INTRODUCTION TO PHP

The Need for Dynamic Content

The web is no longer static, it's dynamic. As the information content of the web grows, so does the need to make web sites more dynamic.

Think of an e-shop that has 1000 products. The owner has to create 1000 web pages (one for each product) and whenever anything changes, the owner has to change all those pages. Wouldn't it be easier to have only one page that created and served the content on the fly from the information about the products stored in a database, depending on the client request?

Nowadays sites have to change constantly and provide up-to-date news, information, stock prices, and customized pages. **PHP** and **SQL** are two ways to make your site dynamic.

In this tutorial we will learn some basic facts about PHP.

What is PHP?

- **PHP stands for PHP: Hypertext Preprocessor**

  The name PHP is a recursive acronym that stands for PHP: Hypertext Preprocessor. It began life called PHP/FI, the "FI" part standing for Forms Interpreter. Though the name was shortened a while back, one of PHP's most powerful features is how easy it becomes to process data submitted in HTML forms.

- **PHP supports many databases**

  PHP can also talk to various database systems, giving you the ability to generate a web page based on a SQL query. For example, you could enter a search keyword into a form field on a web page, query a database with this value, and produce a page of matching results.

- **PHP is a server-side scripting language/PHP scripts are executed on the server**

  The beauty of PHP is that it does not rely on the web browser at all; your script will run the same way whatever browser you use. PHP slots into your web server and processes instructions contained in a web page before that page is sent through to your web browser. Certain elements of the page can therefore be generated on-the-fly so that the page changes each time it is loaded. Any PHP code found in the page is executed and replaced by the output it produces before the web page is sent to the browser. The usual web server configuration is that *somefile.php* will be interpreted by PHP, whereas *somefile.html* will be passed straight through to the web browser, without PHP getting involved.

- **PHP is free and open source software**

Where to Start?

- To get access to a web server with PHP support, you can:
    - ✓ Install Apache or IIS on your own server machine, install PHP and MySQL or
    - ✓ Find a web hosting plan with PHP and MySQL support

- You can find a comprehensive installation guide about PHP at http://www.php.net/manual/en/install.php

- You can find a list of PHP editors at http://en.wikipedia.org/wiki/List_of_PHP_editors . The text editors which come with your OS is quite OK.

There are some excellent all-in-one Windows distributions that contain Apache, PHP, MySQL and other applications in a single installation file, e.g. XAMPP WampServer and Web.Developer. There is nothing wrong with using these packages, although manually installing Apache and PHP will help you learn more about the system and its configuration options.

If you are running PHP from your local PC, PHP code in a script will be executed only if it is accessed through a web server installed in localhost that has the PHP module enabled. If you open a local script directly in the web browser for instance, by double-clicking or dragging and dropping the file into the browser it will be treated as HTML only.

Basic PHP Syntax

A PHP scripting block always starts with **<?php** and ends with **?>**. A PHP scripting block can be placed anywhere in the document.

```
<?php
?>
```

A PHP file normally contains HTML tags, just like an HTML file, and some PHP scripting code. When PHP parses a file, it looks for opening and closing tags, which tell PHP to start and stop interpreting the code between them. Everything outside of a pair of opening and closing tags is ignored by the PHP parser.

Below, we have an example of a simple PHP script which sends the text "Hello World" to the browser:

```
<html>
<body>
<?php
echo "Hello World";//outputs "Hello World"
?>
</body>
</html>
```

Comments in PHP

In PHP, we use // to make a single-line comment or /* and */ to make a large comment block.

```
<html>
<body>
<?php
//This is a comment
/*
This is
a comment
block
*/
?>
</body>
</html>
```

Variables in PHP

Variables in PHP are represented by a dollar sign followed by the name of the variable. The variable name is case-sensitive. A valid variable name starts with a letter or underscore, followed by any number of letters, numbers, or underscores.

```
<?php
$var = 'Bob';
$Var = 'Joe';
echo "$var, $Var";      // outputs "Bob, Joe"

$4site = 'not yet';     // invalid; starts with a number
$_4site = 'not yet';    // valid; starts with an underscore
$täyte = 'mansikka';    // valid; 'ä' is (Extended) ASCII 228.
?>
```

In PHP, a variable does not need to be declared before adding a value to it. In the example above, you see that you do not have to tell PHP which data type the variable is. PHP automatically converts the variable to the correct data type, depending on its value.

By default, variables are always assigned by value. That is to say, when you assign an expression to a variable, the entire value of the original expression is copied into the destination variable.

To assign by reference, simply prepend an ampersand (&) to the beginning of the variable which is being assigned (the source variable). For instance, the following code snippet outputs '*My name is Bob*' twice:

```
<?php
$foo = 'Bob';                // Assign the value 'Bob' to $foo
$bar = &$foo;                // Reference $foo via $bar.
$bar = "My name is $bar";    // Alter $bar...
echo $bar;
echo $foo;                   // $foo is altered too.
?>
```

String Variables in PHP

String variables are used for values that contain characters.After we create a string we can manipulate it. A string can be used directly in a function or it can be stored in a variable.

Below, the PHP script assigns the text "Hello World" to a string variable called $txt:

```
<?php
$txt="Hello World";
echo $txt; //outputs "Hello World"
?>
```

Concatenation operator

There is only one string operator in PHP. The concatenation operator (.) is used to put two string values together. To concatenate two string variables together, use the concatenation operator:

```
<?php
$txt1="Hello World!";
$txt2="What a nice day!";
echo $txt1 . " " .$txt2 //outputs "Hello World! What a nice day!"
```

String Functions in PHP

String functions allow you to manipulate strings. They are part of the PHP core. There is no installation needed to use these functions. For a complete reference of all string functions, go to http://www.w3schools.com/php/php_ref_string.asp

Flow Control in PHP

Conditional statements

In PHP we have the following conditional statements:

- **if statement** - use this statement to execute some code only if a specified condition is true
- **if...else statement** - use this statement to execute some code if a condition is true and another code if the condition is false
- **if...elseif....else statement** - use this statement to select one of several blocks of code to be executed
- **switch statement** - use this statement to select one of many blocks of code to be executed

If statement

Syntax

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

If...else statement

Syntax

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

If...elseif....else statement

Syntax

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

Switch statement

Syntax

```
switch (n)
{
case label1:
  code to be executed if n=label1;
  break;
case label2:
  code to be executed if n=label2;
  break;
default:
  code to be executed if n is different from both label1 and label2;
}
```

Loops

In PHP, we have the following looping statements:

- **while** - loops through a block of code while a specified condition is true
- **do...while** - loops through a block of code once, and then repeats the loop as long as a 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

Syntax

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

do...while statement

Syntax

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

for loop

Syntax

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

Parameters:

- *init*: Mostly used to set a counter (but can be any code to be executed once at the beginning of the loop)
- *condition*: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- *increment*: Mostly used to increment a counter (but can be any code to be executed at the end of the loop)

foreach loop

Syntax

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

Arrays in PHP

In PHP, there are three kind of arrays:

- **Numeric array** - An array with a numeric index
- **Associative array** - An array where each ID key is associated with a value
- **Multidimensional array** - An array containing one or more arrays

Numeric Arrays

A numeric array stores each array element with a numeric index. There are two methods to create a numeric array.

1. In the following example the index are automatically assigned (the index starts at 0):

```
$cars=array("Saab","Volvo","BMW","Toyota");
```

2. In the following example we assign the index manually:

```
$cars[0]="Saab";
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
```

Associative Arrays

An associative array, each ID key is associated with a value. When storing data about specific named values, a numerical array is not always the best way to do it. With associative arrays we can use the values as keys and assign values to them.

In this example we use an array to assign ages to the different persons:

```
$ages = array("Peter"=>32, "Quagmire"=>30, "Joe"=>34);
```

This example is the same as example above, but shows a different way of creating the array:

```
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";
```

Multidimensional Arrays

In a multidimensional array, each element in the main array can also be an array. And each element in the sub-array can be an array, and so on.

In this example we create a multidimensional array, with automatically assigned ID keys:

```
$families = array (
 "Griffin"=>array ( "Peter", "Lois", "Megan" ),
 "Quagmire"=>array ( "Glenn" ,”Aristotle”, “Fox” ),
 "Brown"=>array ( "Cleveland", "Loretta", "Junior" )
 );
```

For a complete reference of all array functions, go to
http://www.w3schools.com/php/php_ref_array.asp

References

- http://www.w3schools.com/php/
- http://www.php.net/manual/en/