

Topic:

Mobile shopping cart

Objective of the Code

The primary objective of this code is to implement a JavaScript-based mobile shopping cart feature using modern ES6 functionalities. The feature aims to enable users to:

- - Add products to their shopping cart.
- - Remove items or update quantities.
- - Calculate the total cost of the cart.
- - Display a cart summary with details like product names, quantities, and total prices.
- - Optionally, apply discount codes for price reductions.

The code focuses on applying key ES6 concepts such as arrow functions, array methods (`map`, `filter`, `reduce`), and object manipulation techniques to handle common shopping cart operations efficiently.

Implemented Operations Code Explanation**1. Add Items to the Cart:**

- The `addItemToCart` function takes product details such as `productId`, `productName`, `quantity`, and `price` as inputs and pushes the product object into the `cart` array using the `push` method. This function represents the action of adding an item to the shopping cart.

Code

```
cart.push({ productId, productName, quantity, price });
```

2. Remove Items from the Cart:

- The `removeItemFromCart` function removes an item from the cart based on its `productId`. It first locates the product's index using `findIndex` and then removes it from the array using `splice`. This function helps in deleting unwanted products from the cart.

Code

```
const index = cart.findIndex(item => item.productId === productId);  
cart.splice(index, 1);  
...
```

3. Update Item Quantity:

- The `updateItemQuantity` function uses the `map` method to update the quantity of a specific product. It checks if the `productId` matches and creates a new cart array with the updated quantity while leaving other items unchanged.

Code

```
cart = cart.map(item =>
```

```

    item.productId === productId ? { ...item, quantity: newQuantity } : item
  );
  ...

```

4. Calculate Total Cost:

- The `calculateTotalCost` function uses the `reduce` method to compute the total price of all the products in the cart, taking into account the product's price and quantity. This is crucial for calculating the total cost before making a purchase.

Code

```

const total = cart.reduce((sum, item) => sum + item.price * item.quantity, 0);
...

```

5. Display Cart Summary:

- The `displayCartSummary` function iterates over the cart using `forEach` (or `map`) to display each product's name, quantity, and total price (price * quantity) for that product. This provides a clear summary of what's in the cart and how much each item costs.

Code

```

cart.forEach(item => {
  const totalPrice = item.price * item.quantity;
  console.log(` ${item.productName}: Quantity: ${item.quantity}, Total:
  $$ {totalPrice.toFixed(2)} `);
});

```

6. Filter Zero Quantity Items:

- The `filterZeroQuantityItems` function removes any items from the cart that have a quantity of zero, using the `filter` method. This ensures that no irrelevant or invalid items remain in the cart.

Code

```

cart = cart.filter(item => item.quantity > 0);

```

7. Apply Discount:

- The `applyDiscount` function applies a discount to the total price of the cart. The function checks if the discount code provided matches predefined discount rates and applies the appropriate percentage discount to the total cost.

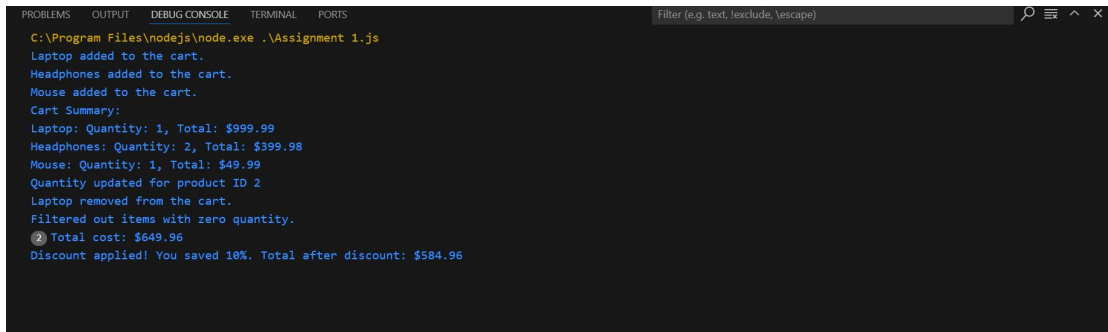
Code

```

const discount = discounts[discountCode];
const totalAfterDiscount = totalBeforeDiscount * (1 - discount);

```

Output:



```
C:\Program Files\nodejs\node.exe .\Assignment 1.js
Laptop added to the cart.
Headphones added to the cart.
Mouse added to the cart.
Cart Summary:
Laptop: Quantity: 1, Total: $999.99
Headphones: Quantity: 2, Total: $399.98
Mouse: Quantity: 1, Total: $49.99
Quantity updated for product ID 2
Laptop removed from the cart.
Filtered out items with zero quantity.
Total cost: $649.96
Discount applied! You saved 10%. Total after discount: $584.96
```

Conclusion

In conclusion, this code demonstrates the power and flexibility of ES6 arrow functions and array methods (`map`, `filter`, `reduce`, `findIndex`) to manage the shopping cart effectively. Key concepts like object manipulation and array transformations are applied to carry out common e-commerce functionalities such as adding, removing, and updating products, as well as calculating the total price and applying discounts.

We learned how to:

- - Structure the shopping cart as an array of objects.
- - Use array methods like `map`, `filter`, and `reduce` to manipulate the cart efficiently.
- - Handle user interactions such as adding, removing, or updating items.
- - Apply modern JavaScript techniques (like arrow functions and the spread operator) for concise and readable code.

The code serves as a solid foundation for building dynamic and responsive shopping cart systems in mobile or web applications, while promoting clean, maintainable, and modern JavaScript practices.