# NexusBerry Training and Solutions

# GRAND JS TEST

In this project, you are required to write functions that are integral part of any E-Commerce application.

## Problem Statement 1 (addToCart.js):

Suppose we have following products in our online store:

```
let products = [
  {id: 1, price: 100, discount: 0.10},
  {id: 2, price: 200, discount: 0.10},
  {id: 3, price: 300, discount: 0.10},
  {id: 4, price: 400, discount: 0.10},
  {id: 5, price: 500, discount: 0.05},
];
```

Initially the cart is empty:

```
let cart = [];
```

You are required to write a function named ‘addToCart’:

```
function addToCart(cart, product) {
  // write your logic here
}
```

## Explanation:

- The ‘addToCart’ function should take the cart array and a product object as parameters and should return the new cart array.

- When the addToCart function will be called by passing a product object and cart, there are two case that need to be handled in this function. Case-I is when a product is being added first time in the cart and Case-II is when the product already exists in the cart.
- Case-I: If the product object does not exist already in cart array, then it should be added to the cart along with two new properties in the object, i.e., quantity and total. The default value of quantity for new product is 1. The total should be calculated based on the price and discount of the product. Please note that the discount of each product is already stored in each object as shown in the products array e.g., for the first four products the discount is 0.10 (10%) and for the last product it is 0.05 (5%).
- Case-II: If the product already exists in the cart, then it should not be added in the cart. Only its quantity will be increment by 1 and its total should also be recalculated ad its quantity has been changed.
- React State Immutability Constraint: The ‘addToCart’, function should never modify the cart array or product object directly that are passed them as parameters. They should be treated as immutable.

**Try-Run:**

```
let products = [
  {id: 1, price: 100, discount: 0.10},
  {id: 2, price: 200, discount: 0.10},
  {id: 3, price: 300, discount: 0.10},
  {id: 4, price: 400, discount: 0.10},
  {id: 5, price: 500, discount: 0.05},
];

let cart = [];
// add or updadte qty of the given product into the cart and return ne
w cart
function addToCart(cart, product) {
 // write your logic here
}

console.log(cart);
// output: []

// add to cart
let product = products[2];
cart = addToCart(cart, product);
console.log(cart);
//output:  [ { id: 3, price: 300, discount: 0.1, qty: 1, total: 270 }
]

// add to cart
product = products[4];
cart = addToCart(cart, product);
console.log(cart);
//output: [
//   { id: 3, price: 300, discount: 0.1, qty: 1, total: 270 },
//   { id: 5, price: 500, discount: 0.05, qty: 1, total: 475 }
// ]

// add to cart
product = products[2];
cart = addToCart(cart, product);
console.log(cart);
//output: [
//   { id: 3, price: 300, discount: 0.1, qty: 2, total: 540 },
//   { id: 5, price: 500, discount: 0.05, qty: 1, total: 475 }
// ]

// add to cart
product = products[0];
cart = addToCart(cart, product);
console.log(cart);
//output: [
//   { id: 3, price: 300, discount: 0.1, qty: 2, total: 540 },
//   { id: 5, price: 500, discount: 0.05, qty: 1, total: 475 },
//   { id: 1, price: 100, discount: 0.1, qty: 1, total: 90 }
// ]

// add to cart
product = products[4];
cart = addToCart(cart, product);
console.log(cart);
```

```
//output: [
//   { id: 3, price: 300, discount: 0.1, qty: 2, total: 540 },
//   { id: 5, price: 500, discount: 0.05, qty: 2, total: 950 },
//   { id: 1, price: 100, discount: 0.1, qty: 1, total: 90 }
// ]
```

## How to run Program:

- Copy paste all the code in the try run in your editor in a cart.js file.
- Write your logic in the addToCat function body and then run the program writing “node cart.js”.

## Problem Statement 2 (removeFromCart.js):

Consider the following function

```
function removeFromCart(cart, product) {
  // write your logic here
}
```

- This function should take the cart array and a product object that is to be removed from the cart as parameters and should return the new cart array.
- React State Immutability Constraint: This function should never modify the cart array or product object directly that are passed them as parameters. They should be treated as immutable.

**Try-Run:**

Suppose we have the following array

```
let cart = [
    { id: 3, price: 300, discount: 0.1, qty: 2, total: 540 },
    { id: 5, price: 500, discount: 0.05, qty: 2, total: 950 },
    { id: 1, price: 100, discount: 0.1, qty: 1, total: 90 }
  ]
  let product = cart[1];
  cart = removeFromCart(cart, product);
  console.log(cart);
//output: [
//   { id: 3, price: 300, discount: 0.1, qty: 2, total: 540 },
//   { id: 5, price: 500, discount: 0.05, qty: 1, total: 950 },
//   { id: 1, price: 100, discount: 0.1, qty: 1, total: 90 }
// ]
```

## Problem Statement 3 (incrementCart.js):

Consider the following function

```
function incrementCart(cart, product) {
  // write your logic here
}
```

- This function should take the cart array and a product object whose quantity is to be incremented from the cart as parameters and should return the new cart array.
- React State Immutability Constraint: This function should never modify the cart array or product object directly that are passed them as parameters. They should be treated as immutable.

**Try-Run:**

Suppose we have the following array

```
   let cart = [
      { id: 3, price: 300, discount: 0.1, qty: 2, total: 540 },
      { id: 5, price: 500, discount: 0.05, qty: 2, total: 950 },
      { id: 1, price: 100, discount: 0.1, qty: 1, total: 90 }
   ]
   let product = cart[1];
   cart = incrementCart(cart, product);
   console.log(cart);
//output: [
//   { id: 3, price: 300, discount: 0.1, qty: 2, total: 540 },
//   { id: 5, price: 500, discount: 0.05, qty: 3, total: 950 },
//   { id: 1, price: 100, discount: 0.1, qty: 1, total: 90 }
// ]
```

## **Problem Statement 4 (decrementCart.js):**

Consider the following function

```
function decrementCart(cart, product) {
  // write your logic here
}
```

- This function should take the cart array and a product object whose quantity is to be decremented from the cart as parameters and should return the new cart array.
- If the quantity becomes ZERO after decrement, then this object should be removed from the cart array.

- React State Immutability Constraint: This function should never modify the cart array or product object directly that are passed them as parameters. They should be treated as immutable.

**Try-Run:**

Suppose we have the following array

```
let cart = [
   { id: 3, price: 300, discount: 0.1, qty: 2, total: 540 },
   { id: 5, price: 500, discount: 0.05, qty: 2, total: 950 },
   { id: 1, price: 100, discount: 0.1, qty: 1, total: 90 }
]
let product = cart[0];
cart = decrementCart(cart, product);
console.log(cart);
```

```
//output: [
//   { id: 3, price: 300, discount: 0.1, qty: 1, total: 540 },
//   { id: 5, price: 500, discount: 0.05, qty: 2, total: 950 },
//   { id: 1, price: 100, discount: 0.1, qty: 1, total: 90 }
// ]
let product = cart[0];
cart = decrementCart(cart, product);
console.log(cart);
//output: [
//   { id: 5, price: 500, discount: 0.05, qty: 2, total: 950 },
//   { id: 1, price: 100, discount: 0.1, qty: 1, total: 90 }
// ]
```

## Problem Statement 5

Suppose we have an array of products and we want to add a property at run time in each of objects whose price is greater than 1000. Write a function named **addDiscountProperty** that should do this task. It should be a pure function and it should not modify its input arguments rather it should return the desired array.

```
let products = [
 {
     id: 1,
     name: ‘Shirt-1’,
     sizes: [‘sm’, ‘md’, ‘lg’, ‘xl’],
     price: 800
},
{
     id: 2,
     name: ‘Shirt-2’,
     sizes: [‘sm’, ‘md’, ‘lg’, ‘xl’],
     price: 1200
}
. . .
];
let prods = addDiscountProperty(products, 0.10);
console.log(prods);
 // output:
```

```
[
 {
     id: 1,
     name: ‘Shirt-1’,
     sizes: [‘sm’, ‘md’, ‘lg’, ‘xl’],
     price: 800
},
{
     id: 2,
     name: ‘Shirt-2’,
     sizes: [‘sm’, ‘md’, ‘lg’, ‘xl’],
     price: 1200,
     discount: 0.10
}
. . .
]
```

```
prods = addDiscountProperty(products, 0.05);
console.log(prods);
 // output:
```

```
[
 {
     id: 1,
     name: ‘Shirt-1’,
     sizes: [‘sm’, ‘md’, ‘lg’, ‘xl’],
     price: 800
},
{
      id: 2,
      name: ‘Shirt-2’,
      sizes: [‘sm’, ‘md’, ‘lg’, ‘xl’],
      price: 1200,
      discount: 0.05
}
. . .
]
```

## Problem Statement 6

Suppose we have an array of products, and we want to filter all those products that contains any of the size given in the sizes array. Write a function named **filterProductsBySize** that should do this task. The filterProductsBySize should be a pure function and it should not modify its input arguments. It should return an array that contains the filters products. Note that if sizes array (that is passed as parameter) is empty then the function should not filter and return all products.

```
function filterProductsBySize(products, sizes) {
      // write your logic here
}

let products = [
 {
      id: 1,
      name: ‘Shirt-1’,
      sizes: [‘sm’, ‘md’, ‘lg’, ‘xl’]
},
{
      id: 2,
      name: ‘Shirt-2’,
      sizes: [‘lg’, ‘xl’]
},
{
      id: 3,
      name: ‘Shirt-3’,
      sizes: [‘sm’, ‘md’],
},
{
      id: 4,
      name: ‘Shirt-4’,
      sizes: [‘md’, ‘lg’, ‘xl’]
},
];

let sizes = [‘sm’];
let prods = filterProductsBySize(products, sizes);
console.log(prods);
// output:
```

```
[
 {
      id: 1,
      name: ‘Shirt-1’,
      sizes: [‘sm’, ‘md’, ‘lg’, ‘xl’]
},
{
      id: 3,
      name: ‘Shirt-3’,
      sizes: [‘sm’, ‘md’],
}
]
```

```
let sizes = [‘md’, ‘lg’];
let prods = filterProductsBySize(products, sizes);
console.log(prods);
// output:
```

```
[
 {
      id: 1,
      name: ‘Shirt-1’,
      sizes: [‘sm’, ‘md’, ‘lg’, ‘xl’]
},
{
      id: 2,
      name: ‘Shirt-2’,
        sizes: [‘lg’, ‘xl’]
},
{
        id: 3,
        name: ‘Shirt-3’,
        sizes: [‘sm’, ‘md’],
},
{
        id: 4,
        name: ‘Shirt-4’,
        sizes: [‘md’, ‘lg’, ‘xl’]
},
]

let sizes1 = [‘sm’, ‘xl’];
let prods = filterProductsBySize(products, sizes);
console.log(prods);
```

```
let sizes = [‘sm’, ‘lg’];
let prods = filterProductsBySize(products, sizes);
console.log(prods);
// output:
```

```
[
 {
        id: 1,
        name: ‘Shirt-1’,
     sizes: [‘sm’, ‘md’, ‘lg’, ‘xl’]
},
{
     id: 2,
     name: ‘Shirt-2’,
     sizes: [‘lg’, ‘xl’]
},
{
     id: 4,
     name: ‘Shirt-4’,
     sizes: [‘md’, ‘lg’, ‘xl’]
},
]
```

```
let sizes = [];
let prods = filterProductsBySize(products, sizes);
console.log(prods);
// output:
```

```
Prints Original Products Array
```