

Unit 4: Server-Side Scripting with PHP

Introduction: Starting to Script on the Server Side

PHP (Hypertext Preprocessor) is a widely used open-source general-purpose scripting language that is especially suited for web development and can be embedded into HTML.

Client-Side vs. Server-Side:

- **Client-Side (e.g., HTML, JavaScript):** Scripts run on the user's computer (browser). The user can view the source code.
- **Server-Side (e.g., PHP, JSP, ASP):** Scripts run on the web server. The server processes the code and sends plain HTML to the browser. The user cannot see the underlying PHP code.

How PHP Works:

1. A browser requests a `.php` page.
 2. The web server (like Apache) processes the PHP instructions.
 3. The server sends the resulting output (HTML/Text) back to the browser.
-

1. PHP Basics

A PHP script can be placed anywhere in the document. A PHP file normally contains HTML tags and some PHP scripting code.

Syntax:

PHP code blocks start with `<?php` and end with `?>`.

```
<?php
    echo "Hello World!";
?>
```

Comments:

```
// Single-line comment
# Single-line comment
/* Multi-line
   comment block */
```

Output Statements:

- **echo:** Can output one or more strings. (Faster, has no return value).
 - **print:** Can only output one string, and returns 1 (slower).
-

2. Variables, Data Types, and Constants

Variables

Variables are “containers” for storing information.

Rules:

1. Starts with a \$ sign (e.g., `$count`).
2. Must start with a letter or underscore (`_`). Cannot start with a number.
3. Case-sensitive (`$age` and `$AGE` are different).

Scope:

- **Local:** Declared inside a function (only accessible there).
- **Global:** Declared outside functions (accessible everywhere except inside functions unless `global` keyword is used).
- **Static:** Local variable that retains its value between function calls.

Example:

```
<?php
$txt = "Hello World";
$x = 5;
$y = 10.5;
?>
```

Data Types

PHP is a loosely typed language (no need to declare types).

1. **String:** Text enclosed in quotes (`"Hello"`).
2. **Integer:** Whole numbers (`10`, `-5`).
3. **Float (Double):** Numbers with decimals (`10.5`).
4. **Boolean:** True/False values (`true`, `false`).
5. **Array:** Stores multiple values in one variable.
6. **Object:** Stores data and processing methods.
7. **NULL:** Variable with no value.
8. **Resource:** Reference to an external resource (e.g., database link).

Example:

```
<?php
$string = "Hello World";
```

```
$integer = 42;
$float = 3.14;
$boolean = true;
$null_var = null;
?>
```

Constants

Values that cannot be changed once defined.

Using `define()`:

```
<?php
define("GREETING", "Welcome to PHP!");
echo GREETING;
?>
```

Using `const` keyword:

```
<?php
const SITE_URL = "www.example.com";
echo SITE_URL;
?>
```

3. Operators and Expressions

Expressions

An expression is anything that has a value.

- Example: `$x = 5;` (assigns 5 to `$x`)
- Example: `$z = $x + $y;` (evaluates to sum)

Operators

Arithmetic Operators

Operator	Name	Example
+	Addition	<code>\$x + \$y</code>
-	Subtraction	<code>\$x - \$y</code>
*	Multiplication	<code>\$x * \$y</code>
/	Division	<code>\$x / \$y</code>
%	Modulus	<code>\$x % \$y</code>
**	Exponentiation	<code>\$x ** \$y</code>

Example:

```
<?php
$x = 10;
$y = 3;
echo $x + $y; // 13
echo $x % $y; // 1
?>
```

Assignment Operators

Operator	Same as
=	<code>\$x = 5;</code>
+=	<code>\$x = \$x + 5;</code>
-=	<code>\$x = \$x - 5;</code>
*=	<code>\$x = \$x * 5;</code>
/=	<code>\$x = \$x / 5;</code>

Comparison Operators

Operator	Name
==	Equal (value only)
===	Identical (value and type)
!= or <>	Not equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal

Example:

```
<?php
$x = 5;
$y = "5";
echo $x == $y; // true (value is same)
echo $x === $y; // false (different types)
?>
```

Increment/Decrement Operators

```
<?php
$x = 10;
echo ++$x; // Pre-increment: outputs 11, then returns
echo $x++; // Post-increment: outputs 11, then increments
echo $x;    // outputs 12
?>
```

Logical Operators

Operator	Name	Example
&& or and	AND	\$x && \$y (true if both true)
\\ \\ or or	OR	\$x \\ \\ \$y (true if either true)
! or not	NOT	!\$x (true if false)

String Operator

```
<?php
$str1 = "Hello";
$str2 = "World";
echo $str1 . " " . $str2; // Outputs: Hello World
?>
```

4. Decisions and Loops

Making Decisions (Conditionals)

1. If Statement

```
<?php
if ($age > 18) {
    echo "Adult";
}
?>
```

2. If...Else Statement

```
<?php
if ($mark >= 50) {
    echo "Pass";
} else {
    echo "Fail";
}
?>
```

3. Elseif Statement

```
<?php
$t = date("H");

if ($t < 10) {
    echo "Good morning!";
} elseif ($t < 20) {
```

```

        echo "Good day!";
    } else {
        echo "Good night!";
    }
?>

```

4. Switch Statement

```

<?php
$color = "red";

switch ($color) {
    case "red":
        echo "Red selected";
        break;
    case "blue":
        echo "Blue selected";
        break;
    case "green":
        echo "Green selected";
        break;
    default:
        echo "No color selected";
}
?>

```

Looping Structures

1. **While Loop** Loops through code as long as condition is true.

```

<?php
$x = 1;
while($x <= 5) {
    echo "Number: $x <br>";
    $x++;
}
?>

```

2. **Do...While Loop** Executes block once, then repeats if condition is true.

```

<?php
$x = 1;
do {
    echo "Number: $x <br>";
    $x++;
} while ($x <= 5);
?>

```

3. For Loop Loop a specified number of times.

```
<?php
// for (init; condition; increment)
for ($i = 0; $i < 5; $i++) {
    echo "Number: $i <br>";
}
?>
```

4. Foreach Loop Loops through arrays specifically.

```
<?php
$colors = array("red", "green", "blue");
foreach ($colors as $value) {
    echo "$value <br>";
}
?>
```

5. Strings

Creating Strings

Single Quotes (') Literal string. Variables are NOT expanded.

```
<?php
$name = "John";
echo 'Hello $name'; // Outputs: Hello $name
?>
```

Double Quotes (") Variables ARE replaced by their values.

```
<?php
$name = "John";
echo "Hello $name"; // Outputs: Hello John
?>
```

Accessing Strings

Characters in a string can be accessed using array index notation (zero-indexed).

```
<?php
$str = "Hello";
echo $str[0]; // Outputs: H
echo $str[1]; // Outputs: e
echo $str[4]; // Outputs: o
?>
```

Searching Strings

strlen() - Get String Length

```
<?php
echo strlen("Hello World"); // Outputs: 11
?>
```

strpos() - Find Text Within a String Returns the position of the first occurrence. Returns false if not found.

```
<?php
echo strpos("Hello World", "World"); // Outputs: 6
echo strpos("Hello World", "xyz"); // Outputs: (nothing - false)
?>
```

str_word_count() - Count Words

```
<?php
echo str_word_count("Hello World"); // Outputs: 2
?>
```

Replacing Strings

str_replace() - Replace Text Replaces all occurrences of the search string with the replacement string.

```
<?php
echo str_replace("World", "Peter", "Hello World!");
// Outputs: Hello Peter!
?>
```

str_ireplace() - Case-Insensitive Replace

```
<?php
echo str_ireplace("WORLD", "Peter", "Hello World!");
// Outputs: Hello Peter!
?>
```

Formatting Strings

strtoupper() - Convert to Uppercase

```
<?php
echo strtoupper("hello"); // Outputs: HELLO
?>
```

strtolower() - Convert to Lowercase

```
<?php
echo strtolower("HELLO"); // Outputs: hello
?>
```

ucfirst() - Uppercase First Character

```
<?php
echo ucfirst("hello"); // Outputs: Hello
?>
```

ucwords() - Uppercase Word Start

```
<?php
echo ucwords("hello world"); // Outputs: Hello World
?>
```

trim() - Remove Whitespace Removes whitespace from both start and end.

```
<?php
$str = " Hello World ";
echo trim($str); // Outputs: Hello World
?>
```

ltrim() / rtrim() - Remove from Left/Right

```
<?php
$str = " Hello ";
echo ltrim($str); // Removes leading spaces
echo rtrim($str); // Removes trailing spaces
?>
```

substr() - Extract Part of String

```
<?php
echo substr("Hello World", 6); // Outputs: World
echo substr("Hello World", 0, 5); // Outputs: Hello
?>
```

6. Arrays

An array stores multiple values in one single variable.

Types of Arrays

1. Indexed Arrays Numeric index (starts at 0).

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo $cars[0]; // Outputs: Volvo
?>
```

Short Syntax (PHP 5.4+)

```
<?php
$cars = ["Volvo", "BMW", "Toyota"];
echo $cars[0]; // Outputs: Volvo
?>
```

Accessing and Modifying

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
$cars[0] = "Honda"; // Modify value
echo count($cars); // Outputs: 3 (array length)
?>
```

2. Associative Arrays Named keys instead of numeric indices.

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
// Outputs: Peter is 35 years old.
?>
```

Modifying Associative Arrays

```
<?php
$age['Peter'] = "36"; // Modify value
$age['John'] = "50"; // Add new element
?>
```

Looping Through Associative Arrays

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37");
foreach($age as $key => $value) {
    echo "Key: " . $key . ", Value: " . $value . "<br>";
}
?>
```

3. Multidimensional Arrays

Arrays containing one or more arrays.

Two-Dimensional Arrays

```
<?php
$cars = array (
    array("Volvo", 22, 18),
    array("BMW", 15, 13),
    array("Saab", 5, 2)
);

echo $cars[0][0]; // Outputs: Volvo
echo $cars[0][1]; // Outputs: 22
echo $cars[1][2]; // Outputs: 13
?>
```

Associative Multidimensional Arrays

```
<?php
$contacts = array(
    array("Name" => "Peter", "Email" => "peter@test.com", "Age" => 35),
    array("Name" => "Ben", "Email" => "ben@test.com", "Age" => 37),
    array("Name" => "Joe", "Email" => "joe@test.com", "Age" => 43)
);

echo $contacts[0]["Email"]; // Outputs: peter@test.com
echo $contacts[1]["Name"]; // Outputs: Ben
?>
```

Looping Multidimensional Arrays

```
<?php
$cars = array (
    array("Volvo", 22),
    array("BMW", 15),
    array("Saab", 5)
);

foreach ($cars as $car) {
    foreach ($car as $value) {
        echo "$value, ";
    }
}
?>
```

7. PHP with Database (MySQL)

The standard way to connect PHP to a MySQL database is using **MySQLi** (MySQL Improved) extension.

Step 1: Connect to Database

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

Step 2: Create Database and Table

```
<?php
$sql = "CREATE DATABASE myDB";
if ($conn->query($sql) === TRUE) {
    echo "Database created successfully";
} else {
    echo "Error: " . $conn->error;
}
?>
```

Create a table:

```
<?php
$sql = "CREATE TABLE MyGuests (
  id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  firstname VARCHAR(30) NOT NULL,
  lastname VARCHAR(30) NOT NULL,
  email VARCHAR(50),
  reg_date TIMESTAMP
)";

if ($conn->query($sql) === TRUE) {
    echo "Table created successfully";
} else {
```

```

    echo "Error: " . $conn->error;
}
?>

```

Step 3: Insert Data

```

<?php
$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";

if ($conn->query($sql) === TRUE) {
    echo "Record inserted successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}
?>

```

Step 4: Select Data

```

<?php
$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // Output data of each row
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";
    }
} else {
    echo "0 results";
}
?>

```

Step 5: Update Data

```

<?php
$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";

if ($conn->query($sql) === TRUE) {
    echo "Record updated successfully";
} else {
    echo "Error: " . $conn->error;
}
?>

```

Step 6: Delete Data

```
<?php
$sql = "DELETE FROM MyGuests WHERE id=3";

if ($conn->query($sql) === TRUE) {
    echo "Record deleted successfully";
} else {
    echo "Error: " . $conn->error;
}
?>
```

Step 7: Close Connection

```
<?php
$conn->close();
?>
```

Summary of Key Concepts

Topic	Key Points
Variables	Start with \$, case-sensitive, loosely typed
Data Types	String, Integer, Float, Boolean, Array, Object, NULL
Constants	Immutable, defined with <code>define()</code> or <code>const</code>
Operators	Arithmetic, Assignment, Comparison, Logical, String
Conditionals	if, else, elseif, switch
Loops	while, do...while, for, foreach
Strings	Can be indexed, searched, replaced, formatted
Arrays	Indexed, Associative, Multidimensional
Database	Connect, Create, Insert, Select, Update, Delete

End of Unit 4 Notes

Prepared for Roland Institute of Technology, Berhampur
Department of Computer Science and Engineering