Instructor: Sudad H. Abed / Email: shabed@uoanbar.edu.iq

Lecture Reference: www.w3schools.com ,
PHP and MySQL for Dynamic Web Sites, 4th Edition

Chapter 2: Programming with PHP

PHP String Functions

PHP has built-in functions that manipulates string:

- **Get the Length of a String**
  The PHP strlen() function returns the length of a string.
  Ex:
```
<?php
echo strlen("Hello world!"); // outputs 12
?>
```
- **Count the Number of Words in a String**
  The PHP str_word_count() function counts the number of words in a string.
  Ex:
```
<?php
echo str_word_count("Hello world!"); // outputs 2
?>
```
- **Search for a Specific Text Within a String**

The PHP strpos() function searches for a specific text within a string. If a match is found, the function returns the character position of the first match. If no match is found, it will return FALSE.

Ex:

```
<?php
echo strpos("Hello world!", "world"); // outputs 6
?>
```

PHP Operators

Operators are used to perform operations on variables and values. The following groups show some of the operators in PHP:

- Arithmetic operators
- Comparison operators
- String operators
- Logical operators

➢ 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 (Introduced in PHP 5.6)

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

➢ 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 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 Conditional Statements

In PHP we have the following conditional statements:

- **if statement** - executes some code if one condition is true.
- **if...else statement** - executes some code if a condition is true and another code if that condition is false.
- **if...elseif....else statement** - executes different codes for more than two conditions.
- **switch statement** - selects one of many blocks of code to be executed.

➢ if Statement

The if statement executes some code if one condition is true.

Syntax

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

Ex:

```
<?php
$t = date("H");
if ($t < "20") {
    echo "Have a good day!";
}
?>
```

➢ if...else Statement

The if....else statement executes some code if a condition is true and another code if that condition is false.

Syntax

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

Ex:

```
<?php
$t = date("H");
if ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
?>
```

- **if...elseif....else Statement**

The if....elseif...else statement executes different codes for more than two conditions.

Syntax

```
if (condition) {
    code to be executed if this
condition is true;
} elseif (condition) {
    code to be executed if this
condition is true;
} else {
    code to be executed if all
conditions are false;
```

- **switch Statement**

Use the switch statement to select one of many blocks of code to be executed**.**

Syntax

```
switch (n) {
    case label1:
        code to be executed if n=label1;
        break;
    case label2:
        code to be executed if n=label2;
        break;
    case label3:
        code to be executed if n=label3;
        break;
    ...
    default:
        code to be executed if n is different from
all labels;
}
```

PHP Loops

In PHP, we have the following looping statements:

- **while** - loops through a block of code as long as the specified condition is true.
- **do...while** - loops through a block of code once, and then repeats the loop as long as the specified condition is true.
- **for** - loops through a block of code a specified number of times.
- **foreach** - loops through a block of code for each element in an array.

while Loop

The while loop executes a block of code as long as the specified condition is true.

Syntax

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

Ex:

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

do...while Loop

The do...while loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.

Syntax

```
do {
    code to be executed;
} while (condition is true);
```

Ex:

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

➢ for Loop

The for loop is used when you know in advance how many times the script should run.

Syntax

```
for (init counter; test counter;
increment counter) {
    code to be executed;
}
```

Ex:

```
<?php
for ($x = 0; $x <= 10; $x++) {
    echo "The number is: $x <br>";
}
?>
```

➢ foreach Loop

The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.

Syntax

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

Ex:

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

PHP Functions

- The real power of PHP comes from its functions; it has more than 1000 built-in functions.
- Besides the built-in PHP functions, Users can create their own functions.
- A user defined function declaration starts with the word "function":

Syntax

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

Note: A function name can start with a letter or underscore (not a number).

Ex:

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

➢ **Function Arguments**

Information can be passed to functions through arguments. An argument is just like a variable.

Ex:

```php
<?php
function familyName($fname, $year) {
    echo "$fname George. Born in $year <br>";
}
familyName("Hege", "1975");
familyName("Stale", "1978");
familyName("Kai Jim", "1983");
?>
```

➢ **Default Argument Value**

The following example shows how to use a default parameter. If we call the function setHeight() without arguments it takes the default value as argument:

Ex:

```php
<?php
function setHeight($minheight = 50) {
    echo "The height is : $minheight <br>";
}
setHeight(350);
setHeight(); // will use the default value of 50
setHeight(135);
?>
```

➢ **Functions - Returning values**

To let a function return a value, use the return statement:

Ex:

```
<?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);
?>
```

PHP Arrays

- An array stores multiple values in one single variable.
- In PHP, the array() function is used to create an array:

Ex:

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

- In PHP, there are three types of arrays:

- **Indexed arrays** - Arrays with a numeric index.
- **Associative arrays** - Arrays with named keys.
- **Multidimensional arrays** - Arrays containing one or more arrays.

➢ **PHP Indexed Arrays**

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");
```

2- the index can be assigned manually:

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

- **The Length of an Array**

The count() function is used to return the length (the number of elements) of an array.

Ex:

```
<?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 a for loop.

Ex:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
$arrlength = count($cars);
for($x = 0; $x < $arrlength; $x++) {
    echo $cars[$x];
    echo "<br>";
}
?>
```

- **PHP Associative Arrays**

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

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

Or

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

- The named keys can then be used in a script:

Ex:

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

- **Loop Through an Associative Array**
  To loop through and print all the values of an associative array, you could use a foreach loop.
  Ex:

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
foreach($age as $x => $x_value) {
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
?>
```

Note: Multidimensional Arrays will be explained in advanced chapters.