Day 4: Functions & Scope

فنکشنز اور سکوپ

> **Quote of the Day:** "*Functions are the building blocks of programs. They're like recipes - write once, use many times.*" - Douglas Crockford
>
> "*فنکشنز پروگرامز کی بنیاد ہیں۔ یہ ریسیپیوں کی طرح ہیں - ایک بار لکھو، بار بار استعمال کرو۔*"

📋 Today's Learning Goals (آج کے اہداف)

By the end of today, you will:

- ☐ Declare and invoke (call) functions confidently
- ☐ Understand the difference between parameters and arguments
- ☐ Use return statements to get values back from functions
- ☐ Understand scope (where variables are accessible)
- ☐ Build a Utility Functions Library with reusable code

Time Breakdown (کل وقت: 150 منٹ)

- 🕐 7:00-7:05 PM (5min): Standup - Share your Cricket Counter scores!
- 🕐 7:05-8:05 PM (60min): Understanding functions (3× Pomodoro)
- 🕐 8:05-8:50 PM (45min): Practice creating utility functions
- 🕐 8:50-9:25 PM (35min): Build your Functions Library
- 🕐 9:25-9:30 PM (5min): Quiz & reflection

🎯 What We're Building Today

Today you'll create a **Utility Functions Library** - a collection of reusable helper functions like currency converter, discount calculator, phone formatter, and email validator!

Why This Matters for Your Career: Every professional codebase uses functions extensively:

- **Daraz:** `calculateShippingCost()`, `applyDiscountCode()`, `formatPrice()`

- **Careem:** `calculateFare()`, `estimateArrivalTime()`, `validatePhoneNumber()`
- **JazzCash:** `convertBalance()`, `checkTransactionLimit()`, `formatAccountNumber()`

Functions make code reusable, organized, and easier to fix!

🧠 سمجھنا (Understanding): Why Functions Exist

The Real-World Analogy

Scenario: Making Biryani

Imagine you're teaching someone to make biryani. You could:

Without Functions (repeating everything):

```
To make chicken biryani:
1. Boil rice with salt and oil
2. Fry onions until golden
3. Mix spices: garam masala, red chili, turmeric
4. Cook chicken with spices
5. Layer rice and chicken
6. Cook on dum for 30 minutes

To make mutton biryani:
1. Boil rice with salt and oil        (SAME AS ABOVE!)
2. Fry onions until golden             (SAME AS ABOVE!)
3. Mix spices: garam masala, red chili, turmeric  (SAME!)
4. Cook mutton with spices
5. Layer rice and mutton
6. Cook on dum for 30 minutes
```

With Functions (reusable recipes):

```
Function: prepareRice()
Function: fryOnions()
Function: mixSpices()
Function: cookOnDum()

To make chicken biryani:
- prepareRice()
- fryOnions()
- mixSpices()
- cookMeat(chicken)
- layer()
```

```
- cookOnDum()

To make mutton biryani:
- prepareRice()        (REUSE!)
- fryOnions()          (REUSE!)
- mixSpices()          (REUSE!)
- cookMeat(mutton)
- layer()
- cookOnDum()          (REUSE!)
```

This is exactly what functions do in programming!

Daily Life Examples

1. **Calculator App:**

```
function add(a, b) - جمع کرنا
function subtract(a, b) - منفی کرنا
function multiply(a, b) - ضرب کرنا
function divide(a, b) - تقسیم کرنا
```

2. **ATM Machine:**

```
function checkBalance()
function withdraw(amount)
function deposit(amount)
function printReceipt()
```

3. **Phone Contacts:**

```
function saveContact(name, number)
function searchContact(name)
function deleteContact(name)
function formatNumber(digits)
```

Why Does This Matter?

Without functions:

- Copy-paste same code everywhere
- Fix one bug? Fix it 100 times!
- Hard to read and maintain
- Messy, confusing code

With functions:

- Write once, use anywhere
- Fix bug once, fixed everywhere!
- Clean, organized code
- Easy to understand and update

The Mental Model

Think of a function like a **vending machine**:

```
INPUT (پیسے) → MACHINE (process) → OUTPUT (drink)
    ↓                ↓                    ↓
Arguments      Function Body         Return Value
```

You put something IN, machine does its job, you get something OUT!

📚 Building Block #1: Function Declaration (فنکشن بنانا)

What is a Function? (کیا ہے؟)

Urdu Analogy: Think of a function like a **dabbawala** (ڈبہ والا) - lunch box delivery person.

```
You give: lunch box (input)
Dabbawala: delivers it (process)
You get: delivered lunch box (output)
```

In JavaScript:

```
function deliverLunch(lunchBox) {
    // Process: take lunch, deliver it
    return "Lunch delivered to office!";
}
```

The Anatomy of a Function

```
function functionName(parameter1, parameter2) {
    // Function body - code to execute
    // Do something with parameters
    return result;  // Send back a value
}
```

Part	What It Does	Example	Urdu
`function`	Keyword to create function	`function`	فنکشن کی شروعات
Name	What you call it	`greetUser`	نام
Parameters	Inputs (placeholders)	`(name, age)`	ان پٹ
Body	Code that runs	`{ ... }`	کام
Return	Output value	`return "Hello"`	آؤٹ پٹ

How It Works - Step by Step

```
// THINKING: Create a simple greeting function

// Step 1: DECLARE (بنانا) - Define the function
function sayHello() {
    console.log("السلام علیکم");
    console.log("Hello!");
}

// Step 2: INVOKE/CALL (چلانا) - Use the function
sayHello();  // This runs the function!

// Output:
// السلام علیکم
// Hello!
```

The pattern:

```
// DECLARE - this just creates it, doesn't run it
function doSomething() {
    // code here
}

// INVOKE - this actually runs it
doSomething();  // Add () to call it!
```

Your First Example

```
// THINKING: Function to greet in Urdu and English

function greetInUrdu() {
    console.log("خوش آمدید!");
    console.log("Welcome!");
}
```

```
// Call it!
greetInUrdu();

// TODO: Create a function called introduceSelf
// It should print your name and city
function introduceSelf() {
    console.log("My name is _______");
    console.log("I live in _______");
}

// TODO: Call your function
_____________();
```

Multiple Calls - Reusability!

```
// THINKING: Call the same function multiple times

function ringBell() {
    console.log("🔔 Ding dong!");
}

// Ring the bell 3 times
ringBell();  // 🔔 Ding dong!
ringBell();  // 🔔 Ding dong!
ringBell();  // 🔔 Ding dong!

// See the power? Write once, call many times!
```

Common Mistakes

❌ Wrong:

```
function sayHello {  // Missing parentheses!
    console.log("Hello");
}
```

✅ Right:

```
function sayHello() {  // Always need ()
    console.log("Hello");
}
```

❌ Wrong:

```
function sayHello() {
    console.log("Hello");
}

sayHello;  // Forgot () - doesn't call it!
```

✅ **Right:**

```
function sayHello() {
    console.log("Hello");
}

sayHello();  // Need () to actually call it
```

Check Your Understanding

- ☐ What keyword creates a function?
- ☐ How do you call/invoke a function?
- ☐ Can you call a function multiple times?
- ☐ What's inside the curly braces `{ }` ?

📚 Building Block #2: Parameters & Arguments

What's the Difference? (فرق کیا ہے؟)

Urdu Analogy: Think of ordering food at a restaurant:

Menu (Parameters - خالی جگہیں):

```
Biryani (what type?)
Drink (which one?)
Spice Level (how spicy?)
```

Your Order (Arguments - اصل چیزیں):

```
Biryani (chicken)
Drink (Coke)
Spice Level (medium)
```

Parameters = Placeholders in function definition **Arguments** = Actual values when calling function

How It Works

```
// THINKING: Function with parameters

// PARAMETERS - placeholders (خالی ڈبے)
function greetUser(userName, userCity) {
    console.log("Hello " + userName);
    console.log("Welcome from " + userCity);
}

// ARGUMENTS - actual values (بھرے ہوئے ڈبے)
greetUser("Hassan", "Lahore");
// Output:
// Hello Hassan
// Welcome from Lahore

greetUser("Fatima", "Karachi");
// Output:
// Hello Fatima
// Welcome from Karachi
```

Multiple Parameters

```
// THINKING: Calculate area of rectangle

function calculateArea(width, height) {
    const area = width * height;
    console.log("Area:", area);
}

// Different rectangles
calculateArea(5, 10);   // Area: 50
calculateArea(7, 3);    // Area: 21
calculateArea(20, 15);  // Area: 300
```

Parameter Order Matters!

```
function introducePerson(name, age, city) {
    console.log(name + " is " + age + " years old");
    console.log("Lives in " + city);
}

// Correct order
introducePerson("Ali", 20, "Lahore");
// Ali is 20 years old
```

```
// Lives in Lahore

// Wrong order - confusing!
introducePerson(20, "Lahore", "Ali");
// 20 is Lahore years old  (❌ Nonsense!)
// Lives in Ali
```

Your First Example

```
// TODO: Create function to introduce student
function introduceStudent(name, rollNo, program) {
    console.log("Student Name: " + _______);
    console.log("Roll Number: " + _______);
    console.log("Program: " + _______);
}

// TODO: Call with your details
introduceStudent(_______, _______, _______);

// TODO: Call with friend's details
introduceStudent(_______, _______, _______);
```

Default Parameters

```
// THINKING: What if user doesn't provide a value?

function greetUser(name = "Guest") {
    console.log("Welcome, " + name + "!");
}

greetUser("Hassan");  // Welcome, Hassan!
greetUser();          // Welcome, Guest! (uses default)
```

Common Mistakes

❌ Wrong:

```
function greet(name) {
    console.log("Hello " + userName);  // Wrong variable!
}

greet("Ali");
```

✅ Right:

```
function greet(name) {
    console.log("Hello " + name);  // Use parameter name
}

greet("Ali");
```

❌ **Wrong:**

```
function add(a, b) {
    console.log(a + b);
}

add(5);  // b is undefined!
// Output: NaN (Not a Number)
```

✅ **Right:**

```
function add(a, b) {
    console.log(a + b);
}

add(5, 3);  // Provide both arguments
// Output: 8
```

Check Your Understanding

- ☐ What are parameters?
- ☐ What are arguments?
- ☐ Does order matter?
- ☐ What happens if you don't pass an argument?

📑 Building Block #3: Return Statements (واپس بھیجنا)

What is Return? (کیا ہے؟)

Urdu Analogy: Think of a function like **photocopying a document**:

```
You give: original document (input)
Machine: makes copy (process)
You get: photocopy back (RETURN!)
```

Without return, the machine just does its job but you get NOTHING back!

How It Works

```
// THINKING: Function WITHOUT return

function add(a, b) {
    const sum = a + b;
    console.log(sum);
}

const result = add(5, 3);
console.log("Result:", result);

// Output:
// 8                    (from console.log inside)
// Result: undefined    (function returned nothing!)
```

```
// THINKING: Function WITH return

function add(a, b) {
    const sum = a + b;
    return sum;  // Send value back!
}

const result = add(5, 3);
console.log("Result:", result);

// Output:
// Result: 8    (got the value back!)
```

Why Return Matters

Without return - can only console.log:

```
function calculateDiscount(price, percent) {
    const discount = price * (percent / 100);
    console.log(discount);  // Just prints it
}

calculateDiscount(1000, 10);  // Shows: 100
// But can't use this value anywhere else!
```

With return - can use the value:

```
function calculateDiscount(price, percent) {
    const discount = price * (percent / 100);
    return discount;  // Send it back!
}

const saved = calculateDiscount(1000, 10);
console.log("You saved Rs.", saved);
const finalPrice = 1000 - saved;
console.log("Pay:", finalPrice);
```

Return Stops Function Immediately

```
// THINKING: Code after return doesn't run!

function checkAge(age) {
    if (age < 18) {
        return "Too young";
        console.log("This never runs!");  // Unreachable!
    }
    return "Old enough";
}

console.log(checkAge(15));  // Too young
console.log(checkAge(20));  // Old enough
```

Your First Example

```
// TODO: Function to calculate square
function calculateSquare(number) {
    const square = number _____ number;
    _______ square;  // Return the result!
}

// TODO: Use the returned value
const result = calculateSquare(5);
console.log("Square of 5 is:", result);

// TODO: Create function to check if number is even
function isEven(num) {
    if (num _____ 2 === 0) {
        return _______;
    } else {
        return _______;
    }
}
```

```
console.log(isEven(4));   // Should return: true
console.log(isEven(7));   // Should return: false
```

Common Mistakes

❌ Wrong:

```
function add(a, b) {
    return a + b;
    const sum = a + b;  // Never runs!
}
```

✅ Right:

```
function add(a, b) {
    const sum = a + b;
    return sum;  // Return at the end
}
```

❌ Wrong:

```
function greet(name) {
    "Hello " + name;  // Missing return!
}

const message = greet("Ali");
console.log(message);  // undefined
```

✅ Right:

```
function greet(name) {
    return "Hello " + name;  // Must use return
}

const message = greet("Ali");
console.log(message);  // Hello Ali
```

Check Your Understanding

- ☐ What does return do?
- ☐ Can you use a returned value in other code?
- ☐ What happens to code after return?

- ☐ What does function return if you don't use return keyword?

📚 Building Block #4: Scope (دائرہ کار)

What is Scope? (کیا ہے؟)

Urdu Analogy: Think of your **house (گھر)** and **rooms (کمرے)**:

```
HOUSE (گھر) = Global Scope
├── Living Room (سٹنگ روم) = Function Scope
│   └── TV (only in this room!)
├── Kitchen (باورچی خانہ) = Function Scope
│   └── Stove (only in this room!)
└── Bedroom (کمرہ) = Function Scope
    └── Bed (only in this room!)

Things in HOUSE = available everywhere
Things in ROOM = only available in that room
```

Global vs Local Scope

```javascript
// THINKING: Variables and their scope

// GLOBAL scope - available everywhere (گھر میں)
const houseName = "Khan House";

function livingRoom() {
    // LOCAL scope - only inside this function (سٹنگ روم میں)
    const tv = "55 inch TV";
    console.log(houseName);  // ✅ Can access global
    console.log(tv);         // ✅ Can access local
}

function kitchen() {
    // LOCAL scope - different function (باورچی خانے میں)
    const stove = "Gas stove";
    console.log(houseName);  // ✅ Can access global
    // console.log(tv);      // ❌ Can't access livingRoom's tv
}

livingRoom();
console.log(houseName);  // ✅ Can access global
// console.log(tv);      // ❌ Can't access function's local variable
```

Block Scope (with let/const)

```
// THINKING: Variables inside { } blocks

function checkAge(age) {
    if (age >= 18) {
        // Block scope - only inside these { }
        const message = "You can vote";
        console.log(message);  // ✅ Works here
    }

    // console.log(message);  // ❌ Error! message doesn't exist here
}
```

Why Scope Matters

Good scope - no conflicts:

```
function calculateTax() {
    const rate = 0.15;  // Local to this function
    return price * rate;
}

function calculateDiscount() {
    const rate = 0.10;  // Different rate, no problem!
    return price * rate;
}

// Both functions can have their own "rate" variable!
```

The Scope Chain

```
// THINKING: JavaScript looks for variables in order

const city = "Lahore";  // Global

function showLocation() {
    const country = "Pakistan";  // Function scope

    function showDetails() {
        const area = "DHA";  // Nested function scope

        // Can access all three!
        console.log(area);     // ✅ Local
        console.log(country);  // ✅ Parent function
```

```
        console.log(city);     // ✅ Global
    }

    showDetails();
    // console.log(area);  // ❌ Can't access child's variable
}

showLocation();
```

Your First Example

```
// TODO: Practice with scope

const university = "Superior University";  // Global

function introduceStudent() {
    const studentName = "Hassan";  // Local

    console.log("University:", _______);  // Can you access global?
    console.log("Student:", _______);     // Can you access local?
}

introduceStudent();

// TODO: Try to access studentName here - what happens?
// console.log(studentName);  // Will this work?
```

Common Mistakes

❌ Wrong:

```
function greet() {
    message = "Hello";  // No const/let! Creates global (bad!)
}

greet();
console.log(message);  // Works but BAD PRACTICE!
```

✅ Right:

```
function greet() {
    const message = "Hello";  // Properly scoped
    return message;
}
```

```
const greeting = greet();
console.log(greeting);  // Use return instead
```

❌ **Wrong:**

```
function calculate() {
    const result = 100;
}

calculate();
console.log(result);  // ❌ Error! result is local
```

✅ **Right:**

```
function calculate() {
    const result = 100;
    return result;  // Return it!
}

const answer = calculate();
console.log(answer);  // ✅ Works!
```

Check Your Understanding

- ☐ What is global scope?
- ☐ What is local scope?
- ☐ Can functions access global variables?
- ☐ Can code outside function access local variables?

💻 Practice Session: Building Utility Functions

🎯 Practice Goal

By the end of this section, you'll create reusable utility functions!

Exercise 1: Guided Practice (ہم ساتھ کریں)

Scenario: Create a function to format Pakistani currency

Starter Code:

```
// TODO Step 1: Create function
// HINT: Takes amount (number), returns formatted string
function formatCurrency(amount) {
    // TODO Step 2: Add "Rs." prefix
    const formatted = "Rs. " + amount;

    // TODO Step 3: Return the result
    _______ formatted;
}

// TODO Step 4: Test it!
console.log(formatCurrency(500));    // Rs. 500
console.log(formatCurrency(1250));   // Rs. 1250
console.log(formatCurrency(99.50));  // Rs. 99.5
```

Exercise 2: Temperature Converter (اب آپ)

Problem: Convert Celsius to Fahrenheit

Formula: F = (C × 9/5) + 32

Requirements:

- ☐ Function named convertToFahrenheit
- ☐ Takes celsius as parameter
- ☐ Returns fahrenheit value
- ☐ Works with decimals

Starter Code:

```
function convertToFahrenheit(celsius) {
    // TODO: Apply formula
    const fahrenheit = (celsius _____ 9/5) _____ 32;

    // TODO: Return result
    _______ fahrenheit;
}

// Test cases
console.log(convertToFahrenheit(0));    // Should be 32
console.log(convertToFahrenheit(100));  // Should be 212
console.log(convertToFahrenheit(37));   // Should be 98.6
```

Don't Look Below Until You Try! ⬇️

Hints (if stuck):

▶ Stuck on formula?

```
const fahrenheit = (celsius * 9/5) + 32;
return fahrenheit;
```

Exercise 3: String Utilities

Problem: Create two string functions

```
// Function 1: Convert to UPPERCASE
function shout(text) {
    // TODO: Use .toUpperCase() method
    return text._____();
}

console.log(shout("hello"));  // HELLO

// Function 2: Count characters
function countLetters(text) {
    // TODO: Use .length property
    return text._____;
}

console.log(countLetters("Lahore"));  // 6
```

Exercise 4: Math Utilities

Problem: Create calculator functions

```
// Function 1: Add two numbers
function add(a, b) {
    return _____;
}

// Function 2: Find maximum of two numbers
function findMax(a, b) {
    if (a _____ b) {
        return a;
    } else {
        return b;
    }
}
```

```
// Function 3: Calculate percentage
function calculatePercentage(obtained, total) {
    return (obtained _____ total) _____ 100;
}

// Test them
console.log(add(10, 5));              // 15
console.log(findMax(20, 15));         // 20
console.log(calculatePercentage(85, 100));  // 85
```

Exercise 5: Validation Function

Problem: Check if student passed (marks >= 50)

```
function hasPassed(marks) {
    // TODO: Return true if marks >= 50, false otherwise
    if (marks _______) {
        return _______;
    } else {
        return _______;
    }
}

// Shorter way:
function hasPassedShort(marks) {
    return marks >= 50;  // Returns true or false directly!
}

// Test
console.log(hasPassed(60));   // true
console.log(hasPassed(45));   // false
```

🚀 آج کا چیلنج (Today's Challenge)

Project: Utility Functions Library

افادی فنکشنز لائبریری

The Problem: You're building a library of helper functions that developers can use in their projects. Create functions for common tasks: currency conversion, discounts, phone formatting, and email validation!

What You're Building: A collection of 4+ reusable utility functions with a menu system to test them all!

Success Criteria:

- ☐ All functions work correctly
- ☐ Each function returns a value (no just console.log)
- ☐ Handles invalid inputs gracefully
- ☐ Code is well-commented
- ☐ Easy to use and test

Phase 1: Planning (سوچیں پہلے)

Before coding, answer:

1. **What functions do I need?**
    - convertCurrency(amount, from, to)
    - calculateDiscount(price, percent)
    - formatPhoneNumber(digits)
    - validateEmail(email)
2. **What should each function do?**
    - Take input parameters
    - Process the data
    - Return result
    - Handle errors
3. **How will I test them?**
    - Call each function with test data
    - Show results in console

Planning Checkpoint:

- ☐ I understand what each function needs to do
- ☐ I know what parameters each needs
- ☐ I know what each should return
- ☐ 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>Utility Functions Library</title>
</head>
<body>
    <h1>⚙️ Utility Functions Library</h1>
    <h2>افادی فنکشنز لائبریری</h2>
    <h3>Press F12 to see all functions in action!</h3>

    <script>
        // ==================================
        // UTILITY FUNCTIONS LIBRARY
        // By: [Your Name]
        // Date: [Today's Date]
        // ==================================

        console.log("⚙️ افادی فنکشنز لائبریری");
        console.log("==============================\n");

        // ========== FUNCTION 1: Currency Converter ==========

        /**
         * Converts currency between PKR, USD, EUR
         * @param {number} amount - Amount to convert
         * @param {string} fromCurrency - Source currency (PKR/USD/EUR)
         * @param {string} toCurrency - Target currency (PKR/USD/EUR)
         * @returns {number} Converted amount
         */
        function convertCurrency(amount, fromCurrency, toCurrency) {
            // TODO: Define exchange rates (approximate)
            const rates = {
                PKR: 1,
                USD: 278,    // 1 USD = 278 PKR
                EUR: 305     // 1 EUR = 305 PKR
            };

            // TODO Step 1: Convert to PKR first
            const inPKR = amount _____ rates[fromCurrency];

            // TODO Step 2: Convert from PKR to target currency
```

```
    const result = inPKR _____ rates[toCurrency];

    // TODO Step 3: Return result (round to 2 decimals)
    return Math.round(result * 100) / 100;
}

// ========== FUNCTION 2: Discount Calculator ==========

/**
 * Calculates discounted price
 * @param {number} price - Original price
 * @param {number} discountPercent - Discount percentage
 * @returns {number} Final price after discount
 */
function calculateDiscount(price, discountPercent) {
    // TODO: Validate inputs
    if (price < 0 || discountPercent < 0 || discountPercent > 100) {
        return "Invalid input!";
    }

    // TODO: Calculate discount amount
    const discount = price _____ (discountPercent / 100);

    // TODO: Calculate final price
    const finalPrice = price _____ discount;

    // TODO: Return final price
    return Math.round(finalPrice * 100) / 100;
}

// ========== FUNCTION 3: Phone Number Formatter ==========

/**
 * Formats Pakistani phone number to +92-XXX-XXXXXXX
 * @param {string} digits - 11 digit number (03001234567)
 * @returns {string} Formatted number
 */
function formatPhoneNumber(digits) {
    // TODO: Remove any spaces or dashes
    const cleaned = digits.replace(/[-\s]/g, "");

    // TODO: Check if 11 digits
    if (cleaned.length _______ 11) {
        return "Error: Must be 11 digits";
    }

    // TODO: Check if starts with 03
    if (!cleaned.startsWith("03")) {
```

```
        return "Error: Must start with 03";
    }

    // TODO: Format as +92-XXX-XXXXXXX
    // HINT: Remove leading 0, add +92
    const formatted = "+92-" + cleaned.slice(1, 4) + "-" + cleaned.slice(4);

    return formatted;
}

// ========== FUNCTION 4: Email Validator ==========

/**
 * Validates email format (basic check)
 * @param {string} email - Email address to validate
 * @returns {boolean} true if valid, false otherwise
 */
function validateEmail(email) {
    // TODO: Check if email is not empty
    if (email.length === 0) {
        return false;
    }

    // TODO: Check if contains @ symbol
    if (!email.includes("@")) {
        return false;
    }

    // TODO: Check if @ is not first or last character
    const atPosition = email.indexOf("@");
    if (atPosition === 0 || atPosition === email.length - 1) {
        return false;
    }

    // TODO: Check if contains . after @
    const domain = email.slice(atPosition);
    if (!domain.includes(".")) {
        return false;
    }

    // All checks passed!
    return true;
}

// ========== TESTING THE FUNCTIONS ==========

console.log("===== FUNCTION TESTS =====\n");
```

```
        // Test 1: Currency Converter
        console.log("1️⃣ CURRENCY CONVERTER");
        console.log("100 USD to PKR:", convertCurrency(100, "USD", "PKR"));
        console.log("5000 PKR to USD:", convertCurrency(5000, "PKR", "USD"));
        console.log("50 EUR to PKR:", convertCurrency(50, "EUR", "PKR"));
        console.log("");

        // Test 2: Discount Calculator
        console.log("2️⃣ DISCOUNT CALCULATOR");
        console.log("Rs. 1000 with 10% off:", calculateDiscount(1000, 10));
        console.log("Rs. 5000 with 25% off:", calculateDiscount(5000, 25));
        console.log("Rs. 750 with 15% off:", calculateDiscount(750, 15));
        console.log("");

        // Test 3: Phone Formatter
        console.log("3️⃣ PHONE NUMBER FORMATTER");
        console.log("03001234567 →", formatPhoneNumber("03001234567"));
        console.log("0321-9876543 →", formatPhoneNumber("0321-9876543"));
        console.log("0300 123 4567 →", formatPhoneNumber("0300 123 4567"));
        console.log("");

        // Test 4: Email Validator
        console.log("4️⃣ EMAIL VALIDATOR");
        console.log("ali@gmail.com:", validateEmail("ali@gmail.com"));
        console.log("invalid.email:", validateEmail("invalid.email"));
        console.log("@gmail.com:", validateEmail("@gmail.com"));
        console.log("test@:", validateEmail("test@"));
        console.log("");

        console.log("==============================");
        console.log("✅ All functions tested!");

        // TODO BONUS: Create a menu system
        // Allow user to choose which function to use

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

Phase 3: Milestones (سنگ میل)

Milestone 1: Currency Converter Works ✅

- ☐ Converts USD to PKR correctly
- ☐ Converts PKR to USD correctly
- ☐ Handles all three currencies

- Test: 100 USD should give ~27,800 PKR

Milestone 2: Discount Calculator Works ✅

- ☐ Calculates discount correctly
- ☐ Returns final price
- ☐ Handles invalid inputs
- Test: Rs. 1000 with 10% off = Rs. 900

Milestone 3: Phone Formatter Works ✅

- ☐ Formats 11-digit numbers correctly
- ☐ Validates number format
- ☐ Handles different input formats
- Test: "03001234567" → "+92-300-1234567"

Milestone 4: Email Validator Works ✅

- ☐ Returns true for valid emails
- ☐ Returns false for invalid emails
- ☐ Checks all validation rules
- Test: "ali@gmail.com" → true

Debugging Guide (اگر پھنس جائیں)

Problem: Function returns undefined

- ☐ Check: Did you use `return` keyword?
- ☐ Check: Is return statement inside the function?
- ☐ Check: Are you spelling variable names correctly?

Problem: Wrong calculations

- ☐ Add console.log inside function to see values
- ☐ Check: Are you using correct operators (* / + -)?
- ☐ Check: Are numbers not strings? (no quotes!)

Problem: String methods not working

- ☐ Check: Is variable actually a string?
- ☐ Check: Are you calling method correctly? (.includes, .slice, etc.)
- ☐ Check: Are you returning the result?

Common Logic Issues:

```
// ❌ WRONG: Forgot to return
function add(a, b) {
    a + b;  // Missing return!
}

// ✅ RIGHT:
function add(a, b) {
    return a + b;
}
```

Extension Challenges (بونس چیلنج)

If you finish early:

🌟 Level 1: Add More Utilities

```
// Calculate CGPA from marks
function calculateCGPA(marks) {
    // marks is array: [85, 90, 78, 92]
    // return average / 25 (assuming 100 marks = 4.0 GPA)
}

// Check if leap year
function isLeapYear(year) {
    // return true if leap year
}

// Generate random number between min and max
function randomBetween(min, max) {
    return Math.floor(Math.random() * (max - min + 1)) + min;
}
```

🌟🌟 Level 2: Error Handling

```
// Add try-catch for error handling
function safeConvert(amount, from, to) {
    try {
        return convertCurrency(amount, from, to);
    } catch (error) {
        return "Error: " + error.message;
    }
}
```

🌟🌟🌟 Level 3: Interactive Menu

```
// Create menu to choose function
function showMenu() {
    console.log("Choose a function:");
    console.log("1. Convert Currency");
    console.log("2. Calculate Discount");
    console.log("3. Format Phone");
    console.log("4. Validate Email");
}
```

📝 Daily Quiz (5 منٹ کا ٹیسٹ)

Instructions: Answer WITHOUT looking at notes!

1. How do you declare a function?

- A) `function = myFunction() {}`
- B) `function myFunction() {}`
- C) `myFunction() = function {}`
- D) `def myFunction() {}`

▶ See Answer (Try first!)

Answer: B - `function myFunction() {}` . Use the `function` keyword, then name, then parentheses, then curly braces with code inside.

---

2. What's the difference between parameters and arguments?

- A) They're the same thing
- B) Parameters are in definition, arguments are in call
- C) Parameters are optional, arguments are required
- D) Parameters are numbers, arguments are strings

▶ See Answer (Try first!)

Answer: B - Parameters are placeholders in function definition (خالی ڈبے). Arguments are actual values when calling function (بھرے ہوئے ڈبے). Example: `function greet(name)` - name is parameter. `greet("Ali")` - "Ali" is argument.

---

3. What does this function return?

```
function calculate(x) {
    const result = x * 2;
    console.log(result);
}
const answer = calculate(5);
```

- A) 10
- B) 5
- C) undefined
- D) Error

▶ See Answer (Try first!)

Answer: C - undefined. The function console.logs 10, but doesn't RETURN anything. Without `return` keyword, functions return undefined. To fix: `return result;`

---

4. What is local scope?

- A) Variables accessible everywhere
- B) Variables only inside a function
- C) Variables declared with var
- D) Global variables

▶ See Answer (Try first!)

Answer: B - Local scope means variables are only accessible inside the function where they're declared. Like items in a room - only accessible in that room (کمرے میں), not in the whole house.

---

5. What's the output?

```
function test() {
    return "Hello";
    return "World";
}
console.log(test());
```

- A) Hello World
- B) World
- C) Hello
- D) undefined

▶ See Answer (Try first!)

Answer: C - "Hello". Once return runs, function stops immediately. The second return never executes. Code after return is unreachable!

Scoring:

- **5/5:** 🎉 Function Master! You understand functions completely!
- **4/5:** 👏 Great! Review the one you missed
- **3/5:** 👍 Good! Read through concepts again
-  **/5:** 💪 Practice more with functions

---

🎓 Today's Homework (گھر کا کام)

Required (لازمی):

- ☐ Complete the Utility Functions Library
- ☐ Test all 4 functions with different inputs
- ☐ Explain to a family member: "What is a function?" in Urdu

Optional (اختیاری):

- ☐ Try the extension challenges
- ☐ Create 3 more utility functions of your choice
- ☐ Build a "Calculator" with add, subtract, multiply, divide functions
- ☐ Make a function that takes array and returns average

For Tomorrow:

- ☐ Think about: "How would I store related functions and data together?"
- ☐ This will help with tomorrow's topic: Arrays & Array Methods!

💭 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 **Arrays & Array Methods** where you'll build a **Daraz Product Manager**!

You'll learn how to:

- Store multiple items in one variable (arrays)
- Loop through arrays
- Use powerful array methods (map, filter, forEach)
- Manipulate lists of data

Get Ready By:

- ☐ Making sure all your functions work
- ☐ Thinking: How would you store 100 student names?
- ☐ Arrays are the answer!

📚 Resources (اگر مزید پڑھنا ہو)

Free Resources (3G-Friendly):

📖 MDN - Functions

- Link: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Functions
- Best for: Deep understanding of functions

📹 JavaScript Functions in Urdu

- Search: "JavaScript functions Urdu tutorial parameters return"
- Best for: Visual learners

💻 Practice Functions

- Create utility functions for daily tasks
- Practice with parameters and return

CodeSensei's Tip of the Day: 💡

"Functions are like recipes - write once, use many times. The best functions do ONE thing well. Don't try to make a function that does everything! Make small, focused functions, then combine them. That's how professional developers code."

"فنکشنز ریسیپیوں کی طرح ہیں - ایک بار لکھو، بار بار استعمال کرو۔ بہترین فنکشن ایک کام اچھی طرح کرتا ہے۔"

👥 Team Activity (Tomorrow's Standup)

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

1. Your favorite utility function you created
2. One thing you learned about return statements
3. Explain scope using Urdu analogy
4. One question about functions

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 4 Complete! See you tomorrow for Arrays! 🚀

اللہ حافظ! Tomorrow we learn to handle lists of data! 📊