PHP & Laravel Backend Engineering

Week 2 – Functions, Arrays & Forms

Program: PHP & Laravel Backend Engineering
Week: 2
Level: Beginner
Duration: 3 Days

□ Learning Objectives

By the end of this week, you will be able to:

- □ Create and use your own functions
- □ Work with different types of arrays
- □ Process forms and user input
- □ Keep your forms safe and secure
- □ Build interactive web applications

Day 1 – Understanding PHP Functions

□ What are Functions?

A **function** is a reusable block of code that performs a specific task.

Real-world analogy:
Think of functions like kitchen appliances. Each one does a specific job:

- A toaster makes toast
- A blender mixes things
- A microwave heats food

In the same way, functions in PHP do specific jobs in your code.

□ Basic Function Syntax

```php
<?php
// Step 1: Define the function
function sayHello($name) {
    return "Hello, " . $name . "!";
}

// Step 2: Call (use) the function
echo sayHello("Ali");  // Output: Hello, Ali!
echo sayHello("Sara"); // Output: Hello, Sara!
?>
```

Explanation:

1. `function` - Keyword to create a function
2. `sayHello` - Name of the function (you choose this)
3. `($name)` - Parameter (input) the function receives
4. `return` - Sends back a result
5. `sayHello("Ali")` - Calling the function with an argument

□ Types of Functions

1. Simple Function (No Parameters)

```php
<?php
function greet() {
    return "Welcome to our website!";
}

echo greet();  // Output: Welcome to our website!
?>
```

2. Function with Parameters

```php
<?php
function addNumbers($num1, $num2) {
    $sum = $num1 + $num2;
    return $sum;
}

echo addNumbers(5, 3);   // Output: 8
echo addNumbers(10, 20); // Output: 30
?>
```

3. Function with Default Values

```php
<?php
function greet($name = "Guest") {
    return "Welcome, " . $name;
}

echo greet();         // Output: Welcome, Guest
echo greet("Sara");   // Output: Welcome, Sara
?>
```

Explanation:
If you don't provide a value for `$name`, it uses "Guest" as the default.

4. Function with Multiple Parameters

```php
<?php
function calculateTotal($price, $quantity, $tax = 0.15) {
    $subtotal = $price * $quantity;
    $taxAmount = $subtotal * $tax;
    $total = $subtotal + $taxAmount;
    return $total;
}

echo calculateTotal(100, 2);      // Output: 230 (with 15% tax)
echo calculateTotal(100, 2, 0.10); // Output: 220 (with 10% tax)
?>
```

▯ Variable Scope

Scope means where a variable can be used.

```php
<?php
// Global variable (can be used anywhere outside functions)
$globalVar = "I am global";

function testScope() {
    // Local variable (only works inside this function)
    $localVar = "I am local";

    // To use global variable inside function, use 'global' keyword
    global $globalVar;
    echo $globalVar;  // Works!

    echo $localVar;   // Works!
}

testScope();
echo $localVar;  // ERROR! $localVar doesn't exist here
?>
```

󰀀 Why Use Functions?

Benefit	Explanation	Example
Reusability	Write once, use many times	Calculate tax for multiple products
Organization	Break big problems into small pieces	Separate login, registration, validation
Maintainability	Fix in one place, fixes everywhere	Update tax calculation in one function
Readability	Makes code easier to understand	`sendEmail()` is clearer than 50 lines of code

󰀀 Practice Exercises

1. **Good Morning Function**

```php
// Create a function that says "Good morning" with a name
function goodMorning($name) {
    return "Good morning, " . $name . "!";
}
```

2. **Circle Area Calculator**

```php
// Create a function to calculate the area of a circle
function circleArea($radius) {
    $pi = 3.14159;
    return $pi * $radius * $radius;
}
```

3. **Even or Odd Checker**

```php
// Write a function that checks if a number is even or odd
function checkEvenOdd($number) {
    if ($number % 2 == 0) {
        return "Even";
    } else {
        return "Odd";
    }
}
```

4. **Temperature Converter**

```php
// Make a function to convert Celsius to Fahrenheit
function celsiusToFahrenheit($celsius) {
    return ($celsius * 9/5) + 32;
}
```

Day 2 – Working with Arrays and Forms

What are Arrays?

An **array** is a special variable that can store multiple values in a single variable.

Real-world analogy:
Arrays are like shopping lists that can hold many items in one variable.

```
<?php
// Instead of this:
$color1 = "Red";
$color2 = "Green";
$color3 = "Blue";

// We can use this:
$colors = ["Red", "Green", "Blue"];
?>
```

Types of Arrays

1. Indexed Array (Numbered List)

Arrays with numeric index (starting from 0).

```
<?php
// Create an indexed array
$colors = ["Red", "Green", "Blue"];

// Access array elements
echo $colors[0];  // Output: Red
echo $colors[1];  // Output: Green
echo $colors[2];  // Output: Blue

// Add new element
$colors[3] = "Yellow";
?>
```

Explanation:

- Index starts at 0 (not 1!)
- `$colors[0]` is the first element
- `$colors[1]` is the second element

2. Associative Array (Labeled List)

Arrays with named keys instead of numbers.

```
<?php
// Create an associative array
$student = [
    "name" => "Ali",
    "age" => 20,
    "grade" => "A",
    "email" => "ali@example.com"
];

// Access array elements
echo $student["name"];   // Output: Ali
echo $student["age"];    // Output: 20
echo $student["grade"];  // Output: A
?>
```

Explanation:

- Use meaningful keys like "name", "age"
- Use `=>` to connect key and value
- Access using `$array["key"]`

3. Multidimensional Array (Array inside Array)

```
<?php
$students = [
    ["name" => "Ali", "age" => 20, "grade" => "A"],
    ["name" => "Sara", "age" => 19, "grade" => "B"],
    ["name" => "Omar", "age" => 21, "grade" => "A"]
];

// Access elements
echo $students[0]["name"];  // Output: Ali
echo $students[1]["age"];   // Output: 19
?>
```

▯ Common Array Functions

Function	What it does	Example	Result
`count()`	Count elements	`count($colors)`	3
`array_push()`	Add to end	`array_push($colors, "Yellow")`	Adds "Yellow"
`array_pop()`	Remove last	`array_pop($colors)`	Removes last item
`in_array()`	Check if exists	`in_array("Red", $colors)`	true/false
`array_merge()`	Combine arrays	`array_merge($arr1, $arr2)`	Combined array
`sort()`	Sort ascending	`sort($colors)`	Sorted array
`array_reverse()`	Reverse order	`array_reverse($colors)`	Reversed array

Practical Examples:

```
<?php
$fruits = ["Apple", "Banana", "Orange"];

// Count elements
echo count($fruits);  // Output: 3

// Add element
array_push($fruits, "Mango");
print_r($fruits);  // Output: Array ( [0] => Apple [1] => Banana [2] => Orange [3] => Mango )

// Check if element exists
if (in_array("Apple", $fruits)) {
    echo "Apple is in the list!";
}

// Loop through array
foreach ($fruits as $fruit) {
    echo $fruit . "<br>";
}
?>
```

▯ Handling Forms

Forms allow users to send data to your PHP script.

Step 1: Create HTML Form (index.html)

```
<!DOCTYPE html>
<html>
<head>
    <title>Contact Form</title>
</head>
<body>
    <h2>Contact Us</h2>
    <form method="POST" action="welcome.php">
        <label>Your Name:</label>
        <input type="text" name="username" required><br><br>

        <label>Your Email:</label>
        <input type="email" name="email" required><br><br>

        <button type="submit">Send</button>
    </form>
</body>
</html>
```

Explanation:

- `method="POST"` - How data is sent (POST is secure)
- `action="welcome.php"` - Where data goes
- `name="username"` - How we access data in PHP
- `required` - Field must be filled

Step 2: Process Form Data (welcome.php)

```
<?php
// Check if form was submitted
if ($_SERVER["REQUEST_METHOD"] == "POST") {

    // Step 1: Get form data
    $name = $_POST['username'];
    $email = $_POST['email'];

    // Step 2: Clean the data (security!)
    $name = htmlspecialchars($name);
    $email = filter_var($email, FILTER_SANITIZE_EMAIL);

    // Step 3: Validate email
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
        echo "Invalid email format!";
    } else {
        // Step 4: Show welcome message
        echo "<h1>Welcome, $name!</h1>";
        echo "<p>Your email is: $email</p>";
    }
}
?>
```

Explanation:

1. `$_SERVER["REQUEST_METHOD"]` - Checks if form was submitted
2. `$_POST['username']` - Gets data from form field
3. `htmlspecialchars()` - Prevents XSS attacks
4. `filter_var()` - Validates and cleans email

□ Form Security Tips

Security Issue	Solution	Why?
XSS Attacks	Use `htmlspecialchars()`	Prevents malicious scripts
SQL Injection	Use prepared statements	Prevents database attacks

Security Issue	Solution	Why?
Email Validation	Use `filter_var()`	Ensures valid email format
Empty Fields	Check with `empty()`	Prevents empty submissions

Secure Form Processing Example:

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Initialize errors array
    $errors = [];

    // Get and clean data
    $name = trim($_POST['username'] ?? '');
    $email = trim($_POST['email'] ?? '');

    // Validate name
    if (empty($name)) {
        $errors[] = "Name is required";
    } elseif (strlen($name) < 3) {
        $errors[] = "Name must be at least 3 characters";
    }

    // Validate email
    if (empty($email)) {
        $errors[] = "Email is required";
    } elseif (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
        $errors[] = "Invalid email format";
    }

    // If no errors, process the form
    if (empty($errors)) {
        $name = htmlspecialchars($name);
        $email = htmlspecialchars($email);
        echo "Success! Welcome, $name";
    } else {
        // Show errors
        foreach ($errors as $error) {
            echo "<p style='color:red;'>$error</p>";
        }
    }
}
?>
```

Day 3 – Student Registration Project

▯ Project Overview

What we'll build:

- A complete student registration system
- Form validation (check if data is correct)
- Secure data handling
- Welcome page with student details

Skills you'll practice:

- Creating HTML forms
- Processing form data with PHP
- Validating user input
- Using sessions
- Displaying dynamic content

▯ Project Requirements

Feature	Description
Full Name	Required, letters and spaces only
Email	Required, must be valid email format
Password	Required, minimum 8 characters
Course	Dropdown selection (required)
Gender	Radio button selection
Hobbies	Multiple checkboxes (optional)

📁 Project Structure

```
/student_registration
  ├── register.html    # Registration form
  ├── process.php      # Form processing and validation
  └── welcome.php      # Success page
```

Step 1: Create the Registration Form (register.html)

```
<!DOCTYPE html>
<html>
<head>
    <title>Student Registration</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            max-width: 500px;
            margin: 50px auto;
            padding: 20px;
            background-color: #f4f4f4;
        }
        .container {
            background: white;
            padding: 30px;
            border-radius: 8px;
            box-shadow: 0 2px 10px rgba(0,0,0,0.1);
        }
        h1 {
            color: #333;
            text-align: center;
        }
        .form-group {
            margin-bottom: 15px;
        }
        label {
            display: block;
            margin-bottom: 5px;
            font-weight: bold;
            color: #555;
        }
        input[type="text"],
        input[type="email"],
        input[type="password"],
        select {
            width: 100%;
            padding: 8px;
            border: 1px solid #ddd;
            border-radius: 4px;
            box-sizing: border-box;
        }
        button {
            width: 100%;
            padding: 10px;
            background-color: #4CAF50;
            color: white;
            border: none;
            border-radius: 4px;
            cursor: pointer;
            font-size: 16px;
        }
        button:hover {
            background-color: #45a049;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>▯ Student Registration</h1>
        <form method="POST" action="process.php">

            <div class="form-group">
                <label>Full Name:</label>
                <input type="text" name="fullname" required>
            </div>

            <div class="form-group">
                <label>Email:</label>
                <input type="email" name="email" required>
            </div>

            <div class="form-group">
                <label>Password:</label>
                <input type="password" name="password" minlength="8" required>
            </div>

            <div class="form-group">
                <label>Course:</label>
                <select name="course" required>
                    <option value="">Select a course</option>
                    <option value="web">Web Development</option>
                    <option value="design">Graphic Design</option>
                    <option value="marketing">Digital Marketing</option>
                    <option value="data">Data Science</option>
                </select>
            </div>

            <div class="form-group">
                <label>Gender:</label>
                <input type="radio" name="gender" value="male" required> Male
                <input type="radio" name="gender" value="female"> Female
            </div>

            <div class="form-group">
                <label>Hobbies (select all that apply):</label>
                <input type="checkbox" name="hobbies[]" value="sports"> Sports<br>
                <input type="checkbox" name="hobbies[]" value="music"> Music<br>
                <input type="checkbox" name="hobbies[]" value="reading"> Reading<br>
                <input type="checkbox" name="hobbies[]" value="coding"> Coding
            </div>

            <button type="submit">Register Now</button>
        </form>
    </div>
```

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

Explanation of HTML Form Elements:

Element	Purpose	Example
`<input type="text">`	Single-line text input	Name, address
`<input type="email">`	Email with validation	Email address
`<input type="password">`	Hidden text input	Password
`<select>`	Dropdown list	Course selection
`<input type="radio">`	Single choice	Gender
`<input type="checkbox">`	Multiple choices	Hobbies
`name="hobbies[]"`	Array of values	Multiple selections

Step 2: Process the Form (process.php)

This file validates the form data and handles errors.

```
<?php
// Step 1: Start the session to store data
session_start();

// Step 2: Create an array to store validation errors
$errors = [];

// Step 3: Check if form was submitted using POST method
if ($_SERVER["REQUEST_METHOD"] == "POST") {

    // Step 4: Get form data and clean it
    // The ?? '' means: if value doesn't exist, use empty string
    $fullname = cleanInput($_POST['fullname'] ?? '');
    $email = cleanInput($_POST['email'] ?? '');
    $password = $_POST['password'] ?? '';
    $course = cleanInput($_POST['course'] ?? '');
    $gender = cleanInput($_POST['gender'] ?? '');
    $hobbies = $_POST['hobbies'] ?? [];  // Array of hobbies

    // Step 5: Validate Full Name
    if (empty($fullname)) {
        $errors[] = "Full name is required";
    } elseif (strlen($fullname) < 3) {
        $errors[] = "Name must be at least 3 characters";
    } elseif (!preg_match("/^[a-zA-Z ]*$/", $fullname)) {
        $errors[] = "Only letters and spaces allowed in name";
    }

    // Step 6: Validate Email
    if (empty($email)) {
        $errors[] = "Email is required";
    } elseif (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
        $errors[] = "Invalid email format";
    }

    // Step 7: Validate Password
    if (empty($password)) {
        $errors[] = "Password is required";
    } elseif (strlen($password) < 8) {
        $errors[] = "Password must be at least 8 characters";
    }

    // Step 8: Validate Course
    if (empty($course)) {
        $errors[] = "Please select a course";
    }

    // Step 9: Validate Gender
    if (empty($gender)) {
        $errors[] = "Please select gender";
    }

    // Step 10: If no errors, process the form
    if (empty($errors)) {
        // Hash the password for security
        $hashedPassword = password_hash($password, PASSWORD_DEFAULT);

        // Create student data array
        $studentData = [
            'fullname' => $fullname,
            'email' => $email,
            'password' => $hashedPassword,  // Store hashed password
            'course' => $course,
            'gender' => $gender,
            'hobbies' => $hobbies,
            'registration_date' => date('Y-m-d H:i:s')
        ];

        // Store in session (in real app, save to database)
        $_SESSION['student'] = $studentData;

        // Redirect to welcome page
        header('Location: welcome.php');
        exit;
    }
}

// Function to clean and sanitize input data
function cleanInput($data) {
    $data = trim($data);              // Remove spaces from beginning and end
    $data = stripslashes($data);      // Remove backslashes
    $data = htmlspecialchars($data);  // Convert special characters to HTML entities
    return $data;
}
?>

<!DOCTYPE html>
<html>
<head>
    <title>Registration Error</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            max-width: 500px;
            margin: 50px auto;
            padding: 20px;
        }
        .error-container {
            background-color: #f8d7da;
            border: 1px solid #f5c6cb;
            color: #721c24;
            padding: 20px;
            border-radius: 5px;
        }
        .error-container h3 {
            margin-top: 0;
        }
        .error-container ul {
            margin-bottom: 0;
        }
        .back-link {
            display: inline-block;
            margin-top: 15px;
            color: #007bff;
            text-decoration: none;
        }
    </style>
</head>
<body>
    <!-- Show errors if any -->
    <?php if (!empty($errors)): ?>
        <div class="error-container">
            <h3>⚠ Please fix these errors:</h3>
            <ul>
                <?php foreach ($errors as $error): ?>
                    <li><?php echo $error; ?></li>
                <?php endforeach; ?>
            </ul>
            <a href="register.html" class="back-link">← Go back to form</a>
        </div>
    <?php endif; ?>
</body>
</html>
```

Explanation of Validation Steps:

Step	What it does	Why it's important
`trim()`	Removes extra spaces	Prevents " John " from being different than "John"
`stripslashes()`	Removes backslashes	Prevents issues with quotes
`htmlspecialchars()`	Converts special characters	Prevents XSS attacks
`filter_var()`	Validates email	Ensures proper email format
`preg_match()`	Checks pattern	Ensures only letters in name
`password_hash()`	Encrypts password	Never store plain passwords!

Step 3: Welcome Page (welcome.php)

This page displays the registration success message and student details.

```
<?php
// Step 1: Start the session
session_start();

// Step 2: Check if student data exists in session
// If not, redirect back to registration form
if (!isset($_SESSION['student'])) {
    header('Location: register.html');
    exit;
}

// Step 3: Get student data from session
$student = $_SESSION['student'];
?>
```

```
<!DOCTYPE html>
<html>
<head>
    <title>Welcome!</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            max-width: 600px;
            margin: 50px auto;
            padding: 20px;
            background-color: #f4f4f4;
        }
        .container {
            background: white;
            padding: 30px;
            border-radius: 8px;
            box-shadow: 0 2px 10px rgba(0,0,0,0.1);
        }
        h1 {
            color: #4CAF50;
            text-align: center;
        }
        .success-message {
            background-color: #d4edda;
            border: 1px solid #c3e6cb;
            color: #155724;
            padding: 15px;
            border-radius: 5px;
            margin-bottom: 20px;
        }
        .detail-box {
            background-color: #f8f9fa;
            padding: 15px;
            border-radius: 5px;
            margin-top: 20px;
        }
        .detail-box p {
            margin: 10px 0;
        }
        .detail-box strong {
            color: #333;
        }
        .btn {
            display: inline-block;
            background-color: #4CAF50;
            color: white;
            padding: 10px 20px;
            text-decoration: none;
            border-radius: 4px;
            margin-top: 20px;
        }
        .btn:hover {
            background-color: #45a049;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>□ Registration Successful!</h1>

        <div class="success-message">
            <p><strong>Thank you for registering, <?php echo htmlspecialchars($student['fullname']); ?>!</strong></p>
            <p>We've sent a confirmation email to <?php echo htmlspecialchars($student['email']); ?></p>
        </div>
        <h2>▯ Your Registration Details:</h2>
        <div class="detail-box">
            <p><strong>Full Name:</strong> <?php echo htmlspecialchars($student['fullname']); ?></p>
            <p><strong>Email:</strong> <?php echo htmlspecialchars($student['email']); ?></p>
            <p><strong>Course:</strong> <?php echo htmlspecialchars(ucfirst($student['course'])); ?></p>
            <p><strong>Gender:</strong> <?php echo ucfirst($student['gender']); ?></p>

            <?php if (!empty($student['hobbies'])): ?>
                <p><strong>Hobbies:</strong>
                    <?php
                    // Convert each hobby to title case and join with commas
                    $hobbies = array_map('ucfirst', $student['hobbies']);
                    echo implode(', ', $hobbies);
                    ?>
                </p>
            <?php else: ?>
                <p><strong>Hobbies:</strong> None selected</p>
            <?php endif; ?>

            <p><strong>Registration Date:</strong> <?php echo $student['registration_date']; ?></p>
        </div>

        <a href="register.html" class="btn">Register Another Student</a>
    </div>
</body>
</html>
```

Explanation of Key PHP Functions:

Function	Purpose	Example
`htmlspecialchars()`	Prevents XSS attacks	Converts `<script>` to safe text
`ucfirst()`	Capitalizes first letter	"web" becomes "Web"
`array_map()`	Applies function to array	Capitalize all hobbies
`implode()`	Joins array with separator	["Sports", "Music"] becomes "Sports, Music"
`isset()`	Checks if variable exists	Prevents errors

▯ Practice Exercises

1. **Add Phone Validation**

```
 // Add this validation in process.php
if (!preg_match("/^[0-9]{10}$/", $phone)) {
    $errors[] = "Phone must be 10 digits";
}
```

2. **Password Strength Checker**

```
function checkPasswordStrength($password) {
    if (strlen($password) < 8) return "Weak";
    if (preg_match("/[A-Z]/", $password) &&
        preg_match("/[0-9]/", $password)) {
        return "Strong";
    }
    return "Medium";
}
```

3. **Remember Form Values**

```
 // In register.html, add this to keep values after error:
value="<?php echo isset($_POST['fullname']) ? htmlspecialchars($_POST['fullname']) : ''; ?>"
```

4. **Add Confirmation Password**

```php
// Add this validation
if ($password !== $confirm_password) {
    $errors[] = "Passwords do not match";
}
```

▯ Resources

- ▯ PHP Form Handling
- ▯ PHP Security Best Practices
- ▯ PHP Array Functions
- ▯ W3Schools PHP Forms
- ▯ OWASP Input Validation

▯ Tips for Success

- ▯ **Test frequently** - Test your code after each step
- ▯ **Debug smartly** - Use `var_dump($_POST)` to see form data
- ▯ **Comment your code** - Explain what each section does
- ▯ **Use browser tools** - Check for errors in developer console
- ▯ **Style matters** - Make it look professional with CSS
- ▯ **Security first** - Always validate and sanitize input
- ▯ **Read documentation** - PHP manual is your best friend

▯ Week 2 Summary

What you learned:

- ▯ Creating and using functions
- ▯ Working with indexed and associative arrays
- ▯ Building and processing HTML forms
- ▯ Validating and sanitizing user input
- ▯ Implementing security best practices
- ▯ Using sessions to store data

Next week: We'll learn about Sessions, Cookies, and Authentication!