# PHP Tutorial

## PHPTutorial

**PHP is a powerful server-side scripting language for creating dynamic and interactive websites.**

**PHP is the widely-used, free, and efficient alternative to competitors such as Microsoft's ASP. PHP is perfectly suited for Web development and can be embedded directly into the HTML code.**

**The PHP syntax is very similar to Perl and C. PHP is often used together with Apache (web server) on various operating systems. It also supports ISAPI and can be used with Microsoft's IIS on Windows.**

**Start learning PHP now!**

## PHPReferences

At W3Schools you will find complete references of all PHP functions:

Zipfunctions

# IntroductiontoPHP

PHP is a server-side scripting language.

## What You Should Already Know

Before you continue you should have a basic understanding of the following:

HTML
Some scripting knowledge

If you want to study these subjects first, find the tutorials on our Homepage.

## What is PHP?

PHP stands for **P**HP: **H**ypertext **P**reprocessor
PHP is a server-side scripting language, like ASP
PHP scripts are executed on the server
PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.)
PHP is an open source software
PHP is free to download and use

## What is a PHP File?

PHP files can contain text, HTML tags and scripts
PHP files are returned to the browser as plain HTML
PHP files have a file extension of ".php", ".php3", or ".phtml"

## What is MySQL?

MySQL is a database server
MySQL is ideal for both small and large applications
MySQL supports standard SQL
MySQL compiles on a number of platforms
MySQL is free to download and use

## PHP + MySQL

PHP combined with MySQL are cross-platform (you can develop in Windows and serve on a Unix platform)

## Why PHP?

PHP runs on different platforms (Windows, Linux, Unix, etc.)
PHP is compatible with almost all servers used today (Apache, IIS, etc.)
PHP is FREE to download from the official PHP resource: www.php.net

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

## Where to Start?

To get access to a web server with PHP support, you can:

Install Apache (or IIS) on your own server, install PHP, and MySQL
Or find a web hosting plan with PHP and MySQL support

# PHPInstallation

## What do You Need?

If your server supports PHP you don't need to do anything. Just create some .php files in your web directory, and the server will parse them for you. Because it is free, most web hosts offer PHP support.

However, if your server does not support PHP, you must install PHP.

Here is a link to a good tutorial from PHP.net on how to install PHP5:
http://www.php.net/manual/en/install.php

## Download PHP

Download PHP for free here: http://www.php.net/downloads.php

## Download MySQL Database

Download MySQL for free here: http://www.mysql.com/downloads/index.html

## Download Apache Server

Download Apache for free here: http://httpd.apache.org/download.cgi

# PHPSyntax

PHP code is executed on the server, and the plain HTML result is sent to the browser.

## 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.

On servers with shorthand support enabled you can start a scripting block with <? and end with ?>.

For maximum compatibility, we recommend that you use the standard form (<?php) rather than the shorthand form.

```
<?php
?>
```

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

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";
?>
</body>
</html>
```

Each code line in PHP must end with a semicolon. The semicolon is a separator and is used to distinguish one set of instructions from another.

There are two basic statements to output text with PHP: **echo** and **print**. In the example above we have used the echo statement to output the text "Hello World".

**Note:** The file must have the .php extension. If the file has a .html extension, the PHP code will not be executed.

## 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>
```

# PHPVariables

Variables are used for storing values, such as numbers, strings or function results, so that they can be used many times in a script.

## Variables in PHP

Variables are used for storing a values, like text strings, numbers or arrays.

When a variable is set it can be used over and over again in your script

All variables in PHP start with a $ sign symbol.

The correct way of setting a variable in PHP:

```
$var_name = value;
```

New PHP programmers often forget the $ sign at the beginning of the variable. In that case it will not work.

Let's try creating a variable with a string, and a variable with a number:

```
<?php
$txt = "Hello World!";
$number = 16;
?>
```

## PHP is a Loosely Typed Language

In PHP a variable does not need to be declared before being set.

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 how they are set.

In a strongly typed programming language, you have to declare (define) the type and name of the variable before using it.

In PHP the variable is declared automatically when you use it.

## Variable Naming Rules

A variable name must start with a letter or an underscore "_"
A variable name can only contain alpha-numeric characters and underscores (a-z, A-Z, 0-9, and _ )
A variable name should not contain spaces. If a variable name is more than one word, it should be separated with underscore ($my_string), or with capitalization ($myString)

# PHPString

A string variable is used to store and manipulate a piece of text.

## Strings in PHP

String variables are used for values that contains character strings.

In this tutorial we are going to look at some of the most common functions and operators used to manipulate strings in PHP.

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 string "Hello World" to a string variable called $txt:

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

The output of the code above will be:

```
Hello World
```

Now, lets try to use some different functions and operators to manipulate our string.

## The Concatenation Operator

There is only one string operator in PHP.

The concatenation operator (.) is used to put two string values together.

To concatenate two variables together, use the dot (.) operator:

```
<?php
$txt1="Hello World";
$txt2="1234";
echo $txt1 . " " . $txt2;
?>
```

The output of the code above will be:

```
Hello World 1234
```

If we look at the code above you see that we used the concatenation operator two times. This is because we had to insert a third string.

Between the two string variables we added a string with a single character, an empty space, to separate the two variables.

## Using the strlen() function

The strlen() function is used to find the length of a string.

Let's find the length of our string "Hello world!":

```
<?php
echo strlen("Hello world!");
```

```
?>
```

The output of the code above will be:

```
12
```

The length of a string is often used in loops or other functions, when it is important to know when the string ends. (i.e. in a loop, we would want to stop the loop after the last character in the string)

## Using the strpos() function

The strpos() function is used to search for a string or character within a string.

If a match is found in the string, this function will return the position of the first match. If no match is found, it will return FALSE.

Let's see if we can find the string "world" in our string:

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

The output of the code above will be:

```
6
```

As you see the position of the string "world" in our string is position 6. The reason that it is 6, and not 7, is that the first position in the string is 0, and not 1.

## Complete PHP String Reference

For a complete reference of all string functions, go to our completePHPString Reference.
The reference contains a brief description and examples of use for each function!

## PHPOperators

Operators are used to operate on values.

## PHP Operators

This section lists the different operators used in PHP.

### Arithmetic Operators

| Operator | Description | Example | Result |
|---|---|---|---|
| + | Addition | x=2<br>x+2 | 4 |
| - | Subtraction | x=2<br>5-x | 3 |
| * | Multiplication | x=4<br>x*5 | 20 |
| / | Division | 15/5<br>5/2 | 3<br>2.5 |
| % | Modulus (division remainder) | 5%2<br>10%8<br>10%2 | 1<br>2<br>0 |
| ++ | Increment | x=5<br>x++ | x=6 |
| -- | Decrement | x=5<br>x-- | x=4 |

### Assignment Operators

| Operator | Example | Is The Same As |
|---|---|---|
| = | x=y | x=y |
| += | x+=y | x=x+y |
| -= | x-=y | x=x-y |
| *= | x*=y | x=x*y |
| /= | x/=y | x=x/y |
| .= | x.=y | x=x.y (gabungin jadi xy; bisa buat generate key) |
| %= | x%=y | x=x%y |

### Comparison Operators

| Operator | Description | Example |
|---|---|---|
| == | is equal to | 5==8 returns false |
| != | is not equal | 5!=8 returns true |
| > | is greater than | 5>8 returns false |
| < | is less than | 5<8 returns true |
| >= | is greater than or equal to | 5>=8 returns false |
| <= | is less than or equal to | 5<=8 returns true |

**Logical Operators**

| Operator | Description | Example |
|---|---|---|
| && | and | x=6<br>y=3<br><br>(x < 10 && y > 1) returns true |
| \|\| | or | x=6<br>y=3<br><br>(x==5 \|\| y==5) returns false |
| ! | not | x=6<br>y=3<br><br>!(x==y) returns true |

# PHPIf...ElseStatements

The if, elseif and else statements in PHP are used to perform different actions based on different conditions.

## Conditional Statements

Very often when you write code, you want to perform different actions for different decisions.

You can use conditional statements in your code to do this.

**if...else statement** - use this statement if you want to execute a set of code when a condition is true and another if the condition is not true

**elseif statement** - is used with the if...else statement to execute a set of code if **one** of several condition are true

# The If...Else Statement

If you want to execute some code if a condition is true and another code if a condition is false, use the if....else statement.

### Syntax

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

### Example

The following example will output "Have a nice weekend!" if the current day is Friday, otherwise it will output "Have a nice day!":

```
<html>
<body>
<?php
$d=date("D");
if ($d=="Fri")
  echo "Have a nice weekend!";
else
  echo "Have a nice day!";
?>
</body>
</html>
```

If more than one line should be executed if a condition is true/false, the lines should be enclosed within curly braces:

```
<html>
<body>
<?php
$d=date("D");
if ($d=="Fri")
  {
  echo "Hello!<br />";
  echo "Have a nice weekend!";
  echo "See you on Monday!";
  }
?>
</body>
```

```
</html>
```

## The ElseIf Statement

If you want to execute some code if one of several conditions are true use the elseif statement

### Syntax

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

### Example

The following example will output "Have a nice weekend!" if the current day is Friday, and "Have a nice Sunday!" if the current day is Sunday. Otherwise it will output "Have a nice day!":

```
<html>
<body>
<?php
$d=date("D");
if ($d=="Fri")
  echo "Have a nice weekend!";
elseif ($d=="Sun")
  echo "Have a nice Sunday!";
else
  echo "Have a nice day!";
?>
</body>
</html>
```

# PHPSwitchStatement

The Switch statement in PHP is used to perform one of several different actions based on one of several different conditions.

## The Switch Statement

If you want to select one of many blocks of code to be executed, use the Switch statement.
The switch statement is used to avoid long blocks of if..elseif..else code.

### Syntax

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

### Example

This is how it works:

- A single expression (most often a variable) is evaluated once
- The value of the expression is compared with the values for each case in the structure
- If there is a match, the code associated with that case is executed
- After a code is executed, **break** is used to stop the code from running into the next case
- The default statement is used if none of the cases are true

```
<html>
<body>
<?php
switch ($x)
{
case 1:
  echo "Number 1";
  break;
case 2:
```

```
  echo "Number 2";
  break;
case 3:
  echo "Number 3";
  break;
default:
  echo "No number between 1 and 3";
}
?>
</body>
</html>
```

# PHPArrays

An array can store one or more values in a single variable name.

## What is an array?

When working with PHP, sooner or later, you might want to create many similar variables.

Instead of having many similar variables, you can store the data as elements in an array.

Each element in the array has its own ID so that it can be easily accessed.

There are three different kind of arrays:

**Numeric array** - An array with a numeric ID key
**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 element with a numeric ID key.

There are different ways to create a numeric array.

### Example 1

In this example the ID key is automatically assigned:

```
$names = array("Peter","Quagmire","Joe");
```

### Example 2

In this example we assign the ID key manually:

```
$names[0] = "Peter";
$names[1] = "Quagmire";
$names[2] = "Joe";
```

The ID keys can be used in a script:

```
<?php
$names[0] = "Peter";
$names[1] = "Quagmire";
$names[2] = "Joe";
echo $names[1] . " and " . $names[2] .
" are ". $names[0] . "'s neighbors";
?>
```

The code above will output:

```
Quagmire and Joe are Peter's neighbors
```

## 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.

### Example 1

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

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

### Example 2

This example is the same as example 1, but shows a different way of creating the array:

```
$ages['Peter'] = "32";
```

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

The ID keys can be used in a script:

```
<?php
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";
echo "Peter is " . $ages['Peter'] . " years old.";
?>
```

The code above will output:

```
Peter is 32 years old.
```

## 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.

### Example

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

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

The array above would look like this if written to the output:

```
Array
(
[Griffin] => Array
   (
   [0] => Peter
   [1] => Lois
   [2] => Megan
   )
[Quagmire] => Array
   (
   [0] => Glenn
   )
[Brown] => Array
   (
   [0] => Cleveland
   [1] => Loretta
   [2] => Junior
   )
)
```

### Example 2

Lets try displaying a single value from the array above:

```
echo "Is " . $families['Griffin'][2] .
" a part of the Griffin family?";
```

The code above will output:

```
Is Megan a part of the Griffin family?
```

# PHPLooping

Looping statements in PHP are used to execute the same block of code a specified number of times.

## Looping

Very often when you write code, you want the same block of code to run a number of times. You can use looping statements in your code to perform this.

In PHP we have the following looping statements:

**while** - loops through a block of code if and as long as a specified condition is true

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

## The while Statement

The while statement will execute a block of code **if and as long as** a condition is true.

### Syntax

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

### Example

The following example demonstrates a loop that will continue to run as long as the variable i is less than, or equal to 5. i will increase by 1 each time the loop runs:

```
<html>
<body>
<?php
$i=1;
while($i<=5)
  {
  echo "The number is " . $i . "<br />";
  $i++;
  }
?>
</body>
</html>
```

## The do...while Statement

The do...while statement will execute a block of code **at least once** - it then will repeat the loop **as long as** a condition is true.

### Syntax

```
do
{
code to be executed;
```

```
}
while (condition);
```

### Example

The following example will increment the value of i at least once, and it will continue incrementing the variable i as long as it has a value of less than 5:

```
<html>
<body>
<?php
$i=0;
do
  {
  $i++;
  echo "The number is " . $i . "<br />";
  }
while ($i<5);
?>
</body>
</html>
```

## The for Statement

The for statement is the most advanced of the loops in PHP.

In it's simplest form, the for statement is used when you know how many times you want to execute a statement or a list of statements.

### Syntax

```
for (init; cond; incr)
{
  code to be executed;
}
```

Parameters:

**init**: Is mostly used to set a counter, but can be any code to be executed once at the beginning of the loop statement.
**cond**: Is evaluated at beginning of each loop iteration. If the condition evaluates to TRUE, the loop continues and the code executes. If it evaluates to FALSE, the execution of the loop ends.
**incr**: Is mostly used to increment a counter, but can be any code to be executed at the end of each loop.

**Note:** Each of the parameters can be empty or have multiple expressions separated by commas.

**cond:** All expressions separated by a comma are evaluated but the result is taken from the last part. This parameter being empty means the loop should be run indefinitely. This is useful when using a conditional break statement inside the loop for ending the loop.

### Example

The following example prints the text "Hello World!" five times:

```
<html>
<body>
<?php
for ($i=1; $i<=5; $i++)
{
  echo "Hello World!<br />";
}
?>
</body>
</html>
```

## The foreach Statement

The foreach statement is used to loop through arrays.

For every loop, the value of the current array element is assigned to $value (and the array pointer is moved by one) - so on the next loop, you'll be looking at the next element.

### Syntax

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

### Example

The following example demonstrates a loop that will print the values of the given array:

```
<html>
<body>
<?php
```

```
$arr=array("one", "two", "three");
foreach ($arr as $value)
{
  echo "Value: " . $value . "<br />";
}
?>
</body>
</html>
```

# PHPFunctions

The real power of PHP comes from its functions.
In PHP - there are more than 700 built-in functions available.

## PHP Functions

In this tutorial we will show you how to create your own functions.
For a reference and examples of the built-in functions, please visit our PHP Reference.

## Create a PHP Function

A function is a block of code that can be executed whenever we need it.

Creating PHP functions:

All functions start with the word "function()"
Name the function - It should be possible to understand what the function does by its name. The name can start with a letter or underscore (not a number)
Add a "{" - The function code starts after the opening curly brace
Insert the function code
Add a "}" - The function is finished by a closing curly brace

### Example

A simple function that writes my name when it is called:

```
<html>
<body>
<?php
function writeMyName()
```

```
  {
  echo "Kai Jim Refsnes";
  }
writeMyName();
?>
</body>
</html>
```

## Use a PHP Function

Now we will use the function in a PHP script:

```
<html>
<body>
<?php
function writeMyName()
  {
  echo "Kai Jim Refsnes";
  }
echo "Hello world!<br />";
echo "My name is ";
writeMyName();
echo ".<br />That's right, ";
writeMyName();
echo " is my name.";
?>
</body>
</html>
```

The output of the code above will be:

```
Hello world!
My name is Kai Jim Refsnes.
That's right, Kai Jim Refsnes is my name.
```

## PHP Functions - Adding parameters

Our first function (writeMyName()) is a very simple function. It only writes a static string.

To add more functionality to a function, we can add parameters. A parameter is just like a variable.

You may have noticed the parentheses after the function name, like: writeMyName(). The parameters are specified inside the parentheses.

## Example 1

The following example will write different first names, but the same last name:

```
<html>
<body>
<?php
function writeMyName($fname)
  {
  echo $fname . " Refsnes.<br />";
  }
echo "My name is ";
writeMyName("Kai Jim");
echo "My name is ";
writeMyName("Hege");
echo "My name is ";
writeMyName("Stale");
?>
</body>
</html>
```

The output of the code above will be:

```
My name is Kai Jim Refsnes.
My name is Hege Refsnes.
My name is Stale Refsnes.
```

## Example 2

The following function has two parameters:

```
<html>
<body>
<?php
function writeMyName($fname,$punctuation)
  {
  echo $fname . " Refsnes" . $punctuation . "<br />";
  }
echo "My name is ";
writeMyName("Kai Jim",".");
echo "My name is ";
writeMyName("Hege","!");
echo "My name is ";
writeMyName("Ståle","...");
?>
</body>
</html>
```

The output of the code above will be:

```
My name is Kai Jim Refsnes.
My name is Hege Refsnes!
My name is Ståle Refsnes...
```

## PHP Functions - Return values

Functions can also be used to return values.

### Example

```
<html>
<body>
<?php
function add($x,$y)
  {
  $total = $x + $y;
  return $total;
  }
echo "1 + 16 = " . add(1,16);
?>
</body>
</html>
```

The output of the code above will be:

```
1 + 16 = 17
```