Day 6: Arrays & Array Methods

صفیں اور طریقے

> **Quote of the Day:** *"Arrays are the backbone of data manipulation. Master them, and you master half of programming."* - Kyle Simpson
>
> *"استعمال کرو۔ Arrays ایک سے زیادہ چیزوں کو سنبھالنا ہو تو"*

📋 Today's Learning Goals (آج کے اہداف)

By the end of today, you will:

- ☐ Create and manipulate arrays confidently
- ☐ Master array methods: push, pop, shift, unshift
- ☐ Use modern array methods: forEach, map, filter
- ☐ Understand zero-based indexing (0, 1, 2...)
- ☐ Build a Daraz Product Manager inventory system

Time Breakdown (کل وقت: 150 منٹ)

- 🕐 7:00-7:05 PM (5min): Standup - Show your Utility Functions!
- 🕐 7:05-8:05 PM (60min): Understanding arrays (3× Pomodoro)
- 🕐 8:05-8:50 PM (45min): Practice with array methods
- 🕐 8:50-9:25 PM (35min): Build Product Manager
- 🕐 9:25-9:30 PM (5min): Quiz & reflection

🎯 What We're Building Today

Today you'll create a **Daraz Product Manager** - an inventory system that stores products as array of objects, adds/removes items, filters out-of-stock products, calculates total value, and manages categories!

Why This Matters for Your Career:
EVERY app uses arrays constantly:

- **Instagram:** Array of posts for your feed
- **Daraz:** Array of products in search results
- **WhatsApp:** Array of messages in each chat
- **YouTube:** Array of videos in your recommendations

Arrays are how we handle lists of data!

🧠 سمجھنا (Understanding): What Are Arrays?

The Real-World Analogy

Scenario: Aloo Paratha Layers (آلو پراٹھے کی تہیں)

Imagine you're making aloo paratha:

```
Layer 1 (top):    Ghee
Layer 2:          Dough
Layer 3:          Potato filling
Layer 4:          Dough
Layer 5 (bottom): Ghee
```

This is an array!

- Each layer has a **position** (1st, 2nd, 3rd...)

- All layers are **stacked together**
- You can **add** more layers (push)
- You can **remove** layers (pop)
- Each layer can be **different** (ghee, dough, filling)

In JavaScript:

```
const paratha = ["Ghee", "Dough", "Potato", "Dough", "Ghee"];
```

Daily Life Examples

1. **Student Roll Numbers in Class:**

```
Class List:
Position 0: Ali (Roll 101)
Position 1: Sara (Roll 102)
Position 2: Hassan (Roll 103)
Position 3: Fatima (Roll 104)
```

2. **Cricket Batting Order:**

```
Position 0: Babar Azam
Position 1: Fakhar Zaman
Position 2: Mohammad Rizwan
... (all the way to position 10)
```

3. **Shopping Cart on Daraz:**

```
Item 0: Laptop - Rs. 75,000
Item 1: Mouse - Rs. 1,200
Item 2: Keyboard - Rs. 3,500
```

Why Does This Matter?

Without arrays:

```
const student1 = "Ali";
const student2 = "Sara";
const student3 = "Hassan";
const student4 = "Fatima";
// ... for 100 students? 100 variables! 😱
```

With arrays:

```
const students = ["Ali", "Sara", "Hassan", "Fatima"];
// Add 100 more? No problem!
// One variable, unlimited data!
```

The Mental Model

Think of an array like a **numbered row of dabbas** (ڈبوں کی قطار):

```
Index:    [0]      [1]       [2]      [3]
Value:   "Ali"   "Sara"  "Hassan" "Fatima"
           ↑        ↑         ↑        ↑
      Dabba 0  Dabba 1   Dabba 2  Dabba 3
```

Key point: Arrays start counting at **ZERO** (not 1)!

📚 Building Block #1: Creating & Accessing Arrays

What is an Array? (کیا ہے؟)

Urdu Analogy: Think of an array like a **tasbih** (تسبیح) - prayer beads on a string.

```
Bead 0: ●  (Bead at position 0)
Bead 1: ●  (Bead at position 1)
Bead 2: ●  (Bead at position 2)
... all on one string!
```

Each bead has a **position**, and they're all connected together!

How to Create Arrays

```
// THINKING: Different ways to create arrays

// Method 1: Array literal (most common)
const cities = ["Lahore", "Karachi", "Islamabad"];

// Method 2: Empty array (add items later)
const scores = [];

// Method 3: Array with different types
const mixed = ["Ali", 20, true, "Lahore"];

// Method 4: Array constructor (less common)
const numbers = new Array(1, 2, 3);
```

Zero-Based Indexing (0 سے شروع)

CRITICAL: Arrays start counting at **ZERO**, not one!

```
const cities = ["Lahore", "Karachi", "Islamabad", "Peshawar"];

// Index:      0         1           2            3
// Position: 1st       2nd         3rd          4th

console.log(cities[0]);  // "Lahore" (first item!)
console.log(cities[1]);  // "Karachi" (second item)
console.log(cities[2]);  // "Islamabad" (third item)
console.log(cities[3]);  // "Peshawar" (fourth item)
```

Why zero-based? It's a computer science convention. Just memorize: **first item = index 0**!

Accessing Array Elements

```
// THINKING: Getting items from array

const fruits = ["Aam", "Kela", "Santra", "Angoor"];

// Get first item (index 0)
console.log(fruits[0]);  // "Aam"

// Get last item (length - 1)
console.log(fruits[3]);  // "Angoor"
console.log(fruits[fruits.length - 1]);  // "Angoor" (works for any
length!)

// Get item in middle
console.log(fruits[1]);  // "Kela"

// Try to get item that doesn't exist
console.log(fruits[10]);  // undefined (no error, just undefined)
```

Array Length Property

```
const students = ["Ali", "Sara", "Hassan"];

console.log(students.length);  // 3 (total number of items)

// Last item is always at: length - 1
console.log(students[students.length - 1]);  // "Hassan"
```

Your First Example

```
// TODO: Create array of Pakistani cities
const cities = ["Lahore", _______, _______, _______];

// TODO: Access first city
console.log("First city:", cities[_____]);

// TODO: Access last city
console.log("Last city:", cities[cities._____ - 1]);

// TODO: How many cities?
console.log("Total cities:", cities._____);

// TODO: Access city at index 2
console.log("City at index 2:", cities[_____]);
```

Common Mistakes

❌ **Wrong:**

```
const arr = ["A", "B", "C"];
console.log(arr[1]);  // "A" ?
```

Why wrong? First item is index 0, not 1!

✅ **Right:**

```
const arr = ["A", "B", "C"];
console.log(arr[0]);  // "A" ✅
console.log(arr[1]);  // "B" ✅
```

❌ **Wrong:**

```
const arr = ["A", "B", "C"];
console.log(arr[arr.length]);  // undefined (out of bounds!)
```

✅ **Right:**

```
const arr = ["A", "B", "C"];
console.log(arr[arr.length - 1]);  // "C" (last item)
```

Check Your Understanding

- ☐ What index is the first item?
- ☐ How do you get the last item?
- ☐ What does array.length return?
- ☐ What happens if you access index 100 in a 3-item array?

📚 Building Block #2: Modifying Arrays (push, pop, shift, unshift)

What are Array Methods? (کیا ہیں؟)

Urdu Analogy: Think of a **queue at National Bank** (قطار):

```
push()    = Join queue at END (آخر میں آنا)
pop()     = Leave from END (آخر سے جانا)
unshift() = Cut the line at START (شروع میں آنا)
shift()   = Leave from START (شروع سے جانا)
```

1. push() - Add to End

```
// THINKING: Adding items to array

const fruits = ["Aam", "Kela"];
console.log(fruits);  // ["Aam", "Kela"]

fruits.push("Santra");
console.log(fruits);  // ["Aam", "Kela", "Santra"]

fruits.push("Angoor");
console.log(fruits);  // ["Aam", "Kela", "Santra", "Angoor"]

// Can add multiple at once!
fruits.push("Anar", "Seb");
console.log(fruits);  // ["Aam", "Kela", "Santra", "Angoor", "Anar", "Seb"]
```

2. pop() - Remove from End

```
// THINKING: Removing items from end

const cities = ["Lahore", "Karachi", "Islamabad"];
console.log(cities);  // ["Lahore", "Karachi", "Islamabad"]

const removed = cities.pop();
console.log(removed);  // "Islamabad" (returns removed item!)
console.log(cities);   // ["Lahore", "Karachi"]

cities.pop();
console.log(cities);   // ["Lahore"]
```

3. unshift() - Add to Start

```
// THINKING: Adding to front (cutting the line!)

const numbers = [2, 3, 4];
console.log(numbers);  // [2, 3, 4]

numbers.unshift(1);
console.log(numbers);  // [1, 2, 3, 4]

numbers.unshift(0);
console.log(numbers);  // [0, 1, 2, 3, 4]
```

4. shift() - Remove from Start

```
// THINKING: Removing from front

const queue = ["Ali", "Sara", "Hassan"];
console.log(queue);  // ["Ali", "Sara", "Hassan"]

const served = queue.shift();
console.log(served);  // "Ali" (first person served!)
console.log(queue);   // ["Sara", "Hassan"]
```

Visual Summary

```
Original: ["A", "B", "C"]

push("D"):    ["A", "B", "C", "D"]    // Add to end
pop():        ["A", "B"]              // Remove from end
unshift("Z"): ["Z", "A", "B", "C"]   // Add to start
shift():      ["B", "C"]              // Remove from start
```

Your First Example

```
// TODO: Practice array methods

const cart = ["Laptop"];

// TODO: Add "Mouse" to end
cart._____(______);

// TODO: Add "Keyboard" to end
cart._____(______);

console.log(cart);  // ["Laptop", "Mouse", "Keyboard"]

// TODO: Remove last item
cart._____();

console.log(cart);  // ["Laptop", "Mouse"]

// TODO: Add "Headphones" to start
cart._____(______);

console.log(cart);  // ["Headphones", "Laptop", "Mouse"]
```

Common Mistakes

❌ **Wrong:**

```
const arr = [1, 2, 3];
arr.push = 4;  // Not how push works!
```

✅ **Right:**

```
const arr = [1, 2, 3];
arr.push(4);  // Use parentheses to call method
```

❌ **Wrong:**

```
const arr = [1, 2, 3];
const removed = arr.pop;   // Missing ()
```

✅ **Right:**

```
const arr = [1, 2, 3];
const removed = arr.pop();   // Call the method!
```

Check Your Understanding

- ☐ Which method adds to the end?
- ☐ Which method removes from the start?
- ☐ What does pop() return?
- ☐ Can you add multiple items with push()?

📚 Building Block #3: Looping Through Arrays

Why Loop Arrays? (کیوں؟)

Urdu Analogy: Think of **checking each student's attendance** (حاضری):

```
For each student in class:
    Call their name
    Mark present/absent
```

Same with arrays - do something for EACH item!

Method 1: for Loop (Traditional)

```
// THINKING: Loop with index

const cities = ["Lahore", "Karachi", "Islamabad"];

for (let i = 0; i < cities.length; i++) {
    console.log("City " + i + ":", cities[i]);
}

// Output:
// City 0: Lahore
// City 1: Karachi
// City 2: Islamabad
```

Method 2: forEach() (Modern & Clean)

```
// THINKING: forEach automatically loops

const cities = ["Lahore", "Karachi", "Islamabad"];

cities.forEach(function(city) {
    console.log("City:", city);
});

// Even cleaner with arrow function:
cities.forEach(city => {
    console.log("City:", city);
});

// Output:
// City: Lahore
// City: Karachi
// City: Islamabad
```

forEach with Index

```
// THINKING: Sometimes you need the index too

const students = ["Ali", "Sara", "Hassan"];

students.forEach((student, index) => {
    console.log(`Student ${index + 1}: ${student}`);
});

// Output:
// Student 1: Ali
// Student 2: Sara
// Student 3: Hassan
```

Your First Example

```
const prices = [500, 1200, 350, 890];

// TODO: Loop and print each price
// HINT: Use forEach

prices.forEach(price => {
    console.log("Rs. " + _____);
});

// TODO: Calculate total using forEach
let total = 0;

prices.forEach(price => {
    total _____ price;
});

console.log("Total:", total);
```

Check Your Understanding

- ☐ What's the difference between for and forEach?
- ☐ Can forEach access the index?

- ☐ Which is more modern?

📚 Building Block #4: Array Methods - map() and filter()

What is map()? (کیا ہے؟)

Urdu Analogy: Think of **tailor converting measurements** (درزی ناپ بدلتا ہے):

```
Input:  [32, 34, 36, 38] (inches)
Process: Convert each to cm (× 2.54)
Output: [81.28, 86.36, 91.44, 96.52] (cm)
```

map() transforms each item and returns a NEW array!

How map() Works

```
// THINKING: Transform all items

const prices = [100, 200, 300];

// Add 10% tax to each price
const withTax = prices.map(price ⇒ {
    return price * 1.1;
});

console.log(prices);   // [100, 200, 300] (original unchanged!)
console.log(withTax);  // [110, 220, 330] (new array!)
```

Real Example: Format Currency

```
const amounts = [500, 1500, 2500];

const formatted = amounts.map(amount => {
    return "Rs. " + amount;
});

console.log(formatted);
// ["Rs. 500", "Rs. 1500", "Rs. 2500"]
```

What is filter()? (کیا ہے؟)

Urdu Analogy: Think of **sorting rice** (چاول چھاننا):

```
Input: [good grain, stone, good grain, bad grain, good grain]
Filter: Keep only good grains
Output: [good grain, good grain, good grain]
```

filter() selects items that pass a test!

How filter() Works

```
// THINKING: Keep only items that match condition

const numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

// Get only even numbers
const evenNumbers = numbers.filter(num => {
    return num % 2 === 0;
});

console.log(evenNumbers);  // [2, 4, 6, 8, 10]
```

Real Example: Filter by City

```
const cities = ["Lahore", "Karachi", "Larkana", "Islamabad", "Lyallpur"];

// Get cities starting with 'L'
const lCities = cities.filter(city => {
    return city.startsWith("L");
});

console.log(lCities);
// ["Lahore", "Larkana", "Lyallpur"]
```

Combining map() and filter()

```
// THINKING: Chain methods together!

const prices = [50, 150, 300, 80, 250];

// Get expensive items (>100) and add tax
const expensiveWithTax = prices
    .filter(price => price > 100)
    .map(price => price * 1.1);

console.log(expensiveWithTax);
// [165, 330, 275] (only items >100, with 10% tax)
```

Your First Example

```
const marks = [45, 78, 92, 33, 67, 88];

// TODO: Use filter to get passing marks (≥50)
const passing = marks.filter(mark => {
    return mark _____ 50;
});

console.log("Passing:", passing);

// TODO: Use map to convert marks to percentages
const percentages = marks.map(mark => {
    return mark + "%";
});

console.log("Percentages:", percentages);

// TODO: Get students who scored above 80
const excellent = marks.filter(mark => mark _____ 80);
console.log("Excellent:", excellent);
```

Common Mistakes

❌ **Wrong:**

```
const doubled = numbers.map(num => {
    num * 2;  // Missing return!
});
// Result: [undefined, undefined, undefined]
```

✅ **Right:**

```
const doubled = numbers.map(num => {
    return num * 2;
});
// OR (arrow function shorthand):
const doubled = numbers.map(num => num * 2);
```

Check Your Understanding

- ☐ What does map() return?
- ☐ What does filter() return?
- ☐ Does map() change the original array?
- ☐ Can you chain map() and filter()?

💻 Practice Session: Array Mastery

🎯 Practice Goal

By the end of this section, you'll manipulate arrays like a pro!

Exercise 1: Pakistani Cities (ہم ساتھ کریں)

Scenario: Manage a list of Pakistani cities

Starter Code:

```
// TODO Step 1: Create array
const cities = ["Lahore", "Karachi", "Islamabad"];

// TODO Step 2: Add "Peshawar" to end
cities._____(______);

// TODO Step 3: Add "Quetta" to end
cities._____(______);

// TODO Step 4: Print all cities using forEach
cities.forEach(city => {
    console.log("• " + _____);
});

// TODO Step 5: Get cities starting with 'L'
const lCities = cities.filter(city => {
    return city._____(_____)
});

console.log("Cities with L:", lCities);
```

Exercise 2: Shopping Cart Total (اب آپ)

Problem: Calculate total price of items in cart

Requirements:

- ☐ Array of prices
- ☐ Calculate total using forEach
- ☐ Apply 15% discount to all items using map
- ☐ Filter items over Rs. 1000

Starter Code:

```
const cartPrices = [500, 1500, 350, 2500, 750];

// TODO: Calculate total
let total = 0;
cartPrices.forEach(price => {
    total _____ price;
});
console.log("Total: Rs.", total);

// TODO: Apply 15% discount
const discounted = cartPrices.map(price => {
    return price _____ 0.85;
});
console.log("After discount:", discounted);

// TODO: Get expensive items (>1000)
const expensive = cartPrices.filter(price => {
    return price _____ 1000;
});
console.log("Expensive items:", expensive);
```

Don't Look Below Until You Try! ⬇️

Hints (if stuck):

▶ Stuck on total?

```
cartPrices.forEach(price => {
    total += price;   // Same as total = total + price
});
```

▶ Stuck on discount?

```
const discounted = cartPrices.map(price => {
    return price * 0.85;   // 85% of original (15% off)
});
```

Exercise 3: Student Grades

Problem: Process student grades

```
const students = [
    { name: "Ali", marks: 85 },
    { name: "Sara", marks: 92 },
    { name: "Hassan", marks: 78 },
    { name: "Fatima", marks: 45 }
];

// TODO: Get all student names
const names = students.map(student => {
    return student._____;
});
console.log("Names:", names);

// TODO: Get students who passed (marks >= 50)
const passed = students.filter(student => {
    return student._____ >= 50;
});
console.log("Passed:", passed);

// TODO: Get names of students who scored above 80
const toppers = students
    .filter(student => student.marks _____ 80)
    .map(student => student._____);

console.log("Toppers:", toppers);
```

🚀 آج کا چیلنج (Today's Challenge)

Project: Daraz Product Manager

ڈراز پروڈکٹ مینیجر

The Problem:
You're managing inventory for a Daraz-style e-commerce platform! Build a product management system that handles adding products, removing out-of-stock items, calculating inventory value, finding expensive products, and filtering by category!

What You're Building:
A console-based inventory system with array methods

Success Criteria:

- [ ] Stores products as array of objects
- [ ] Can add new products
- [ ] Filters out-of-stock items
- [ ] Calculates total inventory value
- [ ] Finds most expensive product
- [ ] Filters by category
- [ ] Has search function

Phase 1: Planning (سوچیں پہلے)

Before coding, answer:

1. **What structure for products?**

```
{
    name: "Laptop",
    price: 75000,
    stock: 5,
    category: "Electronics"
}
```

2. **What operations do I need?**

    - Add product (push)
    - Remove out-of-stock (filter)

    - Calculate total value (forEach or reduce)
    - Find max price (loop and compare)
    - Filter by category (filter)
    - Search by name (filter)

3. **How will I test it?**
    - Start with sample products
    - Test each function separately
    - Check console output

Planning Checkpoint:

- ☐ I understand the product structure
- ☐ I know which array methods to use
- ☐ I have test cases in mind

Phase 2: Foundation (بنیاد)

Starter Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Daraz Product Manager</title>
</head>
<body>
    <h1>🛒 Daraz Product Manager</h1>
    <h2>ڈراز پروڈکٹ مینیجر</h2>
    <h3>Press F12 to see inventory management in action!</h3>

    <script>
        // ===================================
        // DARAZ PRODUCT MANAGER
        // By: [Your Name]
        // Date: [Today's Date]
        // ===================================

        console.log("🛒 ڈراز پروڈکٹ مینیجر");
        console.log("==============================\n");

        // ========== INITIAL INVENTORY ==========

        let products = [
            { name: "Laptop HP", price: 75000, stock: 5, category:
"Electronics" },
            { name: "Mouse Wireless", price: 1200, stock: 15, category:
"Electronics" },
            { name: "Office Chair", price: 8500, stock: 0, category:
"Furniture" },
            { name: "Desk Lamp", price: 2500, stock: 8, category:
"Furniture" },
            { name: "Notebook A4", price: 150, stock: 50, category:
"Stationery" },
            { name: "Pen Set", price: 300, stock: 0, category: "Stationery"
},
            { name: "Water Bottle", price: 450, stock: 20, category:
"Accessories" }
        ];

        console.log("📦 INITIAL INVENTORY");
        console.log("Total products:", products.length);
        console.log("");

        // ========== FUNCTION 1: Display All Products ==========

        function displayProducts(productList) {
```

```
        console.log("📋 PRODUCT LIST:");
        console.log("━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━");

        // TODO: Use forEach to display each product
        productList.forEach((product, index) => {
            const status = product.stock > 0 ? "✅ In Stock" : "❌ Out
of Stock";
            console.log(`${index + 1}. ${product.name}`);
            console.log(`   Price: Rs. ${product.price}`);
            console.log(`   Stock: ${product.stock} units`);
            console.log(`   Category: ${product.category}`);
            console.log(`   Status: ${status}`);
            console.log("---");
        });
        console.log("");
    }

    displayProducts(products);

    // ========== FUNCTION 2: Add New Product ==========

    function addProduct(name, price, stock, category) {
        // TODO: Create product object
        const newProduct = {
            name: _____,
            price: _____,
            stock: _____,
            category: _____
        };

        // TODO: Add to products array
        products._____(newProduct);

        console.log(`✅ Added: ${name}`);
    }

    console.log("➕ ADDING NEW PRODUCTS");
    addProduct("Keyboard Mechanical", 5500, 10, "Electronics");
    addProduct("Monitor 24\"", 22000, 3, "Electronics");
    console.log("");

    // ========== FUNCTION 3: Remove Out-of-Stock Items ==========

    function removeOutOfStock() {
        console.log("🗑️ REMOVING OUT-OF-STOCK ITEMS");

        // TODO: Count out-of-stock items
        const outOfStock = products.filter(product => {
            return product.stock _____ 0;
        });

        console.log(`Found ${outOfStock.length} out-of-stock items:`);
        outOfStock.forEach(product => {
            console.log(`• ${product.name}`);
        });

        // TODO: Filter to keep only in-stock items
        products = products.filter(product => {
            return product.stock _____ 0;
        });

        console.log(`✅ Removed! Now ${products.length} products in
stock`);
        console.log("");
    }

    removeOutOfStock();

    // ========== FUNCTION 4: Calculate Total Inventory Value
==========

    function calculateTotalValue() {
        // TODO: Calculate total value (price × stock for each item)
        let totalValue = 0;

        products.forEach(product => {
            totalValue _____ product.price _____ product.stock;
        });

        console.log("💰 TOTAL INVENTORY VALUE");
        console.log(`Rs. ${totalValue.toLocaleString()}`);
        console.log("");

        return totalValue;
    }

    calculateTotalValue();

    // ========== FUNCTION 5: Find Most Expensive Product ==========

    function findMostExpensive() {
        // TODO: Find product with highest price
        let maxPrice = 0;
        let expensive = null;

        products.forEach(product => {
            if (product.price _____ maxPrice) {
                maxPrice = product.price;
                expensive = product;
            }
        });

        console.log("💎 MOST EXPENSIVE PRODUCT");
        console.log(`${expensive.name}: Rs. ${expensive.price}`);
        console.log("");

        return expensive;
    }

    findMostExpensive();

    // ========== FUNCTION 6: Filter by Category ==========

    function filterByCategory(category) {
        // TODO: Get all products in this category
        const filtered = products.filter(product => {
            return product.category _____ category;
        });

        console.log(`🏷️ ${category.toUpperCase()} PRODUCTS`);
        console.log(`Found ${filtered.length} items:`);

        filtered.forEach(product => {
            console.log(`• ${product.name} - Rs. ${product.price}`);
        });
        console.log("");

        return filtered;
    }

    filterByCategory("Electronics");
    filterByCategory("Furniture");

    // ========== FUNCTION 7: Search Products ==========

    function searchProducts(searchTerm) {
        // TODO: Find products containing search term
        const results = products.filter(product => {
            // Convert to lowercase for case-insensitive search
            return
product.name.toLowerCase().includes(searchTerm.toLowerCase());
        });

        console.log(`🔍 SEARCH: "${searchTerm}"`);

        if (results.length === 0) {
            console.log("No products found");
        } else {
            console.log(`Found ${results.length} result(s):`);
            results.forEach(product => {
                console.log(`• ${product.name} - Rs.
${product.price}`);
            });
        }
        console.log("");

        return results;
    }

    searchProducts("Laptop");
    searchProducts("Mouse");
    searchProducts("Chair");

    // ========== FUNCTION 8: Get Product Names Only ==========

    function getProductNames() {
        // TODO: Use map to get array of just names
        const names = products.map(product => {
            return product._____;
        });

        console.log("📝 ALL PRODUCT NAMES");
        console.log(names.join(", "));
        console.log("");

        return names;
    }

    getProductNames();

    // ========== FUNCTION 9: Apply Discount ==========

    function applyDiscount(percent) {
        // TODO: Create new array with discounted prices
        const discounted = products.map(product => {
            return {
                ...product,  // Copy all properties
                originalPrice: product.price,
                price: product.price _____ (1 - percent / 100)
            };
        });

        console.log(`💸 ${percent}% DISCOUNT APPLIED`);
        console.log("Sample:");
        discounted.slice(0, 3).forEach(product => {
            console.log(`${product.name}`);
            console.log(`  Was: Rs. ${product.originalPrice}`);
            console.log(`  Now: Rs. ${product.price.toFixed(2)}`);
        });
        console.log("");

        return discounted;
    }

    applyDiscount(15);  // 15% off!

    // ========== SUMMARY ==========

    console.log("==============================");
    console.log("📊 INVENTORY SUMMARY");
    console.log("==============================");
    console.log(`Total Products: ${products.length}`);
    console.log(`Total Categories: ${[...new Set(products.map(p ⇒
p.category))].length}`);
    console.log(`Total Value: Rs.
${calculateTotalValue().toLocaleString()}`);
    console.log(`Most Expensive: ${findMostExpensive().name}`);
    console.log("==============================");

</script>
</body>
</html>
```

Phase 3: Milestones (سنگ میل)

Milestone 1: Basic Display Works ✅

- ☐ Products array created
- ☐ displayProducts shows all items
- ☐ Each product shows correctly
- Test: Can you see all 7 products?

Milestone 2: Add/Remove Works ✅

- ☐ addProduct adds new items
- ☐ removeOutOfStock filters correctly

- ☐ Array length changes correctly
- Test: Add 2 products, should have 9 total

Milestone 3: Calculations Work ✅

- ☐ Total value calculates correctly
- ☐ Most expensive product found
- ☐ Math is accurate
- Test: Manual check total value

Milestone 4: Filtering Works ✅

- ☐ Category filter returns correct items
- ☐ Search function finds products
- ☐ Case-insensitive search works
- Test: Search "laptop" and "LAPTOP" both work

Debugging Guide (اگر پھنس جائیں)

Problem: forEach not working

- ☐ Check: Are you using parentheses? `.forEach()`
- ☐ Check: Is the parameter name consistent?
- ☐ Check: Did you use arrow function correctly?

Problem: filter returns empty array

- ☐ Check: Is your condition correct?
- ☐ Check: Are you using === not =?
- ☐ Add console.log inside filter to see what's happening

Problem: map returns undefined

- ☐ Check: Did you use `return` keyword?

- ☐ Check: Are you returning the right thing?
- ☐ Check: Arrow function shorthand: `item ⇒ item.prop`

Common Logic Issues:

```
// ❌ WRONG: Forgot return
const names = products.map(product ⇒ {
    product.name;  // Missing return!
});

// ✅ RIGHT:
const names = products.map(product ⇒ {
    return product.name;
});
// OR
const names = products.map(product ⇒ product.name);
```

Extension Challenges (بونس چیلنج)

If you finish early:

🌟 Level 1: Advanced Filters

```
// Get products between price range
function getByPriceRange(min, max) {
    return products.filter(p ⇒ p.price ⩾ min && p.price ⩽ max);
}

// Sort by price (low to high)
function sortByPrice() {
    return [...products].sort((a, b) ⇒ a.price - b.price);
}
```

🌟🌟 Level 2: Statistics

```javascript
// Calculate average price
function getAveragePrice() {
    const total = products.reduce((sum, p) => sum + p.price, 0);
    return total / products.length;
}

// Get low stock items (< 5)
function getLowStock() {
    return products.filter(p => p.stock < 5 && p.stock > 0);
}
```

🌟🌟🌟 Level 3: Advanced Features

```javascript
// Group products by category
function groupByCategory() {
    const grouped = {};
    products.forEach(product => {
        if (!grouped[product.category]) {
            grouped[product.category] = [];
        }
        grouped[product.category].push(product);
    });
    return grouped;
}

// Generate inventory report
function generateReport() {
    // Create detailed report with all statistics
}
```

📝 Daily Quiz (5 منٹ کا ٹیسٹ)

Instructions: Answer WITHOUT looking at notes!

1. What index is the first item in an array?

- A) 1

- B) 0
- C) -1
- D) Depends on array

▶ See Answer (Try first!)

Answer: B - 0. Arrays are zero-based, meaning the first item is at index 0. Remember: first item = index 0, second item = index 1, etc.

---

2. What does push() do?

- A) Removes first item
- B) Adds item to start
- C) Adds item to end
- D) Removes last item

▶ See Answer (Try first!)

Answer: C - Adds item to end. Think of joining a queue (قطار) at the end. `push()` adds to end, `pop()` removes from end.

---

3. What's the difference between map() and forEach()?

- A) They're the same
- B) map() returns new array, forEach() doesn't
- C) forEach() is faster
- D) map() only works with numbers

▶ See Answer (Try first!)

Answer: B - map() transforms each item and returns a NEW array. forEach() just loops through items and returns nothing. Use map() when you want to transform data!

---

4. What will this return?

```
const arr = [1, 2, 3, 4, 5];
const result = arr.filter(num => num > 3);
```

- A) [4, 5]
- B) [1, 2, 3]
- C) [3, 4, 5]
- D) [1, 2, 3, 4, 5]

▶ See Answer (Try first!)

Answer: A - [4, 5]. filter() keeps only items that pass the test. Only 4 and 5 are greater than 3, so only they remain.

---

5. How do you get the last item of array `arr`?

- A) arr[arr.length]
- B) arr[-1]
- C) arr[arr.length - 1]
- D) arr.last()

▶ See Answer (Try first!)

Answer: C - arr[arr.length - 1]. If array has 5 items, length is 5, but last index is 4 (zero-based). So length - 1 gives you the last index!

Scoring:

- **5/5:** 🎉 Array Master! You crushed it!
- **4/5:** 👏 Excellent! Review the one you missed

- **3/5:** 👍 Good! Practice array methods more
- ❤️ **/5:** 💪 Review all array concepts again

🎓 Today's Homework (گھر کا کام)

Required (لازمی):

- ☐ Complete the Daraz Product Manager
- ☐ Test all functions with different data
- ☐ Explain to a family member: "What is an array?" using aloo paratha analogy

Optional (اختیاری):

- ☐ Try the extension challenges
- ☐ Create a "Contact Book" using array of objects
- ☐ Build a "Todo List" with array methods
- ☐ Make a "Student Grade Manager" similar to Product Manager

For Tomorrow:

- ☐ Think about: "How would I store related data together?"
- ☐ Arrays hold lists, but what about complex data?
- ☐ Tomorrow: Objects & JSON!

💭 Daily Reflection (روزانہ کی سوچ)

آج میں نے کیا سیکھا (What I Learned Today):

مشکل کیا لگا (What I Found Difficult):

مزید کیا سیکھنا ہے (What I Want to Explore More):

My Confidence Level (1-10): ________

🔄 Tomorrow's Preview

Tomorrow we'll learn about **Objects & JSON** where you'll build a **Restaurant Menu System**!

You'll learn how to:

- Store complex data in objects
- Access object properties
- Work with nested objects
- Convert between objects and JSON
- Build real-world data structures

Get Ready By:

- ☐ Making sure your Product Manager works
- ☐ Thinking: What if product needed more data? (images, reviews, ratings)
- ☐ Objects are the answer!

📚 Resources (اگر مزید پڑھنا ہو)

Free Resources (3G-Friendly):

📖 **MDN - Arrays**

- Link: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array
- Best for: Complete array methods reference

🎥 **JavaScript Arrays in Urdu**

- Search: "JavaScript array methods Urdu map filter"
- Best for: Visual learners

💻 **Practice Arrays**

- Create arrays for: shopping lists, student names, cricket scores
- Practice: forEach, map, filter with real data

CodeSensei's Tip of the Day: 💡

"Arrays are everywhere in real apps. Master map() and filter() - they're used constantly in professional code. When you see a list of anything on a website, that's an array being displayed. Instagram posts? Array. YouTube videos? Array. Daraz products? Array. Think in arrays!"

"Arrays ہر جگہ ہیں۔ map() اور filter() سیکھ لو - پروفیشنل کوڈ میں ہر جگہ استعمال ہوتے ہیں۔"

👥 Team Activity (Tomorrow's Standup)

Tomorrow at 7:00 PM, be ready to share:

1. Your Product Manager's total inventory value
2. One array method you found most useful
3. Explain the difference between push() and unshift() in Urdu
4. One question about arrays

کوڈ سیکھنا ایک سفر ہے، منزل نہیں۔ ہر دن ایک قدم آگے۔

"Learning to code is a journey, not a destination. One step forward every day."

Day 6 Complete! See you tomorrow for Objects & JSON! 🚀

اللہ حافظ! Tomorrow we structure our data better! 📦