Expressions & Control Statements in PHP

Unit - I

1.1 History & Advantages of PHP

- PHP is a simple yet powerful scripting language designed for creating HTML content.
- Originally derived from *Personal Home Page* tools, now stands for *PHP:Hypertext Preprocessor*.
- PHP executes on the server, while a comparable alternative, *JavaScript*, executes on the client.

1.1 History & Advantages of PHP

- PHP is an alternative to Microsoft's Active Server Page (ASP).
- PHP script is embedded within a web page along with HTML.
- Before the page is sent to a user that has requested it, the Web server calls PHP to interpret and perform the operations called for in the PHP script.

1.1 History & Advantages of PHP

- PHP can be used in 3 primary ways:
  - ❑Server Side Scripting:
  - ❑Command Line Scripting:
  - ❑Client side GUI Applications:

- An HTML page that includes a PHP script is typically given a file name suffix of ".php", ".php7" or ".phtml"

- PHP scripts can only be interpreted on a server that has PHP installed.

1.1 History & Advantages of PHP

- Advantages:

  - Open Source:

  It is open source and free of cost, which helps developers to install it quickly and readily available for use. Multiple frameworks are available for the developer to choose.

  - Platform Independent:

  Supported by all the operating systems like Windows, Unix, Linux etc. it can be integrated with other programming language & databases easily.

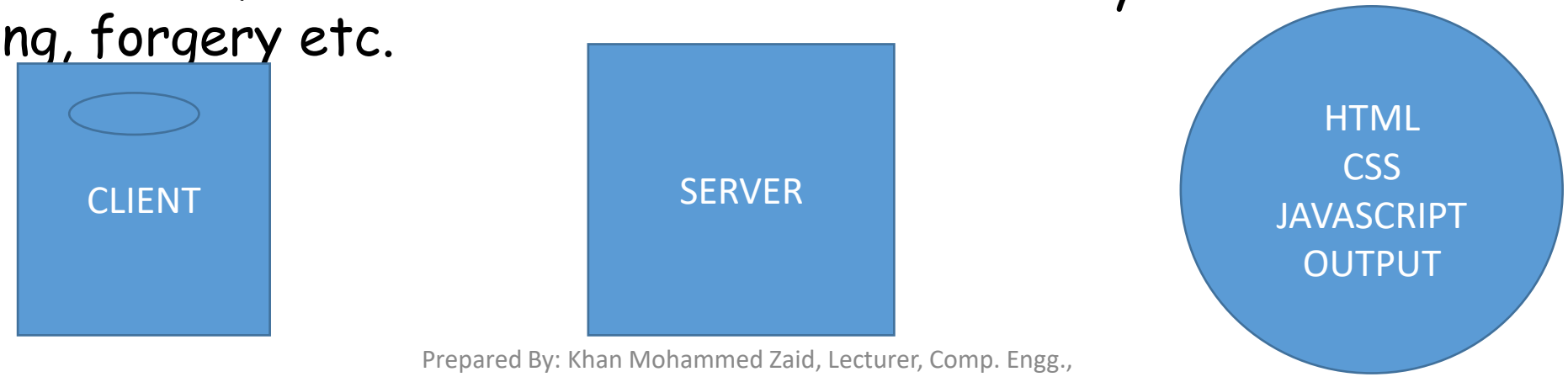
  - Simple and Easy to learn

1.1 History & Advantages of PHP

- Advantages:

  - Database connectivity:
    It can be connected securely with the database. Multiple databases can be integrated with PHP. E.g. MySQL, MariaDB, Db2, MongoDB, Oracle, PostgreSQL, and SQLite.

  - Security
    PHP frameworks has built-in feature and tools make it easier to protect the web applications from the outer attacks and security threats like data tampering, forgery etc.



1.1 Syntax of PHP

- PHP can be placed anywhere in the document:

- PHP script starts with *<?php* And ends with *?>*

e.g.

```
<html>
    <body>
        <h1>My first PHP</h1>
        <?php
            Echo "Hello World!";
        ?>
    </body>
</html>
```

1.1 Syntax of PHP

- PHP statements end with a semicolon ;
- Keywords, classes, functions, user-defined functions are not case sensitive.
- Variables with different cases are treated differently.
- Single line comments in PHP:

  // This is a single line comment

  # This is a single line comment

  /* This is a multi line comment */

1.2 Variables in PHP

- In PHP, variable starts with the $ sign, followed by the name of the variable.
- A variable is created the moment you first assign a value to it.

E.g.

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

1.2 Variables in PHP

- Rules for variables:
  - A variable 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 _).
  - Variable names are case-sensitive ($age and $AGE are two different variables)

1.2 Variables in PHP

- Echo statement can be used to output data to the screen.

e.g.

```
<?php
$txt = “PHP";
echo "I love $txt!";
?>
```

or

```
<?php
$txt = “PHP";
echo "I love " . $txt . "!";
?>
```

Or

```
<?php
$x = 5;
$y = 4;
echo $x + $y;
?>
```

1.2 Variables in PHP

- Scope of variables in PHP:
  - The scope of a variable is the part of the script where the variable can be referenced/used.
  - PHP has 3 different variable scopes:
    1. Local
    2. Global
    3. Static

1.2 Variables in PHP

- Scope of variables in PHP:
  - **Local:**
    - A variable declared within a function has a LOCAL SCOPE and can only be accessed within that function.
    - Local variables with the same name can be used in different functions.

E.g.

```
<?php
  function myTest() {
      $x = 5; // local scope
      echo "<p>Variable x inside function is: $x</p>";
  }
  myTest();
  function myTest2() {
      $x = 5; // local scope
      echo "<p>Variable x inside function is: $x</p>";
  }
  ?>
```

1.2 Variables in PHP

- Scope of variables in PHP:
  - **Global:**
    - *A variable declared outside a function has a GLOBAL SCOPE and can only be accessed outside a function.*

e.g.:

```
<?php
  $x = 5; // global scope
  echo $x;
  function myTest() {
     // using x inside this function will generate an error
     echo "<p>Variable x inside function is: $x</p>";
  }
  myTest();

  echo "<p>Variable x outside function is: $x</p>";
  ?>
```

1.2 Variables in PHP

- Scope of variables in PHP:
  - ***Global keyword*:**
    - ❑ *The global keyword is used to access a global variable from within a function.*
    - ❑ *To do this, use the global keyword before the variables (inside the function)*

```
<?php

$x = 5;
$y = 10;
function myTest() {
    global $x, $y;
    $y = $x + $y;
}
myTest();
echo $y; // outputs 15
?>
```

1.2 Variables in PHP

- Scope of variables in PHP:
  - ***Global array*:**
    - ❑*PHP also stores all global variables in an array called GLOBALS[index].*
    - ❑*The index holds the name of the variable.*
    - ❑*This array is also accessible from within functions and can be used to update global variables directly.*

```
<?php
$x = 5;
$y = 10;
function myTest() {
    $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
}
myTest();
echo $y; // outputs 15
?>
```

1.2 Variables in PHP

- Scope of variables in PHP:
  - ***Static*:**
    - ❑*It is used to declare properties and methods of a class as static. Static properties and methods can be used without creating an instance of the class.*
    - ❑*The static keyword is also used to declare variables in a function which keep their value after the function has ended.*

```
<?php
function add1() {
  static $number = 0;
  $number++;
  return $number; }
echo add1();
echo "<br>";
echo add1();
echo "<br>";
echo add1(); ?>
```

Output:

1

2

3

1.2 Variables in PHP

- Echo & Print statement in PHP:

  - both are used to output data to the screen.
  - Echo has no return value while print has a return value of 1.
  - Echo can take multiple parameters while print can take 1 argument.
  - Echo is marginally faster than print.

1.2 Variables in PHP

- Echo:
  - Echo can be used with or without parenthesis: echo or echo()

e.g.

```
<?php
  echo "<h2>PHP is Fun!</h2>";
  echo "Hello world!<br>";
  echo "I'm about to learn PHP!<br>";
  echo "This ", "string ", "was ", "made ", "with multiple parameters.";
  ?>
```

e.g.

```
<?php
  $txt1 = "Learn PHP";
  $txt2 = "W3Schools.com";
  $x = 5;
  $y = 4;
  echo "<h2>" . $txt1 . "</h2>";
  echo "Study PHP at " . $txt2 . "<br>";
  echo $x + $y;
  ?>
```

1.2 Variables in PHP

- Print:
  - Print can be used with or without parenthesis: print or print()

e.g.

```
<?php
print "<h2>PHP is Fun!</h2>";
print "Hello world!<br>";
print "I'm about to learn PHP!";
?>
```

e.g.

```
<?php
$txt1 = "Learn PHP";
$txt2 = "W3Schools.com";
$x = 5;
$y = 4;
print "<h2>" . $txt1 . "</h2>";
print "Study PHP at " . $txt2 . "<br>";
print $x + $y;
?>
```

1.2 Datatypes in PHP

- PHP supports following datatypes:
  - String
  - Integer
  - Float
  - Boolean
  - Array
  - Object
  - Null
  - Resource

1.2 Datatypes in PHP

- String:
  - A sting is a sequence of characters, like "Hello World!"
  - Single or double quotes can be used to initialize.

- Integer:
  - An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.
  - An integer must have at least one digit.
  - An integer must not have a decimal point.
  - An integer can be either positive or negative

- Float:
  - A float (floating point number) is a number with a decimal point or a number in exponential form.

1.2 Datatypes in PHP

- Boolean:
  - A Boolean represents two possible states: TRUE or FALSE.
  - Booleans are often used in conditional testing.

- Array:

- Object:
  - An object is a data type which stores data and information on how to process that data.
  - In PHP, an object must be explicitly declared. For each object a class must be declared.

1.2 Datatypes in PHP

- NULL:
  - Null is a special data type which can have only one value: NULL.
  - A variable of data type NULL is a variable that has no value assigned to it.
  - If a variable is created without a value, it is automatically assigned a value of NULL.

- Resource:
  - The special resource type is not an actual data type.
  - It is the storing of a reference to functions and resources external to PHP.
  - A common example of using the resource data type is a database call.

1.2 Datatypes in PHP

- Constant:
  - Constants are like variables except that once they are defined they cannot be changed or undefined.
  - A valid constant name starts with a letter or underscore (no $ sign before the constant name).
  - Unlike variables, constants are automatically global across the entire script.
  - define() function is used to create a constant.

Syntax: define(name, value, case-insensitive)

name: Specifies the name of the constant

value: Specifies the value of the constant

case-insensitive: Specifies whether the constant name should be case-insensitive. Default is false

1.2 Operators in PHP

- Following types of operators are available in PHP:
  - Arithmetic operator
  - Assignment operator
  - Comparison operator
  - Increment/Decrement operator
  - Logical operator
  - String operator
  - Array operator
  - Conditional Assignment operator

1.2 Operators in PHP

- Arithmetic operator:
  - Addition
  - Subtraction
  - Multiplication
  - Division
  - Modulus

  - Exponentiation (**):
    Result of raising x to the yth power.

1.2 Operators in PHP

- Assignment operator:
  - Equal: ==
  - Identical (===) : Returns true if x is equal to y and they are of same type.
  - Not Equal: != / <>
  - Not Identical !== : Returns true if x is not equal to y or they are not of the same type.
  - Less than < :
  - Greater than > :
  - Less than Equal to <= :
  - Greater than Equal to >= :
  - Spaceship <=> : Returns an Integer

<0	if	x is less than y
0	if	x is equal to y
0>	if	x is greater than y

1.2 Operators in PHP

- String operator:
  - Concatenation (.): Concatenates 2 strings
  - Concatenation Assignment (.=): Appends string2 to string1.

1.2 Operators in PHP

- Array operator:
  - Union + : Union of 2 arrays
  - Equality == : Returns true if 2 arrays have same key/value pairs.
  - Identity === : Returns true if 2 arrays have same key/value pairs in same order and of the same type.
  - Inequality != , <> :
  - Non Identity: Returns true if x is not identical to y.

1.2 Operators in PHP

- Conditional Assignment operator:
  - Ternary ?: :
  - Null Coalescing ?? : $x = exp1 ?? exp2
    Value of x is exp1 if it exists and is not NULL.
    If exp1 doesn't exists or is NULL, the value of x is exp2.

1.3 Decision making Control Statements in PHP

- if Statement:

  Syntax:

```
if (condition)
{
  code to be executed if condition is true
}
```

1.3 Decision making Control Statements in PHP

- if else Statement:

Syntax:

```
if (condition)
{
  code to be executed if condition is true
}
else
{
  code to be executed if condition is false
}
```

1.3 Decision making Control Statements in PHP

- if… else if… else Statement:

Syntax:

```
if (condition1)
{
  code to be executed if condition1 is true
}
elseif(condition2)
{
  code to be executed if condition1 is false and condition2 is true
}
else
{
  code to be executed if all conditions are false
}
```

1.3 Decision making Control Statements in PHP

- switch Statement:

  Syntax:

```
switch (expr)
{
  case label1:
        code to be executed if expr=label1;
        break;
   case label2:
        code to be executed if expr=label2;
        break;
  ...

default:
  code to be executed if n is different from all labels.
}
```

1.4 Loops in PHP

- While loop:

  Syntax:

```
initialization
while (condition)
{
  code to be executed;
  increment / decrement;
}
```

1.4 Loops in PHP

- Do While loop:

Syntax:

```
initialization
do
{
   code to be executed;
   increment / decrement;
} while (condition);
```

1.4 Loops in PHP

- for loop:

  Syntax:

```
for(initialization ; condition ; increment / decrement)
{
  code to be executed;
}
```

1.4 Loops in PHP

- for each loop:
  - It loops through a block of code for each element in an array.

Syntax:

```
foreach($array  as  $value)
{
  code to be executed;
}
```

e.g.

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