COMSATS UNIVERSITY ISLAMABAD

Name: Malik Arfat Hassan

Reg No: SP22-BSE-024

Assignment No: 01

Date: 09/26/24

1. Add Items to the Cart

Function: addItemToCart(productId, productName, quantity, price)

- **Purpose**: This function is used to add items to the shopping cart. It accepts four parameters:
    - productId: A unique identifier for each product.
    - productName: The name of the product.
    - quantity: The number of units of the product.
    - price: The price per unit of the product.

- **Logic**:
    - A new object representing the product is created, containing the product details (productId, productName, quantity, price).
    - The product is then added to the cart array using cart.push(newProduct).

Code:

```
const addItemToCart = (productId, productName, quantity, price) => {
  const newProduct = { productId, productName, quantity, price };
  cart.push(newProduct);
};
```

---

2. Remove and Update Items

a. Function: removeItemFromCart(productId)

- **Purpose**: This function removes an item from the cart based on the product's unique productId.
- **Logic**:
    - The function finds the index of the product using findIndex() by matching the productId.
    - If the product is found (index is not -1), it removes the product from the cart using cart.splice().

Code:

```
const removeItemFromCart = (productId) => {
  const index = cart.findIndex((product) => product.productId === productId);
  if (index !== -1) {
    cart.splice(index, 1);
  }
};
```

b. Function: updateItemQuantity(productId, newQuantity)

- **Purpose**: This function allows updating the quantity of a specific product in the cart.
- **Logic**:
  - The function searches for the product in the cart using find() by matching the productId.
  - If the product is found, it updates the quantity property with the new quantity (newQuantity).

Code:

```
const updateItemQuantity = (productId, newQuantity) => {
  const product = cart.find((product) => product.productId === productId);
  if (product) {
    product.quantity = newQuantity;
  }
};
```

---

3. Calculate Total Cost

Function: calculateTotalCost()

- **Purpose**: This function calculates the total cost of all items in the cart, taking into account the quantity and price of each product.
- **Logic**:
  - It uses the reduce() method to sum up the total cost by multiplying each product's price by its quantity.
  - The total cost is returned as a number.

Code:

```
const calculateTotalCost = () => {
  return cart.reduce((acc, product) => acc + (product.price * product.quantity), 0);
};
```

---

4. Display Cart Summary

Function: generateCartSummary()

- **Purpose**: This function generates a summary of the cart, displaying each product's name, quantity, and the total price for that item (calculated as price * quantity).
- **Logic**:
  - It first filters out products with a quantity of zero or less using filter().
  - Then it uses map() to transform each product into a new object containing the productName, quantity, and totalPrice.
  - The function returns this summary as an array of objects.

Code:

```
const generateCartSummary = () => {
  const cartSummary = cart
    .filter((product) => product.quantity > 0)
    .map((product) => ({
      productName: product.productName,
      quantity: product.quantity,
      totalPrice: product.price * product.quantity,
    }));
  return cartSummary;
};
```

---

5. Bonus: Apply Discount Code

a. Function: applyDiscountCode(discountCode)

- **Purpose**: This function calculates the total cost of the cart after applying a discount code. It reduces the total cost based on the value of the discount.
- **Logic**:
  - It calls getDiscountAmount() to retrieve the discount amount based on the provided discount code.
  - It subtracts the discount amount from the total cost of the cart (calculated by calculateTotalCost()).

Code:

```
const applyDiscountCode = (discountCode) => {
  const discountAmount = getDiscountAmount(discountCode);
  return calculateTotalCost() - discountAmount;
};
```

b. Function: getDiscountAmount(discountCode)

- **Purpose**: This function returns the discount amount based on the discount code entered by the user.
- **Logic**:
  - For demonstration purposes, this function assumes that a discount code "SUMMER10" applies a 10% discount on the total cart value.
  - If the discount code matches "SUMMER10", it returns 10% of the total cart value as the discount.

Code:

```
const getDiscountAmount = (discountCode) => {
  if (discountCode === "SUMMER10") {
    return calculateTotalCost() * 0.1;
  }
  return 0;
};
```

Example Usage and Output

1. **Adding Items to the Cart**:
    - Three products are added to the cart: EarPods, C-type Charger, and Mobile Phone.

Code:

```
addItemToCart(1, "EarPods", 2, 10.99);

addItemToCart(2, "C-type Charger", 1, 5.99);

addItemToCart(3, "Mobile Phone", 3, 7.99);
```

2. **Displaying Cart Summary**:
    - After adding the items, the cart summary displays the products along with their quantities and total prices.

Code:

```
console.log(generateCartSummary());
```

3. **Removing an Item**:
    - The C-type Charger is removed using the removeItemFromCart() function.

Code:

```
removeItemFromCart(2);
```

4. **Updating Item Quantity**:
    - The quantity of EarPods is updated from 2 to 3.

Code:

```
updateItemQuantity(1, 3);
```

5. **Calculating Total Cost**:
    - The total cost of all the products in the cart is calculated.

Code:

```
console.log(calculateTotalCost());
```

6. **Applying a Discount Code**:
    - The discount code "SUMMER10" is applied, reducing the total cost by 10%.

Code:

```
console.log(applyDiscountCode("SUMMER10"));
```

Conclusion

This shopping cart implementation covers essential functionalities:

- **Adding, removing, and updating** items in the cart.
- **Calculating the total cost** of the cart, with or without discounts.
- **Displaying a cart summary**, which can easily be extended for a real user interface.

The code is flexible and can easily be adapted for different applications, such as e-commerce websites or retail apps, where managing products and applying discounts are core features.