PHP Basics

❖ Basic PHP Syntax

PHP is a **server- side** language which means that PHP script will run on the web server and after execution the **result** will be sent to the browser as html.

PHP Syntax:

- ✓ **<?php ………….?>**
- ✓ Each line code must end with **;**
- ✓ The file saved with **.php extension.**

❖ Case-Sensitivity in PHP

- ✓ PHP is case sensitive in: **variable names**
- ✓ PHP is not case sensitive in: **function names, keywords and classes names**

❖ Comments in PHP

1. **Single line comment:**
   //…………………………………….
   #…………………………….

2. **Multiple line comment**
   /*……………….
   ……………….*/

❖ Outputting Data to the browser

1. **echo statements:**

   echo(st1,st2,st3,……….);

   - ✓ Is used to output one or more string.
   - ✓ Can contain html tags.
   - ✓ If it used to print **single** string the ( ) are **optional**.
   - ✓ If it used to print **more** than one string **don't** use ( ).

Example.

```
<?php
echo ("<hr>");
echo ("<p><b>BAU University</b></p>");
echo "hello<br/>";
echo "One","Two","Three";
?>
```

❖ Variables in PHP

1. **PHP Data Types**
   You don't have to determine the data type of a variable when it is declared.
   PHP supports the following data types:

- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL

2. **Declaring Variables.**
   A variable starts with the **$** sign, followed by the name of the variable.
   When you assign a **text value** to a variable, put **quotes** around the value.

Task4. Write the next code and determine what the output is?

```
<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
$z= true;
$w=null;

echo $txt;
echo "<br>";
echo $x;
echo "<br>";

echo "$x";  // it will not be used as string ,it is 5
echo "<br>";

echo $y;
echo "<br>";
echo $z;
echo "<br>";
echo $w;

?>
```

```
Hello world!
5
5
10.5
1
```

3. Constants Declaration.

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) to create a constant, use the define() function.

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

Example1. creates a constant with a **case-sensitive** name:

```
<?php
// case-sensitive constant name
define("GREETING", "Welcome to BAU!");
echo GREETING;
echo"</br>";
Echo "GREETING"; // not as variable it will print the string
?>
```

```
Welcome to BAU!
GREETING
```

Example2.creates a constant with a **case-insensitive** name:

```
<?php
// case-insensitive constant name
define("GREETING","Welcome to BAU!", true);
echo greeting;
?>
```

```
Welcome to BAU!
```

Constants are Global

Constants can be used across the entire function.

```
<?php
define("GREETING", "Welcome to W3Schools.com!");

function myTest() {
    echo GREETING;
}
myTest();
?>
```

❖ Concatenation in PHP

You can concatenate two values together using the dot **(.)**

Example:

```
$txt2 = "WEB Lab";
echo "Study PHP at " . $txt2 . "<br>";
```

❖ Functions in PHP

1. Creating and calling functions

Syntax

```
function functionName() {
    code to be executed;
}
```

- ✓ A user defined function declaration starts with the word "function":
- ✓ A function name can start with a letter or underscore (not a number)
- ✓ Function names are **NOT** case-sensitive

Example1. Function without parameters

```
<?php
function writeMsg() {
    echo "Hello world!";
}

writeMsg(); // call the function
?>
```

```
Hello world!
```

Example2. Function with parameters

```
<?php
function familyName($fname) {
    echo "$fname Ayyash.<br>";
}
familyName("Ahmad");
familyName("Abdullah");
familyName("Leen");
familyName("Tamara");
familyName("Noor");
?>
```

```
Ahmad Ayyash.
Abdullah Ayyash.
Leen Ayyash.
Tamara Ayyash.
Noor Ayyash.
```

Example3. Function with parameters

```
<?php
function familyName($fname, $year) {
    echo "$fname Al-Abadi. Born in $year <br>";
}
familyName("Ayman", "1975");
familyName("Jumana", "1978");
familyName("Fadi", "1983");
?>
```

```
Ayman Al-Abadi. Born in 1975
Jumana Al-Abadi. Born in 1978
Fadi Al-Abadi. Born in 1983
```

Example4. Function with parameters

```
<?php
function countNum($n) {
    echo "$n"."<br>";
}
for($i=1; $i<=10 ; $i++)

countNum($i);

?>
```

Print numbers from 1 to 10

Example6. PHP Functions - Returning values

```
<?php
function sum($x, $y) {
    $z = $x + $y;
    return $z;
}

echo "5 + 10 = " . sum(5, 10) . "<br>";
echo "7 + 13 = " . sum(7, 13) . "<br>";
echo "2 + 4 = " . sum(2, 4);
?
```

```
5 + 10 = 15
7 + 13 = 20
2 + 4 = 6
```

❖ Variables Scope in PHP

1. **Local scope**

A variable declared within a function has a **LOCAL SCOPE** and can only be accessed within that function:

Example

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

// using x outside the function will generate an error
echo "<p>Variable x outside function is: $x</p>";
?>
```

```
Variable x inside function is 5

error
```

2. **Global scope**

A variable declared **outside** a function has a GLOBAL SCOPE and can only be accessed **outside a function**:

Example1:

```
$x = 5; // global scope

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>";
?>
```

```
Error

Variable x outside function is:5
```

➢ Using Global variables in Functions

PHP 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.

Example3:

Output

```
<?php
$x = 5;
$y = 10;

function myTest() {
    $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
}

myTest();
echo $y; //
?>
```

Arrays in PHP

1. Indexed Arrays

Arrays with a numeric index .There are two ways to create indexed arrays:

1. The index can be assigned automatically (index always starts at 0), like this:
   $cars = **array**("Volvo", "BMW", "Toyota");

or **2.** the index can be assigned manually:

$cars[0] = "Volvo";
$cars[1] = "BMW";
$cars[2] = "Toyota";

Example1.

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2]
. ".";
?>
```

- **The count Function**

Get The **Length of an Array** - The count() Function

Example2.

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo count($cars);
?>
```

- **Loop through an Indexed Array**

To loop through and print all the values of an indexed array, you could use **for** loop.

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
$arrlength = count($cars);

for($x = 0; $x < $arrlength; $x++) {
    echo $cars[$x];
    echo "<br>";
}
?>
```

```
Volvo
BMW
Toyota
```

Example3.using for each loop

```
<?php
$age = array("35","37","43");

foreach($age as $x) {
    echo  $x;    echo "<br>";}

?>
```

```
35

37

43
```

2. Associative Arrays

Associative arrays are arrays that use named keys that you assign to them. There are two ways to create an associative array:

```
1. $age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
```

or:

```
2. $age['Peter'] = "35";
   $age['Ben'] = "37";
   $age['Joe'] = "43";
```

The named keys can then be used in a script; key can be **number** or **string**

The value can be of any type.

Example. $color=array(1=>"Red","b"=>"blue","g"=>"green");

Example1.

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

```
Peter is 35 years old.
```

- **Foreach loop through an Associative Array**

To loop through and print all the values of an associative array, you could use a **foreach** loop.

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

foreach($age as $x => $x_value) {
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";}?>
```

❖ Sort Functions for Arrays in PHP

PHP array sort functions:

- sort() - sort arrays in ascending order
- rsort() - sort arrays in descending order
- asort() - sort associative arrays in ascending order, according to the value
- ksort() - sort associative arrays in ascending order, according to the key

1. Sort Array in Ascending Order - sort()

To sorts the elements of the $cars array in ascending alphabetical order:

Example

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
sort($cars);
?>
```

To sorts the elements of the $numbers array in ascending numerical order:

Example

```
<?php
$numbers = array(4, 6, 2, 22, 11);
sort($numbers);
?>
```

2. Sort Array in Descending Order - rsort()

Ex. To sorts the elements of the $cars array in descending alphabetical order:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
rsort($cars);
?>
```

```
Volvo
Toyota
BMW
```

Ex. To sorts the elements of the $numbers array in descending numerical order:

```
<?php
$numbers = array(4, 6, 2, 22, 11);
rsort($numbers);
?>
```

```
22 11 6 4 2
```