Unit 1: Getting started with PHP

Unit Structure

- Learning Objectives
- Introduction
- Getting with PHP
- PHP Comments
- Working with variables
- Working with constants
- Working with simple expression
- Working with operators

1.1 LEARNING OBJECTIVE

After studying this unit student should be able to:

- Understand basics of PHP
- Understand declaration, initialization and implementing variable in PHP
- Understand various operators and expression in PHP.

1.2 INTRODUCTION

PHP is a general-purpose scripting language that is especially suited to server-side web development where PHP generally runs on a web server.

Any PHP code in a requested file is executed by the PHP runtime, usually to create dynamic web page content or dynamic images used on Web sites or elsewhere.

It can also be used for command-line scripting and client-side graphical user interface (GUI) applications.

PHP can be deployed on most Web servers, many operating systems and platforms, and can be used with many relational database management systems (RDBMS).

It is available free of charge.

Originally designed to create dynamic Web pages, PHP now focuses mainly on server-side scripting, and it is similar to other server-side scripting languages that provide dynamic content from a Web server to a client, such as Microsoft's ASP.NET, Sun Microsystems' JavaServer Pages, and mod_perl.

1.2.1 The main characteristics of PHP are:

- PHP - also known as Hypertext Preprocessor, it is originally stood for "Personal Home Page".
- PHP is free open source software released under the PHP License
- PHP is a server-side scripting language, it is embedded to HTML.
- Fast execution of scripts
- You can connect and manipulate various databases likes MariaDB/MySQL, PostgreSQL, Oracle, Microsoft SQL Server etc.
- Supported by most web servers and operating systems.
- Supports many standard network protocols libraries available for IMAP, NNTP, SMTP, POP3
- Supports many database management systems libraries available for UNIX DBM, MySQL, Oracle

1.2.2 Main uses of PHP

1. Server-side scripting: This is the most traditional and main use for PHP. You need three things to make this work:
   - The PHP parser (CGI or server module),
   - A web server: needs a connected PHP installation
   - A web browser: access PHP page through URL

2. Command Line Scripting: You can make a PHP script to run without any server or browser. You only need the PHP parser to use as Command line Scripting.

3. Client-side GUI applications: PHP is probably not the very best language to write window (GUI) applications, but PHP-GTK (PHP Graphics Tool Kit) can be used to write such programs.

1.3 GETTING STATED WITH PHP

Setting the Server

Download XAMPP from https://www.apachefriends.org/download.html and install the application (as demonstrated during the face-to-face session and the first online session)

Writing your first PHP page:

The following steps will help you create your first PHP file:

1. Open Notepad/Notepad++ or any editor
2. Write the following code:

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

3. Save the file as test.php into the htdocs folder of XAMPP server (Assume C:\xampp\htdocs\)

4. To run the php code

   - Open any browser.
   - Type: http://localhost/test.php

5. The output will be as below:

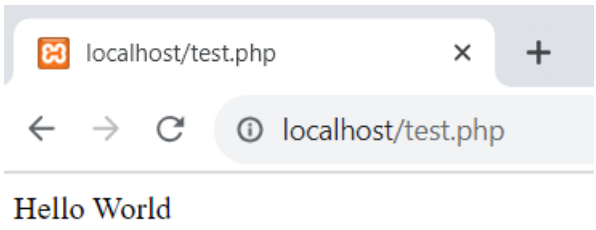


Figure 1 file name: test.php, Hello World

PHP Syntax

The <?php echo "Hello World";?> PHP script run at only server side.
The response is sent back to the browser (Client system) in plain HTML (“Hello World”).
The PHP format (syntax) and the related meanings (semantics) of the text form a set of rules that define how a PHP program can be written and interpreted.

The PHP processor only parses code within its delimiters (the trigger symbols <?php….?>). Anything outside its delimiters is sent directly to the output and not parsed by PHP.

There are four ways of including PHP in a web page.

Method 1:

<?php echo ("Hello world");?>

This method is clear and unambiguous.

Method 2:

```
<script language = "php">
echo "Hello world";
</script>
```

Method 3:

```
<? echo "Hello world";?>
```

Method 4:

```
<% echo "Hello world"; %>
```

Another short opening tags (<%=) are also available for use. This method depends on the server configuration.

echo and print command

We can also use print instead of echo in the methods both functions are nearly identical. The major difference between them is print is slower than echo because the former will return a status indicating if it was successful or not in addition to text to output, whereas the latter does not return a status and only returns the text for output.

1.4 PHP COMMENTS

A comment is the part of a program that exists only for the developer reader and exclude for displaying/executing the programs result. There are two types of commenting formats in PHP.

Single line comment: it is generally used for short explanations or notes relevant to the local code. below code is examples of single line comments

Example of Single line Comment:

```
<?php

// This is a First line comment using double slash ()

echo “Welcome”;

?>
```

Multi-line comment (/* …..*/): It is generally used for multiple line code remarks or notes, below code is examples of multi-line comments**:**

```
<?php

/* This is a multi-line comment ……..

……..

Apply comment using star and slash symbol combination  */

echo "Welcome";

?>
```

1.5 WORKING WITH VARIABLES IN PHP

The main way to store information in the middle of a PHP program is by using a variable.

PHP Variables: A variable can have a short name (like a and b) or a more descriptive name like age, surname, total_volume, etc.

```
$_1name=”John”; // Valid Variable name in PHP

$1_name=”John”; // Invalid Variable name in PHP since is it not allowed to start with number
```

Basic rules for PHP variables:

- A variable always **starts with the $ sign**, followed by the name of the variable.
- A variable name **must start with a letter or the underscore character**.
- A variable name **cannot start with a number**.
- A variable name **can only contain alpha-numeric characters and underscores** (A-z, 0-9 and underscore _ character)
- **Variable names are case-sensitive** ($age and $AGE are two different variables)

Here are the most important things to know about variables in PHP:

- Variables are assigned with the = operator (It is called as assignment operator), with the variable on the left-hand side and the expression to be evaluated on the right.

```
$Module_Name = ”BSc (Hons) Computer Science”;

$semester = 3;
```

- Variables can, but do not need, to be declared before assignment.
- Variables in PHP do not have inherent types - a variable does not know in advance whether it will be used to store a number or a string of characters.
- PHP automatically converting data types from one to another when necessary.
- In other languages such as C, C++, and Java, the programmer must declare the name and type of the variable before using it.

Datatype PHP has a total of eight data types which we use to construct declare our variables:

Sr. No	Datatype	Description
1	Integers	It is whole numbers, without a decimal point, like 4195. Example: $a=10; $b=2104;
2	Double	It is floating-point numbers, like 3.14159 or 49.1. Example: $a=5.3; $b=21.04;
3	Booleans	It has only two possible values either true or false. Example: $a=true; $b=false;

4	NULL	It is a special type that only has one value: NULL. Example: $a=null; $b=NULL;
5	Strings	It is a sequence of characters, like PHP supports string operations. Example: $programme= “BSc (Hons) Computer Science”; $module= ”Server-Side Web Technologies”;
6	Arrays	It is a namedarray() and indexed collections of other values. Example: $courses = array(“B.Sc.-IT”, “B.Sc. CS”, “M.Sc.-IT”, “M.Sc.-CS”)
7	Objects	It is instances of user-defined classes, which can pack up both other kinds of values and functions that are specific to the class. Example: $program = new Course();
8	Resources	It is special variables that hold references to resources external to PHP (such as database connections). Example :$open_todo_file = fopen("todo_list.txt", "r");

The first five are simple types, and the next two (arrays and objects) are compound type. The compound types can package up other uninformed values of subjective type, whereas the simple types cannot.

Kindly remember that PHP variable names are case-sensitive!

Example: $age=”Twenty-five”, $AGE=” Twenty-five”; $age and $AGE are two different variables.

Note: When you assign a text value to a variable, put quotes around the text.

Note: Unlike other programming languages, PHP has no command for declaring a variable. It is created the moment you first assign a value to it.

PHP has a useful function named var_dump() that prints the current type and value for one or more variables. Arrays and objects are printed recursively with their values indented to show structure.

PHP Variables Scope:

In PHP script writing, the variables can be declared anywhere in the script.

The scope of a variable is the part of the script where the variable can be referenced or used.

PHP has four different variable scopes:

1. Local variables
2. Global variables
3. Static variables
4. Function parameters

1. Local Variables

A variable declared in a function is considered local; that is, it can be referenced solely in that function. Any assignment outside of that function will be considered to be an entirely different variable from the one contained in the function:

```
<?php
    $a = 5; // global scope
        function myTest()
        {
        echo $a; // local scope
        }
    myTest();
?>
```

The script above will not produce any output because the echo statement refers to the local scope variable $a, which has not been assigned a value within this scope.

You can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared.

Local variables are deleted as soon as the function is completed.

2. Global variables

Global scope refers to any variable that is defined outside of any function.

Global variables can be accessed from any part of the script that is not inside a function.

To access a global variable from within a function, use the global keyword:

```
<?php
$a = 15;
$b = 6;
function myTest()
{
    global $a, $b;
    $b = $a + $b;
}
myTest();
echo "The value of the operation is ".$b;
?>
```

The script above will output 21.

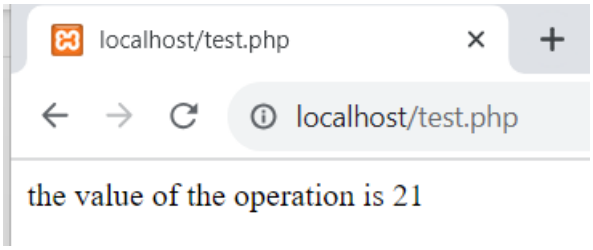


Figure 5 Global Variable output

PHP also stores all global variables in an array called $GLOBALS[index]. Its index is the name of the variable. This array is also accessible from within functions and can be used to update global variables directly.

The example above can be rewritten as this:

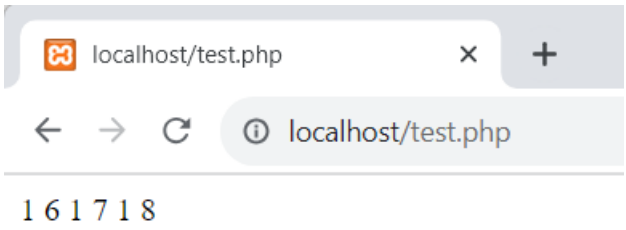
```
<?php
$a = 15;
$b = 6;
        function myTest()
    {
    $GLOBALS['b'] = $GLOBALS['a'] + $GLOBALS['b'];
    }
myTest();
echo "The value of the operation is ".$b;
?>
```

3. Static variables

When a function is completed, all of its variables are normally deleted. However, in some cases, we really want to store the variables even after the fulfillment of function execution. To do this, use the static keyword when you first declare the variable.

Example,

```
<?php
  // function to exhibit static variables
function static_var()
{
    // static variable
    static $sum = 5;
    $num = 0;

    $sum++;
    $num++;

    echo $num, "\n";
    echo $sum, "\n";
}

// first function call
static_var();

  // second function call
static_var();

// third function call
static_var();

?>
```



In the above code, the function “static_var();” is called 3 times. Check the value of $num and $sum. Each time the function is called, that value of static variable $sum is incremented by 1, that is, the variable $sum still has the information it contained from the last time the function was called. On the other hand, the value for the local variable $num is still 1.

Note: The variable is still local to the function.

4. Function Parameters

Function parameters are declared after the function name and inside parentheses. They are declared much like a typical variable would be. A parameter is a local variable whose value is passed to the function by the calling code.

Parameters are declared in a parameter list as part of the function declaration:

```
function myTest($parameter1, $parameter2,...)
{
// function code
}
```

Parameters are also called arguments.

1. 6 Working with operators in PHP

Operators are used to perform operations on variables and values.

PHP divides the operators in the following groups:

1. Arithmetic operators
2. Assignment operators
3. Comparison operators
4. Increment/Decrement operators
5. Logical operators
6. String operators
7. Array operators
8. PHP Type Operator

1. Arithmetic Operators

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

Operator	Name	Description
a+b	Addition	Sum of variables a and b, for example 2+3=5
a-b	Subtraction	Difference of a and b, for example 5-2=3
a*b	Multiplication	Product of a and b, for example 5*2=10
a/b	Division	Quotient of a and b, for example 10/2=5
a%b	Modulus	Remainder of a divided by b, for example 3%2=1
-a	Negation	Opposite of x, for example -5
a.b	Concatenation	Used to concat, for example $a=”Paul”; $b=”Smith”; echo $a.$b; // output is PaulSmith

2. 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.

Example, $Name = “Paul Smith”;

3. Comparison Operators

The PHP comparison operators are used to compare two values (number or string):

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 or equal to $y
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal or to $y

4. 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	Name	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

5. 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

6. 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

7. 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