Unit 05: Server Side Scripting Languages

Primitives, operations and expressions, output, control statements

PHP Primitives

Brief overview of Primitives and datatypes of PHP

PHP Primitives and Data Types

- A primitive in PHP refers to a basic data type that holds a single value and is not composed of other types.
- **These include**:
- **Integer (int) -** Whole numbers (e.g., 42, -7).
- **Float (float or double) -** Decimal numbers (e.g., 3.14, -0.5).
- **Boolean (bool) -** True/False values (true, false).
- **String (string) -** A sequence of characters (e.g., "Hello World").
- These are fundamental building blocks for more complex data types like arrays and objects.

a) Integer (int)

- Whole numbers (positive or negative).
- No decimal points.
- Example: 5, -10, 1000.
- $num = 42; // Integer primitive

b) Floating Point (float or double)

- Numbers with decimal points.
- Example: 3.14, -0.99, 100.5.
- $price = 99.99; // Float primitive

c) Boolean (bool)

- Holds only two values: true or false.
- Often used in conditional statements.
- $isLoggedIn = true; // Boolean primitive

String (string)

- A sequence of characters.
- Can be enclosed in single (') or double (") quotes.
- $name = “Piyusha"; // String primitive

Data Types in PHP

Data types in PHP categorize values. PHP has three main types of data beyond primitives:

a) Compound Data Types

These hold multiple values.

i) **Array (array)**

- A collection of multiple values.
- Can be indexed (numerical keys) or associative (custom keys).

```
$colors = array("Red", "Green", "Blue"); // Indexed array

$user = array("name" => "Alice", "age" => 25); // Associative array
```

ii) Object (object)

- An instance of a class.
- Stores data & functions related to an entity.

```
class Car {

 public $brand = "Toyota"; // Property of the object

}

$myCar = new Car(); // Object creation
```

Special datatypes

These hold special values.

i) NULL (null)

- Represents a variable with no value.
- $emptyVar = null; // NULL type

ii) Resource (resource)

- Holds external connections (e.g., database connections, file handles).
- $file = fopen("test.txt", "r"); // Resource type (file handle)

Sample program showing all types

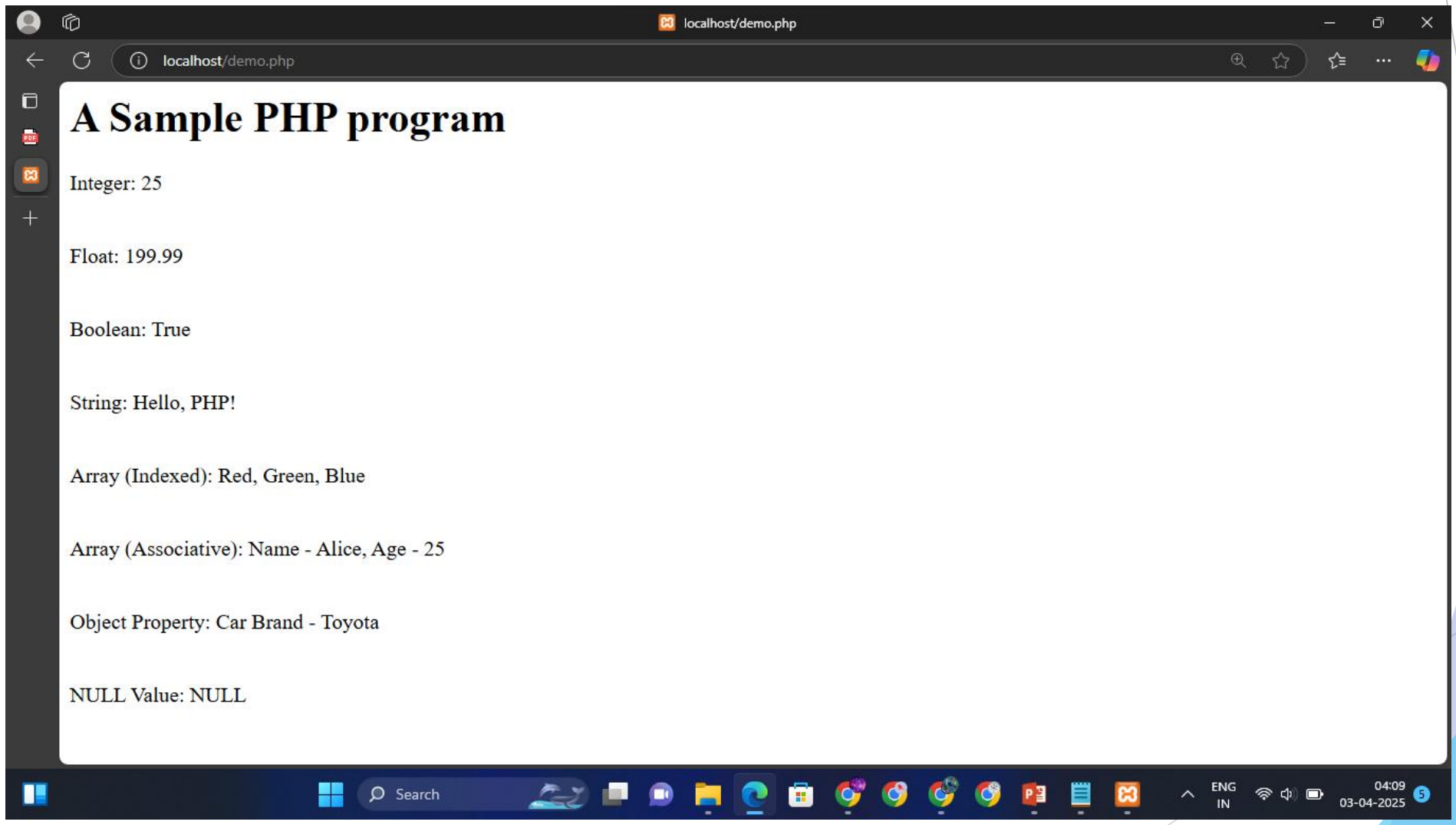
Events.txt | php

File Edit View

```php
<?php
// ---- Primitive Data Types ----
$age = 25; // Integer
$price = 199.99; // Float
$isAvailable = true; // Boolean
$greeting = "Hello, PHP!"; // String

// ---- Compound Data Types ----
$colors = array("Red", "Green", "Blue"); // Indexed Array
$user = array("name" => "Alice", "age" => 25); // Associative Array

// Object Example
class Car {
    public $brand = "Toyota"; // Object Property
}
$myCar = new Car(); // Creating Object

// ---- Special Data Types ----
$unknown = null; // NULL value
$file = fopen("test.txt", "r"); // Resource

// ---- Output ----
echo "Integer: $age\n";
echo "Float: $price\n";
echo "Boolean: " . ($isAvailable ? "True" : "False") . "\n";
echo "String: $greeting\n";

echo "Array (Indexed): " . implode(", ", $colors) . "\n";
echo "Array (Associative): Name - " . $user['name'] . ", Age - " . $user['age'] . "\n";
echo "Object Property: Car Brand - " . $myCar->brand . "\n";
echo "NULL Value: " . ($unknown === null ? "NULL" : "Not NULL") . "\n";
?>
```

Ln 15, Col 2 | 987 characters | 110% | Windows (CRLF) | UTF-8

Output


localhost/demo.php
localhost/demo.php
A Sample PHP program
Integer: 25
Float: 199.99
Boolean: True
String: Hello, PHP!
Array (Indexed): Red, Green, Blue
Array (Associative): Name - Alice, Age - 25
Object Property: Car Brand - Toyota
NULL Value: NULL
Search
ENG
IN
04:09
03-04-2025

PHP Operations

Various types of operators

PHP Operators

Operators are used to perform operations on variables and values.

PHP divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Increment/Decrement operators
- Logical operators
- String operators
- Array operators
- Conditional assignment operators

PHP Arithmetic Operators

The PHP arithmetic operators are used with numeric values to perform common arithmetical operations, such as addition, subtraction, multiplication etc.

Operator	Name	Example	Result
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y
*	Multiplication	$x * $y	Product of $x and $y
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y
**	Exponentiation	$x ** $y	Result of raising $x to the $y'th power

PHP Assignment Operators

The PHP assignment operators are used with numeric values to write a value to a variable.

The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression on the right.

Assignment	Same as...	Description
x = y	x = y	The left operand gets set to the value of the expression on the right
x += y	x = x + y	Addition
x -= y	x = x - y	Subtraction
x *= y	x = x * y	Multiplication
x /= y	x = x / y	Division
x %= y	x = x % y	Modulus

PHP Comparison Operators

Operator	Name	Example	Result
==	Equal	$x == $y	Returns true if $x is equal to $y
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	Returns true if $x is not equal to $y
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y
!==	Not identical	$x !== $y	Returns true if $x is not equal to $y, or they are not of the same type
>	Greater than	$x > $y	Returns true if $x is greater than $y
<	Less than	$x < $y	Returns true if $x is less than $y
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y
<=>	Spaceship	$x <=> $y	Returns an integer less than, equal to, or greater than zero, depending on if $x is less than, equal to, or greater than $y. Introduced in PHP 7.

PHP Increment / Decrement Operators

- The PHP increment operators are used to increment a variable's value.
- The PHP decrement operators are used to decrement a variable's value.

Operator	Same as...	Description
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

PHP Logical Operators

- The PHP logical operators are used to combine conditional statements.

Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $x or $y is true
!	Not	!$x	True if $x is not true

PHP String Operators

- PHP has two operators that are specially designed for strings.

Operator	Name	Example	Result
.	Concatenation	$txt1 . $txt2	Concatenation of $txt1 and $txt2
.=	Concatenation assignment	$txt1 .= $txt2	Appends $txt2 to $txt1

PHP Array Operators

- The PHP array operators are used to compare arrays.

Operator	Name	Example	Result
+	Union	$x + $y	Union of $x and $y
==	Equality	$x == $y	Returns true if $x and $y have the same key/value pairs
===	Identity	$x === $y	Returns true if $x and $y have the same key/value pairs in the same order and of the same types
!=	Inequality	$x != $y	Returns true if $x is not equal to $y
<>	Inequality	$x <> $y	Returns true if $x is not equal to $y
!==	Non-identity	$x !== $y	Returns true if $x is not identical to $y

PHP Conditional Assignment Operators

- The PHP conditional assignment operators are used to set a value depending on conditions

Operator	Name	Example	Result
?:	Ternary	$x = *expr1* ? *expr2* : *expr3*	Returns the value of $x. The value of $x is *expr2* if *expr1* = TRUE. The value of $x is *expr3* if *expr1* = FALSE
??	Null coalescing	$x = *expr1* ?? *expr2*	Returns the value of $x. The value of $x is *expr1* if *expr1* exists, and is not NULL. If *expr1* does not exist, or is NULL, the value of $x is *expr2*. Introduced in PHP 7

Expressions

Overview of PHP expressions

PHP Expressions

- Almost everything in a PHP script is an expression.
- Anything that **has a value** is an expression.
- In a typical assignment statement ($x=100), a literal value, a function or operands processed by operators is an expression
- Anything that appears to the right of assignment operator (=)

```
$x=100; //100 is an expression
$a=$b+$c; //b+$c is an expression
$c=add($a,$b); //add($a,$b) is an expresson
$val=sqrt(100); //sqrt(100) is an expression
$var=$x!=$y; //$x!=$y is an expression
```

Expression with Ternary conditional operator

- Ternary operator has three operands. First one is a logical expression. If it is TRU, second operand expression is evaluated otherwise third one is evaluated

Control Structures

Loops and conditional statements

PHP Control Structures

- Control structures in PHP allow you to control the **flow** of execution based on conditions, loops, and function calls. PHP has three main types of control structures:

- **1. Conditional Statements**

Used to execute different code **based on conditions**.

- **2. Looping Statements**

Used to **repeat** code execution.

- **3. Jump Statements**

Used to **alter** normal loop execution

- **4. Dynamic Controls**

In PHP, these statements are used to **include and reuse files** in scripts. They are **control structures** that manage file inclusion dynamically.

If

- Executes a block of code **only if** the condition is true.

```php
<?php
$age = 18;
if ($age >= 18) {
    echo "You are an adult.\n";
}

?>
```

```
You are an adult.
```

if-else

- Provides an alternative **else block** if the condition is false.

```php
<?php
$score = 90;
if ($score >= 50) {
    echo "You passed!\n";
} else {
    echo "You failed.\n";
}

?>
```

```
You passed!
```

if-elseif-else

- Checks multiple conditions in sequence.

```php
<?php
$marks = 85;
if ($marks >= 90) {
    echo "Grade: A\n";
} elseif ($marks >= 75) {
    echo "Grade: B\n";
} else {
    echo "Grade: C\n";
}
?>
```

Switch case

- Used for multiple fixed-value conditions.

```php
<?php
$day = "Monday";
switch ($day) {
    case "Monday":
        echo "Start of the workweek.\n";
        break;
    case "Friday":
        echo "Weekend is near!\n";
        break;
    default:
        echo "It's a normal day.\n";
}
?>
```

```
Start of the workweek.
```

While loop

- Repeats as long as the condition is true.

```php
<?php
$x = 1;
while ($x <= 5) {
    echo "Number: $x\n";
    $x++;
}

?>
```

Number: 1 Number: 2 Number: 3 Number: 4
Number: 5

do-while Loop

- Executes **at least once**, then checks the condition.

```php
<?php
$y = 1;
do {
    echo "Value: $y\n";
    $y++;
} while ($y <= 3);

?>
```

```
Value: 1 Value: 2 Value: 3
```

for Loop

- Used for a known number of iterations.

```php
<?php
for ($i = 1; $i <= 5; $i++) {
    echo "Iteration: $i\n";
}

?>
```

Iteration: 1 Iteration: 2 Iteration: 3 Iteration: 4 Iteration: 5

foreach Loop

- Iterates over associative arrays.

```php
<?php
$colors = array("Red", "Green",
"Blue");
foreach ($colors as $color) {
    echo "Color: $color\n";
}

?>
```

```
Color: Red Color: Green Color: Blue
```

break Statement

- Exits a loop immediately.

```php
<?php
for ($i = 1; $i <= 10; $i++) {
    if ($i == 5) break; // Stops at 5
    echo "Number: $i\n";
}

?>
```

```
Number: 1 Number: 2 Number: 3 Number: 4
```

continue Statement

- Skips the current iteration and moves to the next.

```php
<?php
for ($i = 1; $i <= 5; $i++) {
    if ($i == 3) continue; // Skips 3
    echo "Count: $i\n";
}

?>
```

Count: 1 Count: 2 Count: 4 Count: 5

PHP include, include_once, require, require_once

In PHP, these statements are used to include and reuse files in scripts. They are control structures that manage file inclusion dynamically.

1] **include Statement**: Includes and executes a file. If the file is missing, PHP shows a warning but continues execution.

2] **include_once Statement**: Similar to include, but ensures the file is included only once. Prevents re-declaration errors.

3] **require Statement**: Mandatory inclusion of a file. If the file is missing, PHP throws a fatal error and stops execution.

4] **require_once Statement**: Similar to require, but ensures the file is included only once.

Syntax

```php
<?php
include "header.php"; // Includes header.php
echo "Main content here!";

include_once "config.php"; // Included only once
include_once "config.php"; // This will be ignored


require "config.php"; // Stops execution if missing
echo "This won't execute if config.php is missing!";

require_once "functions.php"; // Included only once
require_once "functions.php"; // Ignored if already included

?>
```

Output

Executing a PHP program

Thank you!

Presented by Piyusha Supe