What is PHP

PHP stands for Hypertext Preprocessor.

PHP is an interpreted language, i.e., there is no need for compilation.

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 server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.

It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.

What You Should Already Know

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

* **HTML**
* **CSS**
* **JavaScript**

What is a PHP File?

* PHP files can contain text, HTML, CSS, JavaScript, and PHP code
* PHP code are executed on the server, and the result is returned to the browser as plain HTML
* PHP files have extension ".php"

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

With PHP you are not limited to output HTML. You can output images, PDF files, and even Flash movies. You can also output any text, such as XHTML and XML.

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](http://www.php.net/)
* PHP is easy to learn and runs efficiently on the server side

## Basic PHP Syntax

A PHP script can be placed anywhere in the document.

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

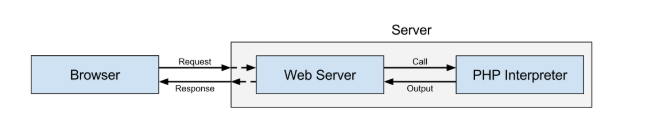
<?php  
// PHP code goes here  
?>

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

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

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

**Note:** PHP statements end with a semicolon (;).



As you can see in the diagram above, we always start with a browser making a request for a web page. This request is going to hit the web server. The web server will then analyze it and determine what to do with it.

If the web server determines that the request is for a PHP file (often index.php), it’ll pass that file to the PHP interpreter. The PHP interpreter will read the PHP file, parse it (and other included files) and then execute it. Once the PHP interpreter finishes executing the PHP file, it’ll return an output. The web server will take that output and send it back as a response to the browser.

# Install PHP

To install PHP, we will suggest you to install AMP (Apache, MySQL, PHP) software stack. It is available for all operating systems. There are many AMP options available in the market that are given below:

* **WAMP** for Windows
* **LAMP** for Linux
* **MAMP** for Mac
* **SAMP** for Solaris
* **FAMP** for FreeBSD
* **XAMPP** (Cross, Apache, MySQL, PHP, Perl) for Cross Platform: It includes some other components too such as FileZilla, OpenSSL, Webalizer, Mercury Mail etc.

If you are on Windows and don't want Perl and other features of XAMPP, you should go for WAMP. In a similar way, you may use LAMP for Linux and MAMP for Macintosh.

# PHP Variables

A variable in PHP is a name of memory location that holds data. A variable is a temporary storage that is used to store data temporarily.

In PHP, a variable is declared using $ sign followed by variable name.

Syntax of declaring a variable in PHP is given below:

$variablename=value;

## PHP Variable: Declaring string, integer and float

**<?php**

$str="hello string";

$x=200;

$y=44.6;

echo "string is: $str **<br/>**";

echo "integer is: $x **<br/>**";

echo "float is: $y **<br/>**";

**?>**

**<?php**

$x=5;

$y=6;

$z=$x+$y;

echo $z;

**?>**

PHP Variables

Rules for PHP variables:

* A variable starts with the $ sign, followed by the name of the variable
* 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)

Remember that PHP variable names are case-sensitive!

## Comments in PHP

// This is a single-line comment  
  
# This is also a single-line comment  
  
/\*  
This is a multiple-lines comment block  
that spans over multiple  
lines  
\*/

PHP Operators

Operators are used to perform operations on variables and values.

PHP divides the operators in the following groups:

* Arithmetic operators
* Assignment operators
* Comparison operators
* Increment/Decrement operators
* Logical operators
* String operators
* Array operators

# PHP If Else

PHP if else statement is used to test condition. There are various ways to use if statement in PHP.

* if
* if-else
* if-else-if
* nested if

## PHP If Statement

PHP if statement is executed if condition is true.

**Syntax**

**if**(condition){

//code to be executed

}

<?php

$num=12;

**if**($num<100){

echo "$num is less than 100";

}

?>

## PHP - The 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;*}

The example below will output "Have a good day!" if the current time is less than 20, and "Have a good night!" otherwise:

### Example

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

## PHP - The 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;*}

### Example

<?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!";  
}  
?>

## PHP - The switch Statement

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

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;*  
}

### 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

Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal code-lines in a script, we can use loops to perform a task like this.

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

## The PHP 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*;  
}

The example below first sets a variable $x to 1 ($x = 1). Then, the while loop will continue to run as long as $x is less than, or equal to 5 ($x <= 5). $x will increase by 1 each time the loop runs ($x++):

### Example

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

## The PHP 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*);

The example below first sets a variable $x to 1 ($x = 1). Then, the do while loop will write some output, and then increment the variable $x with 1. Then the condition is checked (is $x less than, or equal to 5?), and the loop will continue to run as long as $x is less than, or equal to 5:

### Example

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

Notice that in a do while loop the condition is tested AFTER executing the statements within the loop. This means that the do while loop would execute its statements at least once, even if the condition is false the first time.

The example below sets the $x variable to 6, then it runs the loop, **and then the condition is checked**:

### Example

<?php   
$x = 6;  
  
do {  
    echo "The number is: $x <br>";  
    $x++;  
} while ($x <= 5);  
?>

PHP for loops execute a block of code a specified number of times.

## The PHP 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;*  
}

Parameters:

* *init counter*: Initialize the loop counter value
* *test counter*: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
* *increment counter*: Increases the loop counter value

The example below displays the numbers from 0 to 10:

### Example

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

## The PHP 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;*  
}

For every loop iteration, the value of the current array element is assigned to $value and the array pointer is moved by one, until it reaches the last array element.

The following example demonstrates a loop that will output the values of the given array ($colors):

### Example

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

The foreach loop statement

The foreach statement is used to loop through arrays. For each pass the value of the current array element is assigned to $value and the array pointer is moved by one and in the next pass next element will be processed.

### Syntax

foreach (*array* as *value*)

{

*code to be executed;*

}

**<html>**

**<body>**

**<?php**

**$array = array( 1, 2, 3, 4, 5);**

**foreach( $array as $value )**

**{**

**echo "Value is $value <br />";**

**}**

**?>**

**</body>**

**</html>**

# PHP - Arrays

There are three different kind of arrays and each array value is accessed using an ID c which is called array index.

* **Numeric array** − An array with a numeric index. Values are stored and accessed in linear fashion.
* **Associative array** − An array with strings as index. This stores element values in association with key values rather than in a strict linear index order.
* **Multidimensional array** − An array containing one or more arrays and values are accessed using multiple indices

## Numeric Array

These arrays can store numbers, strings and any object but their index will be represented by numbers. By default array index starts from zero.

### Example

Following is the example showing how to create and access numeric arrays.

Here we have used **array()** function to create array.

**<html>**

**<body>**

**<?php**

/\* First method to create array. \*/

**$numbers = array( 1, 2, 3, 4, 5);**

**foreach( $numbers as $value ) {**

**echo "Value is $value <br />";**

**}**

/\* Second method to create array. \*/

**$numbers[0] = "one";**

**$numbers[1] = "two";**

**$numbers[2] = "three";**

**$numbers[3] = "four";**

**$numbers[4] = "five";**

**foreach( $numbers as $value )**

**{**

**echo "Value is $value <br />";**

**}**

**?>**

**</body>**

**</html>**

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

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

Example

<?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, like this:

Example

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

## Associative Arrays

The associative arrays are very similar to numeric arrays in term of functionality but they are different in terms of their index. Associative array will have their index as string so that you can establish a strong association between key and values.

To store the salaries of employees in an array, a numerically indexed array would not be the best choice. Instead, we could use the employees names as the keys in our associative array, and the value would be their respective salary.

**<html>**

**<body>**

**<?php**

**/\* First method to associate create array. \*/**

**$salaries = array("Ajay" => 55000, "Sunil" => 19000, "Vijay" => 5000);**

**echo "Salary of Ajay is ". $salaries['Ajay'] . "<br />";**

**echo "Salary of Sunil is ". $salaries['Sunil']. "<br />";**

**echo "Salary of Vijay is ". $salaries['Vijay']. "<br />";**

**/\* Second method to create array. \*/**

**$salaries['Ajay'] = "high";**

**$salaries['Sunil'] = "medium";**

**$salaries['Vijay'] = "low";**

**echo "Salary of Ajay is ". $salaries['Ajay'] . "<br />";**

**echo "Salary of Sunil is ". $salaries['Sunil']. "<br />";**

**echo "Salary of Vijay is ". $salaries['Vijay']. "<br />";**

**?>**

**</body>**

**</html>**

**Salary of Ajay is 55000  
Salary of Sunil is 19000  
Salary of Vijay is 5000  
Salary of Ajay is high  
Salary of Sunil is medium  
Salary of Vijay is low**

**<html>**

**<body>**

**<?php**

**$salaries = array("Ajay" => 55000, "Sunil" => 19000, "Vijay" => 5000);**

**foreach($salaries as $x => $x\_value)**

**{**

**echo "Name=" . $x . ", Salary=" . $x\_value;**

**echo "<br>";**

**}**

**?>**

**</body>**

**</html>**

Name=Ajay, Salary=55000  
Name=Sunil, Salary=19000  
Name=Vijay, Salary=5000

**<html>**

**<body>**

**<?php**

**$salaries = array("Ajay" => 55000, "Sunil" => 19000, "Vijay" => 5000);**

**$keys = array\_keys($salaries);**

**$round = count($salaries);**

**for($i=0; $i < $round; ++$i) {**

**echo "name=". $keys[$i] . ' ' . "Salary=" . $salaries[$keys[$i]] .**

**"</br>";**

**}**

**?>**

**</body>**

**</html>**

name=Ajay Salary=55000  
name=Sunil Salary=19000  
name=Vijay Salary=5000

## Multidimensional Arrays

A multi-dimensional 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. Values in the multi-dimensional array are accessed using multiple index.

### Example

In this example we create a two dimensional array to store marks of three students in three subjects −

**<html>**

**<body>**

**<?php**

**$marks = array(**

**"Ajay" => array (**

**"physics" => 35,**

**"maths" => 30,**

**"chemistry" => 39**

**),**

**"Sunil" => array (**

**"physics" => 30,**

**"maths" => 32,**

**"chemistry" => 29**

**),**

**"Vijay" => array (**

**"physics" => 31,**

**"maths" => 22,**

**"chemistry" => 39**

**)**

**);**

**/\* Accessing multi-dimensional array values \*/**

**echo "Marks for Ajay in physics : " ;**

**echo $marks['Ajay']['physics'] . "<br />";**

**echo "Marks for Sunil in maths : ";**

**echo $marks['Sunil']['maths'] . "<br />";**

**echo "Marks for Vijay in chemistry : " ;**

**echo $marks['Vijay']['chemistry'] . "<br />";**

**?>**

**</body>**

**</html>**

This will produce the following result −

Marks for Ajay in physics : 35  
Marks for Sunil in maths : 32  
Marks for Vijay in chemistry : 39

**PHP Sorting Arrays**

**The elements in an array can be sorted in alphabetical or numerical order, descending or ascending.**

## PHP - Sort Functions For Arrays

**In this chapter, we will go through the following 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**
* **arsort() - sort associative arrays in descending order, according to the value**
* **krsort() - sort associative arrays in descending order, according to the key**

## Sort Array in Ascending Order - sort()

**The following example sorts the elements of the $cars array in ascending alphabetical order:**

### Example

**<html>**

**<body>**

**<?php**

**$cars=array("san","dan","tan","jan");**

**sort($cars);**

**$clength = count($cars);**

**for($x = 0; $x < $clength; $x++) {**

**echo $cars[$x];**

**echo "<br />";**

**}**

**?>**

**</body>**

**</html>**

**The following example sorts the elements of the $numbers array in ascending numerical order:**

**<!DOCTYPE html>  
<html>  
<body>  
  
<?php  
$numbers = array(4, 6, 2, 22, 11);  
sort($numbers);  
  
$arrlength = count($numbers);  
for($x = 0; $x < $arrlength; $x++) {  
    echo $numbers[$x];  
    echo "<br />";  
}  
  
</body>  
</html>**

## Sort Array in Descending Order - rsort()

**The following example sorts the elements of the $cars array in descending alphabetical order:**

### Example

**<html>**

**<body>**

**<?php**

**$cars=array("san","dan","tan","jan");**

**rsort($cars);**

**$clength = count($cars);**

**for($x = 0; $x < $clength; $x++) {**

**echo $cars[$x];**

**echo "<br />";**

**}**

**?>**

**</body>**

**</html>**

**The following example sorts the elements of the $numbers array in descending numerical order:**

### Example

**<html>  
<body>  
  
<?php  
$numbers = array(4, 6, 2, 22, 11);  
rsort($numbers);  
  
$arrlength = count($numbers);  
for($x = 0; $x < $arrlength; $x++) {  
    echo $numbers[$x];  
    echo "<br>";  
}  
?>  
  
</body>  
</html>**

## Sort Array (Ascending Order), According to Value - asort()

**The following example sorts an associative array in ascending order, according to the value:**

### Example

**<!DOCTYPE html>**

**<html>**

**<body>**

**<?php**

**$age = array("Peter"=>"45", "Ben"=>"67", "Joe"=>"43");**

**asort($age);**

**foreach($age as $x => $x\_value) {**

**echo "Key=" . $x . ", Value=" . $x\_value;**

**echo "<br>";**

**}**

**?>**

**</body>**

**</html>**

Key=Joe, Value=43  
Key=Peter, Value=45  
Key=Ben, Value=67

## Sort Array (Ascending Order), According to Key - ksort()

**The following example sorts an associative array in ascending order, according to the key:**

### Example

**<!DOCTYPE html>  
<html>  
<body>  
  
<?php  
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");  
ksort($age);  
  
foreach($age as $x => $x\_value) {  
    echo "Key=" . $x . ", Value=" . $x\_value;  
    echo "<br>";  
}  
?>  
  
</body>  
</html>**

## Key=Ben, Value=37 Key=Joe, Value=43 Key=Peter, Value=35

## Sort Array (Descending Order), According to Value - arsort()

**The following example sorts an associative array in descending order, according to the value:**

### Example

**<!DOCTYPE html>  
<html>  
<body>  
  
<?php  
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");  
arsort($age);  
  
foreach($age as $x => $x\_value) {  
    echo "Key=" . $x . ", Value=" . $x\_value;  
    echo "<br>";  
}  
?>  
  
</body>  
</html>**

## Sort Array (Descending Order), According to Key - krsort()

**The following example sorts an associative array in descending order, according to the key:**

### Example

**<!DOCTYPE html>  
<html>  
<body>  
  
<?php  
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");  
krsort($age);  
  
foreach($age as $x => $x\_value) {  
    echo "Key=" . $x . ", Value=" . $x\_value;  
    echo "<br>";  
}  
?>  
  
</body>  
</html>**