COMSATS UNIVERSITY ISLAMABAD

ATTOCK CAMPUS

DEPARTMENT OF COMPUTER SCIENCES

SUBMITTED TO:

Sir Muhammad Kamran

SUBMITTED BY:

Malaika

(SP22-BSE-003)

COURSE NAME:

Mobile Application Development.

DATED:

26-SEP-2024

Assignment:

01.

1.Introduction

The objective of this assignment is to develop a JavaScript-based mobile shopping cart feature using modern ES6 features such as arrow functions, array methods (map, filter, reduce), and object manipulation techniques. The implementation simulates core shopping cart functionalities used in e-commerce platforms, providing users the ability to:

1. **Add items to the cart**: Users can add products, each represented by a unique product ID, name, quantity, and price.
2. **Remove items from the cart**: Products can be removed from the cart based on their ID.
3. **Update item quantities**: Users can modify the quantity of any product in the cart.
4. **Calculate total price**: The system calculates the total cost of all products in the cart, considering the quantity and price of each item.
5. **Display cart summary**: A detailed cart summary is generated, displaying each product's name, quantity, and total price.
6. **Filter items with zero quantity**: Items with zero quantity are automatically filtered out of the cart to keep it organized.
7. **Apply discount codes**: Users can apply discount codes (e.g., "DISCOUNT10") to get a percentage reduction in the total price.

2.Code Explanation

Each function in the shopping cart code plays a specific role in managing the cart operations. Here is a breakdown of the logic behind each function:

1. addItem Function

Purpose:
This function adds a product to the shopping cart. It takes four parameters: productId, productName, quantity, and price, and pushes a product object into the cart array.

Logic:

- The function uses the push method to add a new product object to the cart array.
- Each object contains four properties: productId, productName, quantity, and price.

```
const addItem = (productId, productName, quantity, price) => {
    cart.push({ productId, productName, quantity, price });
    console.log(`${productName} added to the cart.`);
};
```

2. removeItem Function

Purpose:
This function removes an item from the cart based on its productId.

Logic:

- It uses the filter method to create a new cart array, excluding the item whose productId matches the one provided.
- The cart is then updated with the filtered array, effectively removing the item.

```
const removeItem = (productId) => {
    cart = cart.filter(item => item.productId !== productId);
    console.log(`Item with ID ${productId} removed from the cart.`);
};
```

---

2. removeItem Function

Purpose:
This function removes an item from the cart based on its productId.

Logic:

- It uses the filter method to create a new cart array, excluding the item whose productId matches the one provided.
- The cart is then updated with the filtered array, effectively removing the item.

javascript

Copy code

```
const removeItem = (productId) => {
    cart = cart.filter(item => item.productId !== productId);
    console.log(`Item with ID ${productId} removed from the cart.`);
};
```

3. updateQuantity Function

Purpose:
This function updates the quantity of a specific item in the cart.

Logic:

- The function uses the map method to iterate over the cart array.

- If the productId matches the provided ID, the quantity of that product is updated with the new value.
- The rest of the items remain unchanged.

```
const updateQuantity = (productId, quantity) => {
    cart = cart.map(item => item.productId === productId ? { ...item, quantity } : item);
    console.log(`Quantity updated for product ID: ${productId}`);
};
```

4. calculateTotal Function

Purpose:
This function calculates the total price of all items in the cart by considering the price and quantity of each product.

Logic:

- The function uses the reduce method to iterate over the cart array.
- For each item, it multiplies the price by quantity and sums up the results to get the total price.

```
const calculateTotal = () => {
    return cart.reduce((total, item) => total + (item.price * item.quantity), 0);
};
```

5. displayCartSummary Function

Purpose:
This function displays a summary of all the items in the cart, including the product name, quantity, and total price for each item.

Logic:

- The function uses the map method to iterate over the cart array.
- For each item, it calculates the total price by multiplying the price by the quantity.
- The name, quantity, and total price are then printed to the console.

```
const displayCartSummary = () => {
    console.log("Cart Summary:");
    cart.map(item => {
        const totalPrice = item.price * item.quantity;
        console.log(`Product: ${item.productName}, Quantity: ${item.quantity}, Total Price
    });
};
```

6. filterZeroQuantity Function

Purpose:
This function removes items from the cart that have a quantity of zero.

Logic:

- The function uses the filter method to iterate over the cart array.
- Only items with a quantity greater than zero are retained in the cart, removing those with a quantity of zero.

```
const filterZeroQuantity = () => {
    cart = cart.filter(item => item.quantity > 0);
    console.log("Items with zero quantity have been filtered out.");
};
```

7. applyDiscount Function

Purpose:
This function applies a discount to the total price based on a discount code.

Logic:

- The function checks if the provided discountCode is equal to "DISCOUNT10". If so, it applies a 10% discount.
- The calculateTotal function is called to get the current total price, and the discount is subtracted from this total.
- The discounted total is printed to the console.

```
const applyDiscount = (discountCode) => {
    const discount = discountCode === "DISCOUNT10" ? 0.1 : 0;
    const total = calculateTotal();
    const finalPrice = total - (total * discount);
    console.log(`Total after applying discount: $${finalPrice}`);
};
```

3.Output SS:

```
[Running] node "d:\Malaika 6 sem\MAD\assignment.js"
Laptop added to the cart.
Mouse added to the cart.
Keyboard added to the cart.

Initial Cart:
Cart Summary:
Product: Laptop, Quantity: 1, Total Price: $1000
Product: Mouse, Quantity: 2, Total Price: $40
Product: Keyboard, Quantity: 0, Total Price: $0
Total Price: $1040

Quantity updated for product ID: 2
Items with zero quantity have been filtered out.

Updated Cart After Filtering and Updating Quantity:
Cart Summary:
Product: Laptop, Quantity: 1, Total Price: $1000
Product: Mouse, Quantity: 3, Total Price: $60
Updated Total Price: $1060

Total after applying discount: $954

[Done] exited with code=0 in 0.358 seconds
```

4.Conclusion

Through this assignment, I gained valuable insights into the practical applications of JavaScript, particularly in building interactive web features like a shopping cart. Here are some key reflections on what I learned and the challenges I faced during the development process:

Learning Outcomes:

1. **Understanding JavaScript Concepts:**
    - I deepened my understanding of core JavaScript concepts, especially ES6 features such as arrow functions and array methods (map, filter, reduce).
2. **Object Manipulation:**
    - I learned how to effectively use objects to represent complex data structures.

3. **Function Composition:**
    - The importance of modular programming was highlighted as I broke down the cart functionalities into distinct functions
4. **User Experience Considerations:**
    - I gained insights into how essential it is to provide clear and user-friendly messages in the console
5. **Debugging and Problem-Solving:**
    - Throughout the implementation process, I encountered bugs and logical errors that required critical thinking and debugging skills.

Challenges Faced:

1. **Managing State:**
    - One of the primary challenges was managing the state of the cart, particularly when removing items or updating quantities.
2. **Implementing Discounts:**
    - Implementing the discount feature posed challenges in determining how to effectively calculate and apply discounts while maintaining the total price accurately.
3. **Error Handling:**
    - I initially struggled with how to handle cases where users might try to remove items that are not in the cart or update quantities to invalid values (e.g., negative numbers).
4. **Ensuring Code Readability:**
    - Balancing concise code with readability was a constant consideration

Overall Reflection:

This assignment has significantly enhanced my JavaScript skills and provided practical experience in developing a key feature commonly used in e-commerce applications. It has also underscored the importance of good coding practices, effective state management, and user-centric design. I look forward to applying these skills in future projects and continuing to explore more complex programming concepts.