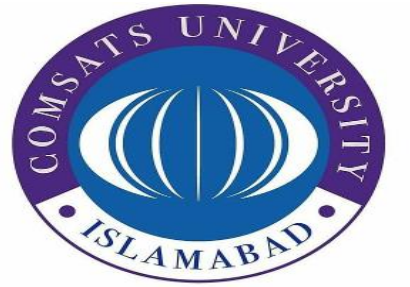
COMSATS UNIVERSITY ISLAMABAD

(ATTOCK CAMPUS)



:Assignment No 01:

Submitted By:

Laiba Faryal

Reg No:

Sp22-Bse-050

Course:

Mobile App Development

Date:

26,sep 2024.

Submitted To:

Sir Muhammad Kamran

INTRODUCTION:

The objective of this code is to implement a **Shopping cart feature** for a mobile or web-based e-commerce application using modern JavaScript (ES6+). The code allows users to:

- Add products to the cart.
- Update or remove items in the cart.
- Calculate the total cost of the items.
- Display a summary of the cart.
- Optionally, apply a discount to the total price.

Operations Implemented:

1. Adding Items to the Cart (`addItemToCart`):
- Adds a product (object with `productId`, `productName`, `quantity`, and `price`) to the cart array.
- If the product already exists in the cart, it updates the quantity instead of duplicating the entry.

2. Removing Items from the Cart (`removeItemFromCart`):
- Removes a product from the cart by locating it using its `productId` and then using the `splice` method to delete it.

3. Updating Item Quantity (`updateItemQuantity`):
- Updates the quantity of a specific product in the cart using `map` to create a new array where only the quantity of the matching product is modified.

4. Calculating Total Cost (`calculateTotalCost`):
- Uses `reduce` to calculate the total price of all items in the cart, considering the quantity and price of each product.

5. Displaying Cart Summary(`displayCartSummary`):
- Uses `map` to create a summary of the cart, displaying each product's name, quantity, and the total price for that product.

6. Filtering Zero Quantity Items (`filterZeroQuantityItems`):
- Filters out and removes items with zero quantity from the cart using the `filter` method.

7. Applying Discount (`applyDiscount`):
- Optionally applies a discount based on a discount code. The total cost is reduced by a percentage defined by the discount code.

Key Concepts:

Arrow Functions: Used for cleaner, more concise function definitions.
Array Methods: (`push`, `map`, `filter`, `reduce`, `splice`): Used for manipulating the cart array in various ways.
Object Manipulation: Each product in the cart is an object, and object properties are accessed/updated based on the operations needed.

Code Explanation:

Logic behind each function….

1. AddItemToCart:
- **Logic:** This function checks if the item is already in the cart by comparing the `productId`. If the item exists, it updates the quantity by adding the new quantity to the existing one. If not, it creates a new product object and adds it to the `cart` array using the `push` method.

2. RemoveItemFromCart:
- **Logic:** Finds the index of the product in the cart using `findIndex` based on the `productId`. If the item is found, it removes the product from the cart array using the `splice` method, which deletes the item at the specified index.

3. UpdateItemQuantity:
- **Logic:** Iterates over the `cart` array using `map` to create a new array. If the `productId` matches, it updates the `quantity` of the product by returning a new object with the updated quantity. For all other products, it returns the same object.

4. CalculateTotalCost:
- **Logic:** Uses `reduce` to iterate through the `cart` and calculates the total cost. For each product, it multiplies the `price` by the `quantity` and adds this value to an accumulating total. The result is the total price of all items in the cart.

5. DisplayCartSummary:
- **Logic:** Uses `map` to iterate over the `cart` and creates an array of summary objects, where each object contains the `productName`, `quantity`, and `totalPrice` (calculated as `price * quantity`). It then logs or returns this summary.

6. FilterZeroQuantityItems:

- **Logic:** Uses `filter` to create a new array that excludes products where `quantity` is `0`. The cart is then updated to this new filtered array, effectively removing any items with zero quantity.

7. ApplyDiscount:

- **Logic:** Checks if a valid discount code is passed by looking it up in the `discounts` object. It then calculates the discount (as a percentage of the total cost) and subtracts this discount from the total price. If the code is invalid, no discount is applied.

Code:

```
// Shopping cart array to hold product items
let cart = [];

// 1. Add Items to the Cart
const addItemToCart = (productId, productName, quantity, price) => {
    // Check if item already exists in the cart
    const existingProduct = cart.find(item => item.productId === productId);
    if (existingProduct) {
        // Update the quantity if product already exists
        existingProduct.quantity += quantity;
    } else {
        // Push a new product object into the cart
        cart.push({ productId, productName, quantity, price });
    }
};

// 2. Remove Items from the Cart
const removeItemFromCart = (productId) => {
    // Find index of the product to remove by productId
    const productIndex = cart.findIndex(item => item.productId === productId);
    if (productIndex !== -1) {
        // Remove product using splice
        cart.splice(productIndex, 1);
    }
};

// 2. Update Item Quantity
const updateItemQuantity = (productId, newQuantity) => {
```

```
    cart = cart.map(item =>
        item.productId === productId ? { ...item, quantity: newQuantity } : item
    );
};

// 3. Calculate Total Cost
const calculateTotalCost = () => {
    // Use reduce to calculate total price based on quantity and price
    return cart.reduce((total, item) => total + (item.price * item.quantity), 0);
};

// 4. Display Cart Summary
const displayCartSummary = () => {
    // Use map to generate summary for each product
    const summary = cart.map(item => ({
        productName: item.productName,
        quantity: item.quantity,
        totalPrice: item.price * item.quantity
    }));

    console.log('Cart Summary:', summary);
    return summary;
};

// 4. Filter out Items with Zero Quantity
const filterZeroQuantityItems = () => {
    // Use filter to remove items with zero quantity
    cart = cart.filter(item => item.quantity > 0);
};

// 5. Apply Discount Code (Optional)
const applyDiscount = (discountCode) => {
    const discounts = {
        'DISCOUNT10': 0.10,  // 10% discount
        'DISCOUNT20': 0.20,  // 20% discount
    };

    // Check if discount code is valid
    const discount = discounts[discountCode] || 0;
    const totalCost = calculateTotalCost();
```

```
    // Apply discount to total cost
    return totalCost - (totalCost * discount);
};

// Test the shopping cart functionality

// Adding items to the cart
addItemToCart(1, 'Android Mobile', 5, 120000);
addItemToCart(2, 'Tablets', 4, 8000);
addItemToCart(3, 'EarPhones', 6, 1500);

// Display cart summary
displayCartSummary();

// Calculate total cost
console.log('Total Cost:', calculateTotalCost());

// Apply a discount
console.log('Total Cost after DISCOUNT10:', applyDiscount('DISCOUNT10'));

// Update quantity of an item
updateItemQuantity(2, 3);
console.log('Cart after updating quantity:');
displayCartSummary();

// Remove an item from the cart
removeItemFromCart(3);
console.log('Cart after removing Headphones:');
displayCartSummary();

// Filter out items with zero quantity (if any)
filterZeroQuantityItems();
console.log('Cart after filtering zero-quantity items:');
displayCartSummary();
```

Output Screenshot:

```
[Running] node "d:\mad\script.js"
Cart Summary: [
  { productName: 'Android Mobile', quantity: 5, totalPrice: 600000 },
  { productName: 'Tablets', quantity: 4, totalPrice: 32000 },
  { productName: 'EarPhones', quantity: 6, totalPrice: 9000 }
]
Total Cost: 641000
Total Cost after DISCOUNT10: 576900
Cart after updating quantity:
Cart Summary: [
  { productName: 'Android Mobile', quantity: 5, totalPrice: 600000 },
  { productName: 'Tablets', quantity: 3, totalPrice: 24000 },
  { productName: 'EarPhones', quantity: 6, totalPrice: 9000 }
]
Cart after removing Headphones:
Cart Summary: [
  { productName: 'Android Mobile', quantity: 5, totalPrice: 600000 },
  { productName: 'Tablets', quantity: 3, totalPrice: 24000 }
]
Cart after filtering zero-quantity items:
Cart Summary: [
  { productName: 'Android Mobile', quantity: 5, totalPrice: 600000 },
  { productName: 'Tablets', quantity: 3, totalPrice: 24000 }
]

[Done] exited with code=0 in 0.225 seconds
```

Conclusion:

In this code, I learned how to efficiently manage a shopping cart using modern JavaScript features like **arrow functions** and array methods such as map, filter, and reduce. The logic behind updating, removing, and calculating totals in a cart became clear, demonstrating how functional programming simplifies operations on data arrays.

Challenges faced:

- Ensuring efficient cart updates
- Discount Logic

  These challenges were solved using array methods and proper condition checks.