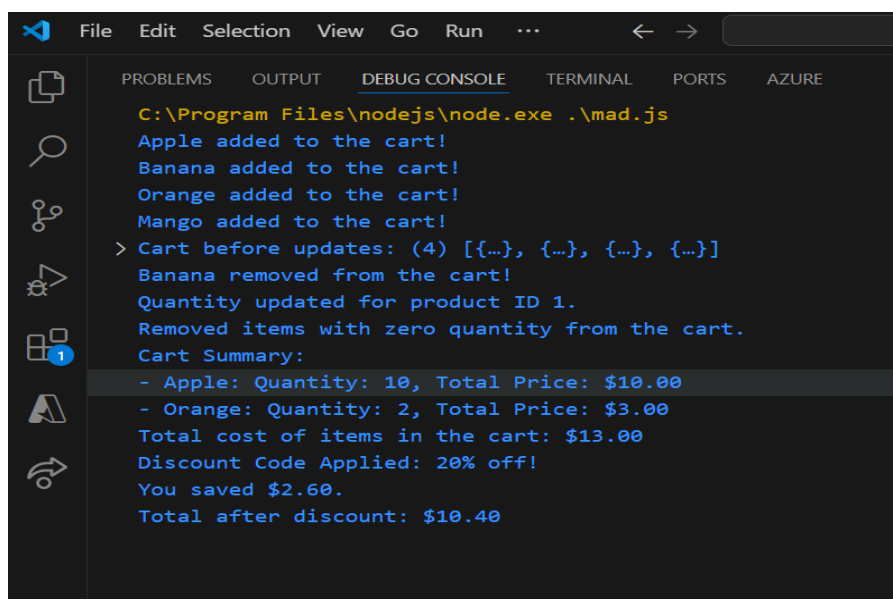
## 6. removeZeroQuantityItems()

- **Purpose:** Filters out items from the cart that have a quantity of zero.
- **Logic:**
  - ❖ The function uses the filter method to create a new array that only includes items with a quantity greater than zero.
  - ❖ The cart is updated to this filtered array, and a message is logged to the console.

## 7. applyDiscount(discountCode)

- **Purpose:** Applies a discount to the total price based on the input discount code.
- **Logic:**
  - ❖ The function first checks if the provided discountCode matches any predefined discount codes (e.g., DISCOUNT10, DISCOUNT20, etc.) from an object.
  - ❖ If the code is valid, the corresponding discount percentage is applied to the total cost, which is calculated by calling calculateTotalCost().
  - ❖ The discount amount is calculated by multiplying the total cost by the discount percentage divided by 100. This is subtracted from the total cost to get the final discounted total.
  - ❖ The original total, discount applied, and final total are logged to the console.
  - ❖ If the discount code is invalid, an error message is shown.

## SCREENSHOT:



```
C:\Program Files\nodejs\node.exe .\mad.js
Apple added to the cart!
Banana added to the cart!
Orange added to the cart!
Mango added to the cart!
> Cart before updates: (4) [{...}, {...}, {...}, {...}]
Banana removed from the cart!
Quantity updated for product ID 1.
Removed items with zero quantity from the cart.
Cart Summary:
- Apple: Quantity: 10, Total Price: $10.00
- Orange: Quantity: 2, Total Price: $3.00
Total cost of items in the cart: $13.00
Discount Code Applied: 20% off!
You saved $2.60.
Total after discount: $10.40
```

## CONCLUSION:

This assignment provided valuable insights into effectively managing a shopping cart system using JavaScript. I learned to utilize arrays and objects to represent products and their properties, reinforcing my understanding of data structures. The implementation of functional

programming concepts, such as map, filter, and reduce, allowed for cleaner and more maintainable code.

The importance of user interaction was highlighted through the design of functions that provide clear feedback on actions taken, such as adding or removing items and applying discounts. This aspect reinforced the necessity of robust error handling to enhance the user experience and prevent unwanted behavior.

Additionally, implementing discount logic showcased the flexibility of JavaScript for integrating business rules into applications, though it also raised considerations about potential future complexities.

Challenges such as managing the cart's state, handling zero quantity items, and ensuring the correct implementation of discount features required careful thought and planning. Testing each functionality was crucial but time-consuming, emphasizing the importance of thorough validation.

Overall, this assignment not only improved my programming skills but also underscored the significance of writing code that is both functional and readable. It equipped me with the tools and knowledge to tackle similar projects in the future, while the challenges faced served as valuable learning opportunities, reinforcing critical thinking about software design and functionality.