What is PHP?

PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages.

PHP is an acronym for "PHP: Hypertext Preprocessor"

PHP is a widely-used, open source scripting language

PHP scripts are executed on the server

PHP is free to download and use

PHP is a widely-used, free, and efficient alternative to competitors such as Microsoft's ASP.

What is a PHP File?

PHP files can contain text, HTML, CSS, JavaScript, and PHP code

PHP code is executed on the server, and the result is returned to the browser as plain HTML

PHP files have extension ".php"

Example

```
<!DOCTYPE html>
<?php
echo "My first PHP script!";
?>
</body>
</html
```

What Can PHP Do?

PHP can generate dynamic page content

PHP can create, open, read, write, delete, and close files on the server

PHP can collect form data

PHP can send and receive cookies

PHP can add, delete, modify data in your database

PHP can be used to control user-access

PHP can encrypt data.

Why PHP?

PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)

PHP is compatible with almost all servers used today (Apache, IIS, etc.)

PHP supports a wide range of databases

PHP is free. Download it from the official PHP resource: www.php.net

PHP is easy to learn and runs efficiently on the server side.

Set Up PHP on Your Own PC

However, if your server does not support PHP, you must:

install a web server

install PHP

install a database, such as MySQL

The official PHP website (PHP.net) has installation instructions for PHP: http://php.net/manual/en/install.php

Basic PHP Syntax

A PHP script starts with <?php and ends with ?>

The default file extension for PHP files is ".php".

A PHP file normally contains HTML tags, and some PHP scripting code.

uses a built-in PHP function "echo" to output the text "Hello World!" on a web page:

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

Output.

My first PHP page

Hello World

PHP Case Sensitivity

In PHP, NO keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are case-sensitive.

However; all variable names are case-sensitive.

Example

```
<!DOCTYPE html>
<html>
<body>
<?php
$color = "red";
echo "My car is " . $color . "<br>";
ECHO "My bike is " . $color . "<br>"
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?>
</body>
</html>
```

Output

My car is red

My bike is red

Comments in PHP

Syntax for single line comment:

// Single line comment

Single line comment

Syntax for multiple line comments

/* Multiple lines

--- */

PHP Variables

In PHP, a variable starts with the $ sign, followed by the name of the variable.

It is created the moment you first assign a value to it.

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)

```
<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
?>
echo $txt;
echo "<br>";
echo $x;
echo "<br>";
echo $y;
?>
</body>
</html>
```

Output

Hello world!

5

10.5

Global and Local Scope

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

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

```
<!DOCTYPE html>
<html>
<body>
<?php
$y=8;
function myTest() {
 $x = 5;
 echo "<p>Variable x inside function is: $x</p>";
}
myTest();
echo "<p>Variable x outside function is: $x</p>";
echo "<p>Variable y outside function is: $y</p>";
?>
</body>
</html
```

output

Variable x inside function is: 5

Variable x outside function is:

Variable y outside function is: 8

The **global keyword** is used to access a global variable from within a function.

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

```
function myTest() {
   global $x, $y;
   $y = $x + $y;
}
myTest();
echo $y;
?>
```

outputs -15

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
?>
```

PHP The static Keyword

Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job.

To do this, use the static keyword when you first declare the variable:

Example

```
<?php
function myTest() {
```

```
  static $x = 0;
  echo $x;
  $x++;
}
myTest();
myTest();
myTest();
?>
```

PHP Conditional Statements

Very often when you write code, you want to perform different actions for different conditions. You can use conditional statements in your code to do this.

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;
}
```

Example

Output "Have a good day!" if the current time (HOUR) is less than 20:

```
<?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;
}
```

Example

```
<?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 first condition is false and this condition is true;
} else {
   code to be executed if all conditions are false;
}
```

Example

Output "Have a good morning!" if the current time is less than 10, and "Have a good day!" if the current time is less than 20. Otherwise it will output "Have a good night!":

```
<?php
$t = date("H");
if ($t < "10") {
   echo "Have a good morning!";
} elseif ($t < "20") {
   echo "Have a good day!";
} else {
   echo "Have a good night!";
}
?>
```

switch Statement

The switch statement is used to perform different actions based on different conditions.

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;
}
```

Example

```
<?php
$favcolor = "red";

switch ($favcolor) {
  case "red":
    echo "Your favorite color is red!";
    break;
  case "blue":
    echo "Your favorite color is blue!";
    break;
  case "green":
    echo "Your favorite color is green!";
    break;
  default:
```

```
    echo "Your favorite color is neither red, blue, nor green!";
}
?>
```

PHP Loops

Loops are used to execute the same block of code again and again, as long as a certain condition is true.

In PHP, we have the following loop types:

- 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;
}
```

Example

The example below displays the numbers from 1 to 5:

```
<?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);
```

Example

```
<?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 for each iteration;
```

```
}
```

Example

```
<?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;
}
```

Example

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