PHP Tutorial

❮ HomeNext ❯

Learn PHP

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

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

Start learning PHP now »

Easy Learning with "PHP Tryit"

With our online "PHP Tryit" editor, you can edit the PHP code, and click on a button to view the result.

ExampleGet your own PHP Server

<!DOCTYPE html>

<html>

<body>

**<?php**

echo "My first PHP script!";

**?>**

</body>

</html>

PHP References

W3Schools' PHP reference contains different categories of all PHP functions, keywords and constants, along with examples.

Array Calendar Date Directory Error Exception Filesystem Filter FTP JSON Keywords Libxml Mail Math Misc MySQLi Network Output RegEx SimpleXML Stream String Var Handling XML Parser Zip Timezones

# PHP Introduction

❮ PreviousNext ❯

PHP code is executed on the server.

## What You Should Already Know

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

* HTML
* CSS
* JavaScript

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

## What is PHP?

* 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 an amazing and popular language!**

It is powerful enough to be at the core of the biggest blogging system on the web (WordPress)!  
It is deep enough to run large social networks!  
It is also easy enough to be a beginner's first server side language!

## 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"

## 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 or PDF files. You can also output any text, such as XHTML and XML.

ADVERTISEMENT

## 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

## What's new in PHP 7

* PHP 7 is much faster than the previous popular stable release (PHP 5.6)
* PHP 7 has improved Error Handling
* PHP 7 supports stricter Type Declarations for function arguments
* PHP 7 supports new operators (like the spaceship operator: <=>)

PHP Installation

❮ PreviousNext ❯

What Do I Need?

To start using PHP, you can:

* Find a web host with PHP and MySQL support
* Install a web server on your own PC, and then install PHP and MySQL

Use a Web Host With PHP Support

If your server has activated support for PHP you do not need to do anything.

Just create some .php files, place them in your web directory, and the server will automatically parse them for you.

You do not need to compile anything or install any extra tools.

Because PHP is free, most web hosts offer PHP support.

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

PHP Online Compiler / Editor

With w3schools' online PHP compiler, you can edit PHP code, and view the result in your browser.

Run »

<?php  
$txt = "PHP";  
echo "I love $txt!";  
?>

I love PHP!

Try it Yourself »

Click on the "Try it Yourself" button to see how it works.

PHP Version

To check your php version you can use the phpversion() function:

ExampleGet your own PHP Server

Display the PHP version:

echo phpversion();

Try it Yourself »

# PHP Syntax

❮ PreviousNext ❯

A PHP script is executed on the server, and the plain HTML result is sent back to the browser.

## 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.

Below, we have an example of a simple PHP file, with a PHP script that uses a built-in PHP function "echo" to output the text "Hello World!" on a web page:

### ExampleGet your own PHP Server

A simple .php file with both HTML code and PHP code:

<!DOCTYPE html>

<html>

<body>

<h1>My first PHP page</h1>

**<?php**

echo "Hello World!";

**?>**

</body>

</html>

Try it Yourself »

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

ADVERTISEMENT

## PHP Case Sensitivity

In PHP, keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are not case-sensitive.

In the example below, all three echo statements below are equal and legal:

### Example

ECHO is the same as echo:

<!DOCTYPE html>

<html>

<body>

**<?php**

ECHO "Hello World!<br>";

echo "Hello World!<br>";

EcHo "Hello World!<br>";

**?>**

</body>

</html>

Try it Yourself »

**Note:** However; all variable names are case-sensitive!

Look at the example below; only the first statement will display the value of the $color variable! This is because $color, $COLOR, and $coLOR are treated as three different variables:

### Example

$COLOR is not same as $color:

<!DOCTYPE html>

<html>

<body>

**<?**php  
$color = "red";

echo "My car is " . $color . "<br>";

echo "My house is " . $COLOR . "<br>";

echo "My boat is " . $coLOR . "<br>";

**?>**

</body>

</html>

Try it Yourself »

## PHP Exercises

Top of Form

PHP Comments

❮ PreviousNext ❯

Comments in PHP

A comment in PHP code is a line that is not executed as a part of the program. Its only purpose is to be read by someone who is looking at the code.

Comments can be used to:

* Let others understand your code
* Remind yourself of what you did - Most programmers have experienced coming back to their own work a year or two later and having to re-figure out what they did. Comments can remind you of what you were thinking when you wrote the code
* Leave out some parts of your code

PHP supports several ways of commenting:

ExampleGet your own PHP Server

Syntax for comments in PHP code:

// This is a single-line comment

# This is also a single-line comment

/\* This is a

multi-line comment \*/

Try it Yourself »

Single Line Comments

Single line comments start with //.

Any text between // and the end of the line will be ignored (will not be executed).

You can also use # for single line comments, but in this tutorial we will use //.

The following examples uses a single-line comment as an explanation:

Example

A comment before the code:

// Outputs a welcome message:

echo "Welcome Home!";

Try it Yourself »

Example

A comment at the end of a line:

echo "Welcome Home!"; // Outputs a welcome message

Try it Yourself »

Comments to Ignore Code

We can use comments to prevent code lines from being executed:

Example

Do not display a welcome message:

// echo "Welcome Home!";

Try it Yourself »

# PHP Multiline Comments

❮ PreviousNext ❯

## Multi-line Comments

Multi-line comments start with /\* and end with \*/.

Any text between /\* and \*/ will be ignored.

The following example uses a multi-line comment as an explanation:

### ExampleGet your own PHP Server

Multi-line comment as an explanation:

/\*

The next statement will

print a welcome message

\*/

echo "Welcome Home!";

Try it Yourself »

## Multi-line Comments to Ignore Code

We can use multi-line comments to prevent blocks of code from being executed:

### Example

Multi-line comment to ignore code:

/\*

echo "Welcome to my home!";

echo "Mi casa su casa!";

\*/

echo "Hello!";

Try it Yourself »

## Comments in the Middle of the Code

The multi-line comment syntax can also be used to prevent execution of parts inside a code-line:

### Example

The  + 15 part will be ignored in the calculation:

$x = 5 /\* + 15 \*/ + 5;

echo $x;

PHP Variables

❮ PreviousNext ❯

Variables are "containers" for storing information.

Creating (Declaring) PHP Variables

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

ExampleGet your own PHP Server

$x = 5;

$y = "John"

Try it Yourself »

In the example above, the variable $x will hold the value 5, and the variable $y will hold the value "John".

**Note:** When you assign a text value to a variable, put quotes around the value.

**Note:** Unlike other programming languages, PHP has no command for declaring a variable. It is created the moment you first assign a value to it.

Think of variables as containers for storing data.

PHP Variables

A variable can have a short name (like $x and $y) or a more descriptive name ($age, $carname, $total\_volume).

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!

ADVERTISEMENT

Output Variables

The PHP echo statement is often used to output data to the screen.

The following example will show how to output text and a variable:

Example

$txt = "W3Schools.com";

echo "I love $txt!";

Try it Yourself »

The following example will produce the same output as the example above:

Example

$txt = "W3Schools.com";

echo "I love " . $txt . "!";

Try it Yourself »

The following example will output the sum of two variables:

Example

$x = 5;

$y = 4;

echo $x + $y;

Try it Yourself »

**Note:** You will learn more about the echo statement and how to output data to the screen in the PHP Echo/Print chapter.

PHP is a Loosely Typed Language

In the example above, notice that we did not have to tell PHP which data type the variable is.

PHP automatically associates a data type to the variable, depending on its value. Since the data types are not set in a strict sense, you can do things like adding a string to an integer without causing an error.

In PHP 7, type declarations were added. This gives an option to specify the data type expected when declaring a function, and by enabling the strict requirement, it will throw a "Fatal Error" on a type mismatch.

You will learn more about strict and non-strict requirements, and data type declarations in the PHP Functions chapter.

Variable Types

PHP has no command for declaring a variable, and the data type depends on the value of the variable.

Example

$x = 5; // $x is an integer

$y = "John"; // $y is a string

echo $x;

echo $y;

Try it Yourself »

PHP supports the following data types:

* String
* Integer
* Float (floating point numbers - also called double)
* Boolean
* Array
* Object
* NULL
* Resource

Get the Type

To get the data type of a variable, use the var\_dump() function.

Example

The var\_dump() function returns the data type and the value:

$x = 5;

var\_dump($x);

Try it Yourself »

Example

See what var\_dump() returns for other data types:

var\_dump(5);

var\_dump("John");

var\_dump(3.14);

var\_dump(true);

var\_dump([2, 3, 56]);

var\_dump(NULL);

Try it Yourself »

Assign String to a Variable

Assigning a string to a variable is done with the variable name followed by an equal sign and the string:

Example

$x = "John";

echo $x;

Try it Yourself »

String variables can be declared either by using double or single quotes, but you should be aware of the differences. Learn more about the differences in the PHP Strings chapter.

Assign Multiple Values

You can assign the same value to multiple variables in one line:

Example

All three variables get the value "Fruit":

$x = $y = $z = "Fruit";

Try it Yourself »

PHP Variables Scope

❮ PreviousNext ❯

PHP Variables Scope

In PHP, variables can be declared anywhere in the script.

The scope of a variable is the part of the script where the variable can be referenced/used.

PHP has three different variable scopes:

* local
* global
* static

Global and Local Scope

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

ExampleGet your own PHP Server

Variable with global scope:

$x = 5; // global scope

function myTest() {

// using x inside this function will generate an error

echo "<p>Variable x inside function is: $x</p>";

}

myTest();

echo "<p>Variable x outside function is: $x</p>";

Try it Yourself »

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

Example

Variable with local scope:

function myTest() {

$x = 5; // local scope

echo "<p>Variable x inside function is: $x</p>";

}

myTest();

// using x outside the function will generate an error

echo "<p>Variable x outside function is: $x</p>";

Try it Yourself »

You can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared.

ADVERTISEMENT

PHP The global Keyword

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

To do this, use the global keyword before the variables (inside the function):

Example

$x = 5;

$y = 10;

function myTest() {

global $x, $y;

$y = $x + $y;

}

myTest();

echo $y; // outputs 15

Try it Yourself »

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.

The example above can be rewritten like this:

Example

$x = 5;

$y = 10;

function myTest() {

$GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];

}

myTest();

echo $y; // outputs 15

Try it Yourself »

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

function myTest() {

static $x = 0;

echo $x;

$x++;

}

myTest();

myTest();

myTest();

Try it Yourself »

Then, each time the function is called, that variable will still have the information it contained from the last time the function was called.

**Note:** The variable is still local to the function.

# PHP echo and print Statements

❮ PreviousNext ❯

With PHP, there are two basic ways to get output: echo and print.

In this tutorial we use echo or print in almost every example. So, this chapter contains a little more info about those two output statements.

## PHP echo and print Statements

echo and print are more or less the same. They are both used to output data to the screen.

The differences are small: echo has no return value while print has a return value of 1 so it can be used in expressions. echo can take multiple parameters (although such usage is rare) while print can take one argument. echo is marginally faster than print.

## The PHP echo Statement

The echo statement can be used with or without parentheses: echo or echo().

**Display Text**

The following example shows how to output text with the echo command (notice that the text can contain HTML markup):

### ExampleGet your own PHP Server

echo "<h2>PHP is Fun!</h2>";

echo "Hello world!<br>";

echo "I'm about to learn PHP!<br>";

echo "This ", "string ", "was ", "made ", "with multiple parameters.";

Try it Yourself »

**Display Variables**

The following example shows how to output text and variables with the echo statement:

### Example

$txt1 = "Learn PHP";

$txt2 = "W3Schools.com";

$x = 5;

$y = 4;

echo "<h2>" . $txt1 . "</h2>";

echo "Study PHP at " . $txt2 . "<br>";

echo $x + $y;

Try it Yourself »

ADVERTISEMENT

## The PHP print Statement

The print statement can be used with or without parentheses: print or print().

**Display Text**

The following example shows how to output text with the print command (notice that the text can contain HTML markup):

### Example

print "<h2>PHP is Fun!</h2>";

print "Hello world!<br>";

print "I'm about to learn PHP!";

Try it Yourself »

**Display Variables**

The following example shows how to output text and variables with the print statement:

### Example

$txt1 = "Learn PHP";

$txt2 = "W3Schools.com";

$x = 5;

$y = 4;

print "<h2>" . $txt1 . "</h2>";

print "Study PHP at " . $txt2 . "<br>";

print $x + $y;

Try it Yourself »

PHP Data Types

❮ PreviousNext ❯

PHP Data Types

Variables can store data of different types, and different data types can do different things.

PHP supports the following data types:

* String
* Integer
* Float (floating point numbers - also called double)
* Boolean
* Array
* Object
* NULL
* Resource

Getting the Data Type

You can get the data type of any object by using the var\_dump() function.

ExampleGet your own PHP Server

The var\_dump() function returns the data type and the value:

$x = 5;

var\_dump($x);

Try it Yourself »

PHP String

A string is a sequence of characters, like "Hello world!".

A string can be any text inside quotes. You can use single or double quotes:

Example

$x = "Hello world!";

$y = 'Hello world!';

var\_dump($x);

echo "<br>";

var\_dump($y);

Try it Yourself »

PHP Integer

An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.

Rules for integers:

* An integer must have at least one digit
* An integer must not have a decimal point
* An integer can be either positive or negative
* Integers can be specified in: decimal (base 10), hexadecimal (base 16), octal (base 8), or binary (base 2) notation

In the following example $x is an integer. The PHP var\_dump() function returns the data type and value:

Example

$x = 5985;

var\_dump($x);

Try it Yourself »

ADVERTISEMENT

PHP Float

A float (floating point number) is a number with a decimal point or a number in exponential form.

In the following example $x is a float. The PHP var\_dump() function returns the data type and value:

Example

$x = 10.365;

var\_dump($x);

Try it Yourself »

PHP Boolean

A Boolean represents two possible states: TRUE or FALSE.

Example

$x = true;

var\_dump($x);

Try it Yourself »

Booleans are often used in conditional testing.

You will learn more about conditional testing in the PHP If...Else chapter.

PHP Array

An array stores multiple values in one single variable.

In the following example $cars is an array. The PHP var\_dump() function returns the data type and value:

Example

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

var\_dump($cars);

Try it Yourself »

You will learn a lot more about arrays in later chapters of this tutorial.

PHP Object

Classes and objects are the two main aspects of object-oriented programming.

A class is a template for objects, and an object is an instance of a class.

When the individual objects are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.

Let's assume we have a class named Car that can have properties like model, color, etc. We can define variables like $model, $color, and so on, to hold the values of these properties.

When the individual objects (Volvo, BMW, Toyota, etc.) are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.

If you create a \_\_construct() function, PHP will automatically call this function when you create an object from a class.

Example

class Car {

public $color;

public $model;

public function \_\_construct($color, $model) {

$this->color = $color;

$this->model = $model;

}

public function message() {

return "My car is a " . $this->color . " " . $this->model . "!";

}

}

$myCar = new Car("red", "Volvo");

var\_dump($myCar);

Try it Yourself »

Do not worry if you do not understand the PHP Object syntax, you will learn more about that in the PHP Classes/Objects chapter.

PHP NULL Value

Null is a special data type which can have only one value: NULL.

A variable of data type NULL is a variable that has no value assigned to it.

**Tip:** If a variable is created without a value, it is automatically assigned a value of NULL.

Variables can also be emptied by setting the value to NULL:

Example

$x = "Hello world!";

$x = null;

var\_dump($x);

Try it Yourself »

Change Data Type

If you assign an integer value to a variable, the type will automatically be an integer.

If you assign a string to the same variable, the type will change to a string:

Example

$x = 5;

var\_dump($x);

$x = "Hello";

var\_dump($x);

Try it Yourself »

If you want to change the data type of an existing variable, but not by changing the value, you can use casting.

Casting allows you to change data type on variables:

Example

$x = 5;

$x = (string) $x;

var\_dump($x);

Try it Yourself »

You will learn more about casting in the PHP Casting Chapter.

PHP Resource

The special resource type is not an actual data type. It is the storing of a reference to functions and resources external to PHP.

A common example of using the resource data type is a database call.

We will not talk about the resource type here, since it is an advanced topic.

# HP Strings

❮ PreviousNext ❯

A string is a sequence of characters, like "Hello world!".

## Strings

Strings in PHP are surrounded by either double quotation marks, or single quotation marks.

### ExampleGet your own PHP Server

echo "Hello";

echo 'Hello';

Try it Yourself »

**Note** There is a big difference between double quotes and single quotes in PHP.

Double quotes process special characters, single quotes does not.

## Double or Single Quotes?

You can use double or single quotes, but you should be aware of the differences between the two.

Double quoted strings perform action on special characters.

E.g. when there is a variable in the string, it returns the value of the variable:

### Example

Double quoted string literals perform operations for special characters:

$x = "John";

echo "Hello $x";

Try it Yourself »

Single quoted strings does not perform such actions, it returns the string like it was written, with the variable name:

### Example

Single quoted string literals returns the string as it is:

$x = "John";

echo 'Hello $x';

Try it Yourself »

ADVERTISEMENT

## String Length

The PHP strlen() function returns the length of a string.

### Example

Return the length of the string "Hello world!":

echo strlen("Hello world!");

Try it Yourself »

## Word Count

The PHP str\_word\_count() function counts the number of words in a string.

### Example

Count the number of word in the string "Hello world!":

echo str\_word\_count("Hello world!");

Try it Yourself »

## Search For 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.

### Example

Search for the text "world" in the string "Hello world!":

echo strpos("Hello world!", "world");

Try it Yourself »

**Tip:** The first character position in a string is 0 (not 1).

## Complete PHP String Reference

For a complete reference of all string functions, go to our complete PHP String Reference.

The PHP string reference contains description and example of use, for each function!

# PHP String Functions

❮ PreviousNext ❯

## PHP String Functions

The PHP string functions are part of the PHP core. No installation is required to use these functions.

|  |  |
| --- | --- |
| **Function** | **Description** |
| addcslashes() | Returns a string with backslashes in front of the specified characters |
| addslashes() | Returns a string with backslashes in front of predefined characters |
| bin2hex() | Converts a string of ASCII characters to hexadecimal values |
| chop() | Removes whitespace or other characters from the right end of a string |
| chr() | Returns a character from a specified ASCII value |
| chunk\_split() | Splits a string into a series of smaller parts |
| convert\_cyr\_string() | Converts a string from one Cyrillic character-set to another |
| convert\_uudecode() | Decodes a uuencoded string |
| convert\_uuencode() | Encodes a string using the uuencode algorithm |
| count\_chars() | Returns information about characters used in a string |
| crc32() | Calculates a 32-bit CRC for a string |
| crypt() | One-way string hashing |
| echo() | Outputs one or more strings |
| explode() | Breaks a string into an array |
| fprintf() | Writes a formatted string to a specified output stream |
| get\_html\_translation\_table() | Returns the translation table used by htmlspecialchars() and htmlentities() |
| hebrev() | Converts Hebrew text to visual text |
| hebrevc() | Converts Hebrew text to visual text and new lines (\n) into <br> |
| hex2bin() | Converts a string of hexadecimal values to ASCII characters |
| html\_entity\_decode() | Converts HTML entities to characters |
| htmlentities() | Converts characters to HTML entities |
| htmlspecialchars\_decode() | Converts some predefined HTML entities to characters |
| htmlspecialchars() | Converts some predefined characters to HTML entities |
| implode() | Returns a string from the elements of an array |
| join() | Alias of implode() |
| lcfirst() | Converts the first character of a string to lowercase |
| levenshtein() | Returns the Levenshtein distance between two strings |
| localeconv() | Returns locale numeric and monetary formatting information |
| ltrim() | Removes whitespace or other characters from the left side of a string |
| md5() | Calculates the MD5 hash of a string |
| md5\_file() | Calculates the MD5 hash of a file |
| metaphone() | Calculates the metaphone key of a string |
| money\_format() | Returns a string formatted as a currency string |
| nl\_langinfo() | Returns specific local information |
| nl2br() | Inserts HTML line breaks in front of each newline in a string |
| number\_format() | Formats a number with grouped thousands |
| ord() | Returns the ASCII value of the first character of a string |
| parse\_str() | Parses a query string into variables |
| print() | Outputs one or more strings |
| printf() | Outputs a formatted string |
| quoted\_printable\_decode() | Converts a quoted-printable string to an 8-bit string |
| quoted\_printable\_encode() | Converts an 8-bit string to a quoted printable string |
| quotemeta() | Quotes meta characters |
| rtrim() | Removes whitespace or other characters from the right side of a string |
| setlocale() | Sets locale information |
| sha1() | Calculates the SHA-1 hash of a string |
| sha1\_file() | Calculates the SHA-1 hash of a file |
| similar\_text() | Calculates the similarity between two strings |
| soundex() | Calculates the soundex key of a string |
| sprintf() | Writes a formatted string to a variable |
| sscanf() | Parses input from a string according to a format |
| str\_getcsv() | Parses a CSV string into an array |
| str\_ireplace() | Replaces some characters in a string (case-insensitive) |
| str\_pad() | Pads a string to a new length |
| str\_repeat() | Repeats a string a specified number of times |
| str\_replace() | Replaces some characters in a string (case-sensitive) |
| str\_rot13() | Performs the ROT13 encoding on a string |
| str\_shuffle() | Randomly shuffles all characters in a string |
| str\_split() | Splits a string into an array |
| str\_word\_count() | Count the number of words in a string |
| strcasecmp() | Compares two strings (case-insensitive) |
| strchr() | Finds the first occurrence of a string inside another string (alias of strstr()) |
| strcmp() | Compares two strings (case-sensitive) |
| strcoll() | Compares two strings (locale based string comparison) |
| strcspn() | Returns the number of characters found in a string before any part of some specified characters are found |
| strip\_tags() | Strips HTML and PHP tags from a string |
| stripcslashes() | Unquotes a string quoted with addcslashes() |
| stripslashes() | Unquotes a string quoted with addslashes() |
| stripos() | Returns the position of the first occurrence of a string inside another string (case-insensitive) |
| stristr() | Finds the first occurrence of a string inside another string (case-insensitive) |
| strlen() | Returns the length of a string |
| strnatcasecmp() | Compares two strings using a "natural order" algorithm (case-insensitive) |
| strnatcmp() | Compares two strings using a "natural order" algorithm (case-sensitive) |
| strncasecmp() | String comparison of the first n characters (case-insensitive) |
| strncmp() | String comparison of the first n characters (case-sensitive) |
| strpbrk() | Searches a string for any of a set of characters |
| strpos() | Returns the position of the first occurrence of a string inside another string (case-sensitive) |
| strrchr() | Finds the last occurrence of a string inside another string |
| strrev() | Reverses a string |
| strripos() | Finds the position of the last occurrence of a string inside another string (case-insensitive) |
| strrpos() | Finds the position of the last occurrence of a string inside another string (case-sensitive) |
| strspn() | Returns the number of characters found in a string that contains only characters from a specified charlist |
| strstr() | Finds the first occurrence of a string inside another string (case-sensitive) |
| strtok() | Splits a string into smaller strings |
| strtolower() | Converts a string to lowercase letters |
| strtoupper() | Converts a string to uppercase letters |
| strtr() | Translates certain characters in a string |
| substr() | Returns a part of a string |
| substr\_compare() | Compares two strings from a specified start position (binary safe and optionally case-sensitive) |
| substr\_count() | Counts the number of times a substring occurs in a string |
| substr\_replace() | Replaces a part of a string with another string |
| trim() | Removes whitespace or other characters from both sides of a string |
| ucfirst() | Converts the first character of a string to uppercase |
| ucwords() | Converts the first character of each word in a string to uppercase |
| vfprintf() | Writes a formatted string to a specified output stream |
| vprintf() | Outputs a formatted string |
| vsprintf() | Writes a formatted string to a variable |
| wordwrap() | Wraps a string to a given number of characters |

# PHP - Modify Strings

❮ PreviousNext ❯

PHP has a set of built-in functions that you can use to modify strings.

## Upper Case

### ExampleGet your own PHP Server

The strtoupper() function returns the string in upper case:

$x = "Hello World!";

echo strtoupper($x);

Try it Yourself »

## Lower Case

### Example

The strtolower() function returns the string in lower case:

$x = "Hello World!";

echo strtolower($x);

Try it Yourself »

## Replace String

The PHP str\_replace() function replaces some characters with some other characters in a string.

### Example

Replace the text "World" with "Dolly":

$x = "Hello World!";

echo str\_replace("World", "Dolly", $x);

Try it Yourself »

ADVERTISEMENT

## Reverse a String

The PHP strrev() function reverses a string.

### Example

Reverse the string "Hello World!":

$x = "Hello World!";

echo strrev($x);

Try it Yourself »

## Remove Whitespace

Whitespace is the space before and/or after the actual text, and very often you want to remove this space.

### Example

The trim() removes any whitespace from the beginning or the end:

$x = " Hello World! ";

echo trim($x);

Try it Yourself »

Learn more in our trim() Function Reference.

## Convert String into Array

The PHP explode() function splits a string into an array.

The first parameter of the explode() function represents the "separator". The "separator" specifies where to split the string.

**Note:** The separator is required.

### Example

Split the string into an array. Use the space character as separator:

$x = "Hello World!";

$y = explode(" ", $x);

//Use the print\_r() function to display the result:

print\_r($y);

/\*

Result:

Array ( [0] => Hello [1] => World! )

\*/

# PHP - Concatenate Strings

❮ PreviousNext ❯

## String Concatenation

To concatenate, or combine, two strings you can use the . operator:

### ExampleGet your own PHP Server

$x = "Hello";

$y = "World";

$z = $x . $y;

echo $z;

Try it Yourself »

The result of the example above is HelloWorld, without a space between the two words.

You can add a space character like this:

### Example

$x = "Hello";

$y = "World";

$z = $x . " " . $y;

echo $z;

Try it Yourself »

An easier and better way is by using the power of double quotes.

By surrounding the two variables in double quotes with a white space between them, the white space will also be present in the result:

### Example

$x = "Hello";

$y = "World";

$z = "$x $y";

echo $z;

PHP - Slicing Strings

❮ PreviousNext ❯

Slicing

You can return a range of characters by using the substr() function.

Specify the start index and the number of characters you want to return.

ExampleGet your own PHP Server

Start the slice at index 6 and end the slice 5 positions later:

$x = "Hello World!";

echo substr($x, 6, 5);

Try it Yourself »

**Note** The first character has index 0.

Slice to the End

By leaving out the *length* parameter, the range will go to the end:

Example

Start the slice at index 6 and go all the way to the end:

$x = "Hello World!";

echo substr($x, 6);

Try it Yourself »

Slice *From* the End

Use negative indexes to start the slice from the end of the string:

Example

Get the 3 characters, starting from the "o" in world (index -5):

$x = "Hello World!";

echo substr($x, -5, 3);

Try it Yourself »

**Note** The last character has index -1.

Negative Length

Use negative *length*to specify how many characters to omit, starting from the end of the string:

Example

From the string "Hi, how are you?", get the characters starting from index 5, and continue until you reach the 3. character from the end (index -3).

Should end up with "ow are y":

$x = "Hi, how are you?";

echo substr($x, 5, -3);

PHP - Escape Characters

❮ PreviousNext ❯

Escape Character

To insert characters that are illegal in a string, use an escape character.

An escape character is a backslash \ followed by the character you want to insert.

An example of an illegal character is a double quote inside a string that is surrounded by double quotes:

ExampleGet your own PHP Server

$x = "We are the so-called "Vikings" from the north.";

Try it Yourself »

To fix this problem, use the escape character \":

Example

$x = "We are the so-called \"Vikings\" from the north.";

Try it Yourself »

Escape Characters

Other escape characters used in PHP:

|  |  |
| --- | --- |
| **Code** | **Result** |
| \' | Single Quote |
| \" | Double Quote |
| \$ | PHP variables |
| \n | New Line |
| \r | Carriage Return |
| \t | Tab |
| \f | Form Feed |
| \ooo | Octal value |
| \xhh | Hex value |

# PHP Numbers

❮ PreviousNext ❯

In this chapter we will look in depth into Integers, Floats, and Number Strings.

## PHP Numbers

There are three main numeric types in PHP:

* Integer
* Float
* Number Strings

In addition, PHP has two more data types used for numbers:

* Infinity
* NaN

Variables of numeric types are created when you assign a value to them:

### ExampleGet your own PHP Server

$a = 5;

$b = 5.34;

$c = "25";

To verify the type of any object in PHP, use the var\_dump() function:

### Example

var\_dump($a);

var\_dump($b);

var\_dump($c);

Try it Yourself »

## PHP Integers

2, 256, -256, 10358, -179567 are all integers.

An integer is a number without any decimal part.

An integer data type is a non-decimal number between -2147483648 and 2147483647 in 32 bit systems, and between -9223372036854775808 and 9223372036854775807 in 64 bit systems. A value greater (or lower) than this, will be stored as float, because it exceeds the limit of an integer.

**Note:** Another important thing to know is that even if 4 \* 2.5 is 10, the result is stored as float, because one of the operands is a float (2.5).

Here are some rules for integers:

* An integer must have at least one digit
* An integer must NOT have a decimal point
* An integer can be either positive or negative
* Integers can be specified in three formats: decimal (base 10), hexadecimal (base 16 - prefixed with 0x), octal (base 8 - prefixed with 0) or binary (base 2 - prefixed with 0b)

PHP has the following predefined constants for integers:

* PHP\_INT\_MAX - The largest integer supported
* PHP\_INT\_MIN - The smallest integer supported
* PHP\_INT\_SIZE -  The size of an integer in bytes

PHP has the following functions to check if the type of a variable is integer:

* is\_int()
* is\_integer() - alias of is\_int()
* is\_long() - alias of is\_int()

### Example

Check if the type of a variable is integer:

$x = 5985;

var\_dump(is\_int($x));

$x = 59.85;

var\_dump(is\_int($x));

Try it Yourself »

ADVERTISEMENT

## PHP Floats

A float is a number with a decimal point or a number in exponential form.

2.0, 256.4, 10.358, 7.64E+5, 5.56E-5 are all floats.

The float data type can commonly store a value up to 1.7976931348623E+308 (platform dependent), and have a maximum precision of 14 digits.

PHP has the following predefined constants for floats (from PHP 7.2):

* PHP\_FLOAT\_MAX - The largest representable floating point number
* PHP\_FLOAT\_MIN - The smallest representable positive floating point number
* PHP\_FLOAT\_DIG - The number of decimal digits that can be rounded into a float and back without precision loss
* PHP\_FLOAT\_EPSILON - The smallest representable positive number x, so that x + 1.0 != 1.0

PHP has the following functions to check if the type of a variable is float:

* is\_float()
* is\_double() - alias of is\_float()

### Example

Check if the type of a variable is float:

$x = 10.365;

var\_dump(is\_float($x));

Try it Yourself »

## PHP Infinity

A numeric value that is larger than PHP\_FLOAT\_MAX is considered infinite.

PHP has the following functions to check if a numeric value is finite or infinite:

* is\_finite()
* is\_infinite()

However, the PHP var\_dump() function returns the data type and value:

### Example

Check if a numeric value is finite or infinite:

$x = 1.9e411;

var\_dump($x);

Try it Yourself »

## PHP NaN

NaN stands for Not a Number.

NaN is used for impossible mathematical operations.

PHP has the following functions to check if a value is not a number:

* is\_nan()

However, the PHP var\_dump() function returns the data type and value:

### Example

Invalid calculation will return a NaN value:

$x = acos(8);

var\_dump($x);

Try it Yourself »

## PHP Numerical Strings

The PHP is\_numeric() function can be used to find whether a variable is numeric. The function returns true if the variable is a number or a numeric string, false otherwise.

### Example

Check if the variable is numeric:

$x = 5985;

var\_dump(is\_numeric($x));

$x = "5985";

var\_dump(is\_numeric($x));

$x = "59.85" + 100;

var\_dump(is\_numeric($x));

$x = "Hello";

var\_dump(is\_numeric($x));

Try it Yourself »

**Note:** From PHP 7.0: The is\_numeric() function will return FALSE for numeric strings in hexadecimal form (e.g. 0xf4c3b00c), as they are no longer considered as numeric strings.

## PHP Casting Strings and Floats to Integers

Sometimes you need to cast a numerical value into another data type.

The (int), (integer), and intval() functions are often used to convert a value to an integer.

### Example

Cast float and string to integer:

// Cast float to int

$x = 23465.768;

$int\_cast = (int)$x;

echo $int\_cast;

echo "<br>";

// Cast string to int

$x = "23465.768";

$int\_cast = (int)$x;

echo $int\_cast;

Try it Yourself »

# PHP Casting

❮ PreviousNext ❯

Sometimes you need to change a variable from one data type into another, and sometimes you want a variable to have a specific data type. This can be done with casting.

## Change Data Type

Casting in PHP is done with these statements:

* (string) - Converts to data type String
* (int) - Converts to data type Integer
* (float) - Converts to data type Float
* (bool) - Converts to data type Boolean
* (array) - Converts to data type Array
* (object) - Converts to data type Object
* (unset) - Converts to data type NULL

## Cast to String

To cast to string, use the (string) statement:

### ExampleGet your own PHP Server

$a = 5; // Integer

$b = 5.34; // Float

$c = "hello"; // String

$d = true; // Boolean

$e = NULL; // NULL

$a = (string) $a;

$b = (string) $b;

$c = (string) $c;

$d = (string) $d;

$e = (string) $e;

//To verify the type of any object in PHP, use the var\_dump() function:

var\_dump($a);

var\_dump($b);

var\_dump($c);

var\_dump($d);

var\_dump($e);

Try it Yourself »

## Cast to Integer

To cast to integer, use the (int) statement:

### Example

$a = 5; // Integer

$b = 5.34; // Float

$c = "25 kilometers"; // String

$d = "kilometers 25"; // String

$e = "hello"; // String

$f = true; // Boolean

$g = NULL; // NULL

$a = (int) $a;

$b = (int) $b;

$c = (int) $c;

$d = (int) $d;

$e = (int) $e;

$f = (int) $f;

$g = (int) $g;

Try it Yourself »

## Cast to Float

To cast to float, use the (float) statement:

### Example

$a = 5; // Integer

$b = 5.34; // Float

$c = "25 kilometers"; // String

$d = "kilometers 25"; // String

$e = "hello"; // String

$f = true; // Boolean

$g = NULL; // NULL

$a = (float) $a;

$b = (float) $b;

$c = (float) $c;

$d = (float) $d;

$e = (float) $e;

$f = (float) $f;

$g = (float) $g;

Try it Yourself »

## Cast to Boolean

To cast to boolean, use the (bool) statement:

### Example

$a = 5; // Integer

$b = 5.34; // Float

$c = 0; // Integer

$d = -1; // Integer

$e = 0.1; // Float

$f = "hello"; // String

$g = ""; // String

$h = true; // Boolean

$i = NULL; // NULL

$a = (bool) $a;

$b = (bool) $b;

$c = (bool) $c;

$d = (bool) $d;

$e = (bool) $e;

$f = (bool) $f;

$g = (bool) $g;

$h = (bool) $h;

$i = (bool) $i;

Try it Yourself »

If a value is 0, NULL, false, or empty, the (bool) converts it into false, otherwise true.

Even -1 converts to true.

## Cast to Array

To cast to array, use the (array) statement:

### Example

$a = 5; // Integer

$b = 5.34; // Float

$c = "hello"; // String

$d = true; // Boolean

$e = NULL; // NULL

$a = (array) $a;

$b = (array) $b;

$c = (array) $c;

$d = (array) $d;

$e = (array) $e;

Try it Yourself »

When converting into arrays, most data types converts into an indexed array with one element.

NULL values converts to an empty array object.

Objects converts into associative arrays where the property names becomes the keys and the property values becomes the values:

### Example

Converting Objects into Arrays:

class Car {

public $color;

public $model;

public function \_\_construct($color, $model) {

$this->color = $color;

$this->model = $model;

}

public function message() {

return "My car is a " . $this->color . " " . $this->model . "!";

}

}

$myCar = new Car("red", "Volvo");

$myCar = (array) $myCar;

var\_dump($myCar);

Try it Yourself »

## Cast to Object

To cast to object, use the (object) statement:

### Example

$a = 5; // Integer

$b = 5.34; // Float

$c = "hello"; // String

$d = true; // Boolean

$e = NULL; // NULL

$a = (object) $a;

$b = (object) $b;

$c = (object) $c;

$d = (object) $d;

$e = (object) $e;

Try it Yourself »

When converting into objects, most data types converts into a object with one property, named "scalar", with the corresponding value.

NULL values converts to an empty object.

Indexed arrays converts into objects with the index number as property name and the value as property value.

Associative arrays converts into objects with the keys as property names and values as property values.

### Example

Converting Arrays into Objects:

$a = array("Volvo", "BMW", "Toyota"); // indexed array

$b = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43"); // associative array

$a = (object) $a;

$b = (object) $b;

Try it Yourself »

## Cast to NULL

To cast to NULL, use the (unset) statement:

### Example

$a = 5; // Integer

$b = 5.34; // Float

$c = "hello"; // String

$d = true; // Boolean

$e = NULL; // NULL

$a = (unset) $a;

$b = (unset) $b;

$c = (unset) $c;

$d = (unset) $d;

$e = (unset) $e;

PHP Math

❮ PreviousNext ❯

PHP has a set of math functions that allows you to perform mathematical tasks on numbers.

PHP pi() Function

The pi() function returns the value of PI:

ExampleGet your own PHP Server

echo(pi());

Try it Yourself »

PHP min() and max() Functions

The min() and max() functions can be used to find the lowest or highest value in a list of arguments:

Example

echo(min(0, 150, 30, 20, -8, -200));

echo(max(0, 150, 30, 20, -8, -200));

Try it Yourself »

PHP abs() Function

The abs() function returns the absolute (positive) value of a number:

Example

echo(abs(-6.7));

Try it Yourself »

ADVERTISEMENT

PHP sqrt() Function

The sqrt() function returns the square root of a number:

Example

echo(sqrt(64));

Try it Yourself »

PHP round() Function

The round() function rounds a floating-point number to its nearest integer:

Example

echo(round(0.60));

echo(round(0.49));

Try it Yourself »

Random Numbers

The rand() function generates a random number:

Example

echo(rand());

Try it Yourself »

To get more control over the random number, you can add the optional *min* and *max* parameters to specify the lowest integer and the highest integer to be returned.

For example, if you want a random integer between 10 and 100 (inclusive), use rand(10, 100):

Example

echo(rand(10, 100));

Try it Yourself »

Complete PHP Math Reference

For a complete reference of all math functions, go to our complete PHP Math Reference.

The PHP math reference contains description and example of use, for each function.

PHP Math Functions

❮ PreviousNext ❯

PHP Math Introduction

The math functions can handle values within the range of integer and float types.

Installation

The PHP math functions are part of the PHP core. No installation is required to use these functions.

PHP Math Functions

|  |  |
| --- | --- |
| **Function** | **Description** |
| abs() | Returns the absolute (positive) value of a number |
| acos() | Returns the arc cosine of a number |
| acosh() | Returns the inverse hyperbolic cosine of a number |
| asin() | Returns the arc sine of a number |
| asinh() | Returns the inverse hyperbolic sine of a number |
| atan() | Returns the arc tangent of a number in radians |
| atan2() | Returns the arc tangent of two variables x and y |
| atanh() | Returns the inverse hyperbolic tangent of a number |
| base\_convert() | Converts a number from one number base to another |
| bindec() | Converts a binary number to a decimal number |
| ceil() | Rounds a number up to the nearest integer |
| cos() | Returns the cosine of a number |
| cosh() | Returns the hyperbolic cosine of a number |
| decbin() | Converts a decimal number to a binary number |
| dechex() | Converts a decimal number to a hexadecimal number |
| decoct() | Converts a decimal number to an octal number |
| deg2rad() | Converts a degree value to a radian value |
| exp() | Calculates the exponent of e |
| expm1() | Returns exp(x) - 1 |
| floor() | Rounds a number down to the nearest integer |
| fmod() | Returns the remainder of x/y |
| getrandmax() | Returns the largest possible value returned by rand() |
| hexdec() | Converts a hexadecimal number to a decimal number |
| hypot() | Calculates the hypotenuse of a right-angle triangle |
| intdiv() | Performs integer division |
| is\_finite() | Checks whether a value is finite or not |
| is\_infinite() | Checks whether a value is infinite or not |
| is\_nan() | Checks whether a value is 'not-a-number' |
| lcg\_value() | Returns a pseudo random number in a range between 0 and 1 |
| log() | Returns the natural logarithm of a number |
| log10() | Returns the base-10 logarithm of a number |
| log1p() | Returns log(1+number) |
| max() | Returns the highest value in an array, or the highest value of several specified values |
| min() | Returns the lowest value in an array, or the lowest value of several specified values |
| mt\_getrandmax() | Returns the largest possible value returned by mt\_rand() |
| mt\_rand() | Generates a random integer using Mersenne Twister algorithm |
| mt\_srand() | Seeds the Mersenne Twister random number generator |
| octdec() | Converts an octal number to a decimal number |
| pi() | Returns the value of PI |
| pow() | Returns x raised to the power of y |
| rad2deg() | Converts a radian value to a degree value |
| rand() | Generates a random integer |
| round() | Rounds a floating-point number |
| sin() | Returns the sine of a number |
| sinh() | Returns the hyperbolic sine of a number |
| sqrt() | Returns the square root of a number |
| srand() | Seeds the random number generator |
| tan() | Returns the tangent of a number |
| tanh() | Returns the hyperbolic tangent of a number |

ADVERTISEMENT

PHP Predefined Math Constants

|  |  |  |
| --- | --- | --- |
| **Constant** | **Value** | **Description** |
| INF | INF | The infinite |
| M\_E | 2.7182818284590452354 | Returns e |
| M\_EULER | 0.57721566490153286061 | Returns Euler constant |
| M\_LNPI | 1.14472988584940017414 | Returns the natural logarithm of PI: log\_e(pi) |
| M\_LN2 | 0.69314718055994530942 | Returns the natural logarithm of 2: log\_e 2 |
| M\_LN10 | 2.30258509299404568402 | Returns the natural logarithm of 10: log\_e 10 |
| M\_LOG2E | 1.4426950408889634074 | Returns the base-2 logarithm of E: log\_2 e |
| M\_LOG10E | 0.43429448190325182765 | Returns the base-10 logarithm of E: log\_10 e |
| M\_PI | 3.14159265358979323846 | Returns Pi |
| M\_PI\_2 | 1.57079632679489661923 | Returns Pi/2 |
| M\_PI\_4 | 0.78539816339744830962 | Returns Pi/4 |
| M\_1\_PI | 0.31830988618379067154 | Returns 1/Pi |
| M\_2\_PI | 0.63661977236758134308 | Returns 2/Pi |
| M\_SQRTPI | 1.77245385090551602729 | Returns the square root of PI: sqrt(pi) |
| M\_2\_SQRTPI | 1.12837916709551257390 | Returns 2/square root of PI: 2/sqrt(pi) |
| M\_SQRT1\_2 | 0.70710678118654752440 | Returns the square root of 1/2: 1/sqrt(2) |
| M\_SQRT2 | 1.41421356237309504880 | Returns the square root of 2: sqrt(2) |
| M\_SQRT3 | 1.73205080756887729352 | Returns the square root of 3: sqrt(3) |
| NAN | NAN | Not A Number |
| PHP\_ROUND\_HALF\_UP | 1 | Round halves up |
| PHP\_ROUND\_HALF\_DOWN | 2 | Round halves down |
| PHP\_ROUND\_HALF\_EVEN | 3 | Round halves to even numbers |
| PHP\_ROUND\_HALF\_ODD | 4 | Round halves to odd numbers |

# PHP Constants

❮ PreviousNext ❯

Constants are like variables, except that once they are defined they cannot be changed or undefined.

## PHP Constants

A constant is an identifier (name) for a simple value. The value cannot be changed during the script.

A valid constant name starts with a letter or underscore (no $ sign before the constant name).

**Note:** Unlike variables, constants are automatically global across the entire script.

## Create a PHP Constant

To create a constant, use the define() function.

### Syntax

define(*name*, *value*, *case-insensitive*);

Parameters:

* *name*: Specifies the name of the constant
* *value*: Specifies the value of the constant
* *case-insensitive*: Specifies whether the constant name should be case-insensitive. Default is false. **Note:** Defining case-insensitive constants was deprecated in PHP 7.3. PHP 8.0 accepts only false, the value true will produce a warning.

### ExampleGet your own PHP Server

Create a constant with a **case-sensitive** name:

define("GREETING", "Welcome to W3Schools.com!");

echo GREETING;

Try it Yourself »

### Example

Create a constant with a **case-insensitive** name:

define("GREETING", "Welcome to W3Schools.com!", true);

echo greeting;

Try it Yourself »

ADVERTISEMENT

## PHP const Keyword

You can also create a constant by using the const keyword.

### Example

Create a constant with the const keyword:

const MYCAR = "Volvo";

echo MYCAR;

Try it Yourself »

**const vs. define()**

* const are always case-sensitive
* define() has has a case-insensitive option.
* const cannot be created inside another block scope, like inside a function or inside an if statement.
* define can be created inside another block scope.

## PHP Constant Arrays

From PHP7, you can create an Array constant using the define() function.

### Example

Create an Array constant:

define("cars", [

"Alfa Romeo",

"BMW",

"Toyota"

]);

echo cars[0];

Try it Yourself »

## Constants are Global

Constants are automatically global and can be used across the entire script.

### Example

This example uses a constant inside a function, even if it is defined outside the function:

define("GREETING", "Welcome to W3Schools.com!");

function myTest() {

echo GREETING;

}

myTest();

PHP Magic Constants

❮ PreviousNext ❯

PHP Predefined Constants

PHP has nine predefined constants that change value depending on where they are used, and therefor they are called "magic constants".

These magic constants are written with a double underscore at the start and the end, except for the ClassName::class constant.

Magic Constants

Here are the magic constants, with descriptions and examples:

|  |  |
| --- | --- |
| **Constant** | **Description** |
| \_\_CLASS\_\_ | If used inside a class, the class name is returned. |
| \_\_DIR\_\_ | The directory of the file. |
| \_\_FILE\_\_ | The file name including the full path. |
| \_\_FUNCTION\_\_ | If inside a function, the function name is returned. |
| \_\_LINE\_\_ | The current line number. |
| \_\_METHOD\_\_ | If used inside a function that belongs to a class, both class and function name is returned. |
| \_\_NAMESPACE\_\_ | If used inside a namespace, the name of the namespace is returned. |
| \_\_TRAIT\_\_ | If used inside a trait, the trait name is returned. |
| ClassName::class | Returns the name of the specified class and the name of the namespace, if any. |

**Note:**

The magic constants are case-insensitive, meaning \_\_LINE\_\_ returns the same as \_\_line\_\_.

# PHP Operators

❮ PreviousNext ❯

## 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
* Conditional assignment operators

## PHP 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 |

## PHP Assignment Operators

The PHP assignment operators are used with numeric values to write a value to a variable.

The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression on the right.

|  |  |  |
| --- | --- | --- |
| **Assignment** | **Same as...** | **Description** |
| x = y | x = y | The left operand gets set to the value of the expression on the right |
| x += y | x = x + y | Addition |
| x -= y | x = x - y | Subtraction |
| x \*= y | x = x \* y | Multiplication |
| x /= y | x = x / y | Division |
| x %= y | x = x % y | Modulus |

ADVERTISEMENT

## 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 |
| <=> | Spaceship | $x <=> $y | Returns an integer less than, equal to, or greater than zero, depending on if $x is less than, equal to, or greater than $y. Introduced in PHP 7. |

## PHP Increment / Decrement Operators

The PHP increment operators are used to increment a variable's value.

The PHP decrement operators are used to decrement a variable's value.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Same as...** | **Description** |
| ++$x | Pre-increment | Increments $x by one, then returns $x |
| $x++ | Post-increment | Returns $x, then increments $x by one |
| --$x | Pre-decrement | Decrements $x by one, then returns $x |
| $x-- | Post-decrement | Returns $x, then decrements $x by one |

## 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 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 Array Operators

The PHP array operators are used to compare arrays.

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Name** | **Example** | **Result** |
| + | Union | $x + $y | Union of $x and $y |
| == | Equality | $x == $y | Returns true if $x and $y have the same key/value pairs |
| === | Identity | $x === $y | Returns true if $x and $y have the same key/value pairs in the same order and of the same types |
| != | Inequality | $x != $y | Returns true if $x is not equal to $y |
| <> | Inequality | $x <> $y | Returns true if $x is not equal to $y |
| !== | Non-identity | $x !== $y | Returns true if $x is not identical to $y |

## PHP Conditional Assignment Operators

The PHP conditional assignment operators are used to set a value depending on conditions:

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Name** | **Example** | **Result** |
| ?: | Ternary | $x = expr1 ? expr2 : expr3 | Returns the value of $x. The value of $x is expr2 if expr1 = TRUE. The value of $x is expr3 if expr1 = FALSE |
| ?? | Null coalescing | $x = expr1 ?? expr2 | Returns the value of $x. The value of $x is expr1 if expr1 exists, and is not NULL. If expr1 does not exist, or is NULL, the value of $x is expr2. Introduced in PHP 7 |

# PHP if Statements

❮ PreviousNext ❯

Conditional statements are used to perform different actions based on different conditions.

## 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

## PHP - The if Statement

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

### Syntax

if (*condition*) {

*// code to be executed if condition is true*;

}

### ExampleGet your own PHP Server

Output "Have a good day!" if 5 is larger than 3:

if (5 > 3) {

echo "Have a good day!";

}

Try it Yourself »

We can also use variables in the if statement:

### Example

Output "Have a good day!" if $t is less than 20:

$t = 14;

if ($t < 20) {

echo "Have a good day!";

}

PHP if Operators

❮ PreviousNext ❯

Comparison Operators

If statements usually contain conditions that compare two values.

ExampleGet your own PHP Server

Check if $t is equal to 14:

$t = 14;

if ($t == 14) {

echo "Have a good day!";

}

Try it Yourself »

To compare two values, we need to use a comparison operator.

Here are the PHP comparison operators to use in if statements:

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Name** | **Result** | **Try it** |
| == | Equal | Returns true if the values are equal | Try it » |
| === | Identical | Returns true if the values and data types are identical | Try it » |
| != | Not equal | Returns true if the values are not equal | Try it » |
| <> | Not equal | Returns true if the values are not equal | Try it » |
| !== | Not identical | Returns true if the values or data types are not identical | Try it » |
| > | Greater than | Returns true if the first value is greater than the second value | Try it » |
| < | Less than | Returns true if the first value is less than the second value | Try it » |
| >= | Greater than or equal to | Returns true if the first value is greater than, or equal to, the second value | Try it » |
| <= | Less than or equal to | Returns true if the first value is less than, or equal to, the second value | Try it » |

Logical Operators

To check more than one condition, we can use logical operators, like the && operator:

Example

Check if $a is greater than $b, AND if $a is less than $c:

$a = 200;

$b = 33;

$c = 500;

if ($a > $b && $c < $a ) {

echo "Both conditions are true";

}

Try it Yourself »

Here are the PHP logical operators to use in if statements:

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Name** | **Description** | **Try it** |
| and | And | True if both conditions are true | Try it » |
| && | And | True if both conditions are true | Try it » |
| or | Or | True if either condition is true | Try it » |
| || | Or | True if either condition is true | Try it » |
| xor | Xor | True if either condition is true, but not both | Try it » |
| ! | Not | True if condition is not true | Try it » |

We can compare as many conditions as we like in one if statement:

Example

Check if $a is either 2, 3, 4, 5, 6, or 7:

$a = 5;

if ($a == 2 || $a == 3 || $a == 4 || $a == 5 || $a == 6 || $a == 7) {

echo "$a is a number between 2 and 7";

}

PHP if...else Statements

❮ PreviousNext ❯

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*;

}

ExampleGet your own PHP Server

Output "Have a good day!" if the current time is less than 20, and "Have a good night!" otherwise:

$t = date("H");

if ($t < "20") {

echo "Have a good day!";

} else {

echo "Have a good night!";

}

Try it Yourself »

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 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!":

$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!";

}

Try it Yourself »

PHP Shorthand if Statements

❮ PreviousNext ❯

Short Hand If

To write shorter code, you can write if statements on one line.

ExampleGet your own PHP Server

One-line if statement:

$a = 5;

if ($a < 10) $b = "Hello";

echo $b

Try it Yourself »

Short Hand If...Else

if...else statements can also be written in one line, but the syntax is a bit different.

Example

One-line if...else statement:

$a = 13;

$b = $a < 10 ? "Hello" : "Good Bye";

echo $b;

Try it Yourself »

This technique is known as **Ternary Operators**, or **Conditional Expressions**.

PHP Nested if Statement

❮ PreviousNext ❯

Nested If

You can have if statements inside if statements, this is called *nested* if statements.

ExampleGet your own PHP Server

An if inside an if:

$a = 13;

if ($a > 10) {

echo "Above 10";

if ($a > 20) {

echo " and also above 20";

} else {

echo " but not above 20";

}

}

# PHP switch Statement

❮ PreviousNext ❯

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

## The PHP switch Statement

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

### Syntax

switch (*expression*) {

case *label1:*

//*code block*

break;

case *label2:*

//*code block;*

break;

case *label3:*

//*code block*

break;

default:

//*code block*

}

This is how it works:

* The expression is evaluated once
* The value of the expression is compared with the values of each case
* If there is a match, the associated block of code is executed
* The break keyword breaks out of the switch block
* The default code block is executed if there is no match

### ExampleGet your own PHP Server

$favcolor = "red";

switch ($favcolor) {

case "red":

echo "Your favorite color is red!";

break;

case "blue":

"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!";

}

Try it Yourself »

ADVERTISEMENT

## The break Keyword

When PHP reaches a break keyword, it breaks out of the switch block.

This will stop the execution of more code, and no more cases are tested.

The last block does not need a break, the block breaks (ends) there anyway.

**Warning:** If you omit the break statement in a case that is not the last, and that case gets a match, the next case will also be executed even if the evaluation does not match the case!

### Example

What happens if we remove the break statement from case "red"?

$favcolor is red, so the code block from case "red" is executed, but since it has no break statement, the code block from case "blue" will also be executed:

$favcolor = "red";

switch ($favcolor) {

case "red":

echo "Your favorite color is red!";

case "blue":

"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!";

}

Try it Yourself »

## The default Keyword

The default keyword specifies the code to run if there is no case match:

### Example

If no cases get a match, the default block is executed:

$d = 4;

switch ($d) {

case 6:

echo "Today is Saturday";

break;

case 0:

echo "Today is Sunday";

break;

default:

echo "Looking forward to the Weekend";

}

Try it Yourself »

The default case does not have to be the last case in a switch block:

### Example

Putting  the default block elsewhere than at the end of the switch block is allowed, but not recommended.

$d = 4;

switch ($d) {

default:

echo "Looking forward to the Weekend";

break;

case 6:

echo "Today is Saturday";

break;

case 0:

echo "Today is Sunday";

}

Try it Yourself »

**Note:** If default is not the last block in the switch block, remember to end the default block with a break statement.

## Common Code Blocks

If you want multiple cases to use the same code block, you can specify the cases like this:

### Example

More than one case for each code block:

$d = 3;

switch ($d) {

case 1:

case 2:

case 3:

case 4:

case 5:

echo "The weeks feels so long!";

break;

case 6:

case 0:

echo "Weekends are the best!";

break;

default:

echo "Something went wrong";

}

# PHP Loops

❮ PreviousNext ❯

In the following chapters you will learn how to repeat code by using loops in PHP.

## PHP Loops

Often when you write code, you want the same block of code to run over and over again a certain number of times. So, instead of adding several almost equal code-lines in a script, we can use 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

The following chapters will explain and give examples of each loop type.

PHP while Loop

❮ PreviousNext ❯

The while loop - Loops through a block of code as long as the specified condition is true.

The PHP while Loop

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

ExampleGet your own PHP Server

Print $i as long as $i is less than 6:

$i = 1;

while ($i < 6) {

echo $i;

$i++;

}

Try it Yourself »

**Note:** remember to increment $i, or else the loop will continue forever.

The while loop does not run a specific number of times, but checks after each iteration if the condition is still true.

The condition does not have to be a counter, it could be the status of an operation or any condition that evaluates to either true or false.

The break Statement

With the break statement we can stop the loop even if the condition is still true:

Example

Stop the loop when $i is 3:

$i = 1;

while ($i < 6) {

if ($i == 3) break;

echo $i;

$i++;

}

Try it Yourself »

ADVERTISEMENT

The continue Statement

With the continue statement we can stop the current iteration, and continue with the next:

Example

Stop, and jump to the next iteration if $i is 3:

$i = 0;

while ($i < 6) {

$i++;

if ($i == 3) continue;

echo $i;

}

Try it Yourself »

Alternative Syntax

The while loop syntax can also be written with the endwhile statement like this

Example

Print $i as long as $i is less than 6:

$i = 1;

while ($i < 6):

echo $i;

$i++;

endwhile;

Try it Yourself »

Step 10

If you want the while loop count to 100, but only by each 10, you can increase the counter by 10 instead 1 in each iteration:

Example

Count to 100 by tens:

$i = 0;

while ($i < 100) {

$i+=10;

echo $i "<br>";

}

Try it Yourself »

PHP Exercises

PHP do while Loop

❮ PreviousNext ❯

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

The PHP do...while Loop

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

ExampleGet your own PHP Server

Print $i as long as $i is less than 6:

$i = 1;

do {

echo $i;

$i++;

} while ($i < 6);

Try it Yourself »

**Note:** In a do...while loop the condition is tested AFTER executing the statements within the loop. This means that the do...while loop will execute its statements at least once, even if the condition is false. See example below.

Let us see what happens if we set the $i variable to 8 instead of 1, before execute the same do...while loop again:

Example

Set $i = 8, then print $i as long as $i is less than 6:

$i = 8;

do {

echo $i;

$i++;

} while ($i < 6);

Try it Yourself »

The code will be executed once, even if the condition is never true.

ADVERTISEMENT

The break Statement

With the break statement we can stop the loop even if the condition is still true:

Example

Stop the loop when $i is 3:

$i = 1;

do {

if ($i == 3) break;

echo $i;

$i++;

} while ($i < 6);

Try it Yourself »

The continue Statement

With the continue statement we can stop the current iteration, and continue with the next:

Example

Stop, and jump to the next iteration if $i is 3:

$i = 0;

do {

$i++;

if ($i == 3) continue;

echo $i;

} while ($i < 6);

# PHP for Loop

❮ PreviousNext ❯

The for loop - Loops through a block of code a specified number of times.

## The PHP for Loop

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

### Syntax

for (*expression1*, *expression2*, *expression3*) {

// *code block*

}

This is how it works:

* expression1 is evaluated once
* expression2 is evaluated before each iterarion
* expression3 is evaluated after each iterarion

### ExampleGet your own PHP Server

Print the numbers from 0 to 10:

for ($x = 0; $x <= 10; $x++) {

echo "The number is: $x <br>";

}

Try it Yourself »

### Example Explained

1. The first expression, $x = 0;, is evaluated once and sets a counter to 0.
2. The second expression, $x <= 10;, is evaluated before each iteration, and the code block is only executed if this expression evaluates to true. In this example the expression is true as long as $x is less than, or equal to, 10.
3. The third expression, $x++;, is evaluated after each iteration, and in this example, the expression increases the value of $x by one at each iteration.

## The break Statement

With the break statement we can stop the loop even if the condition is still true:

### Example

Stop the loop when $i is 3:

for ($x = 0; $x <= 10; $x++) {

if ($i == 3) break;

echo "The number is: $x <br>";

}

Try it Yourself »

ADVERTISEMENT

## The continue Statement

With the continue statement we can stop the current iteration, and continue with the next:

### Example

Stop, and jump to the next iteration if $i is 3:

for ($x = 0; $x <= 10; $x++) {

if ($x == 3) continue;

echo "The number is: $x <br>";

}

Try it Yourself »

## Step 10

This example counts to 100 by tens:

### Example

for ($x = 0; $x <= 100; $x+=10) {

echo "The number is: $x <br>";

}

Try it Yourself »

## PHP Exercises

PHP foreach Loop

❮ PreviousNext ❯

The foreach loop - Loops through a block of code for each element in an array or each property in an object.

The foreach Loop on Arrays

The most common use of the foreach loop, is to loop through the items of an array.

ExampleGet your own PHP Server

Loop through the items of an indexed array:

$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $x) {

echo "$x <br>";

}

Try it Yourself »

For every loop iteration, the value of the current array element is assigned to the variabe $x. The iteration continues until it reaches the last array element.

Keys and Values

The array above is an indexed array, where the first item has the key 0, the second has the key 1, and so on.

Associative arrays are different, associative arrays use named keys that you assign to them, and when looping through associative arrays, you might want to keep the key as well as the value.

This can be done by specifying both the key and value in the foreach defintition, like this:

Example

Print both the key and the value from the $members array:

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

foreach ($members as $x => $y) {

echo "$x : $y <br>";

}

Try it Yourself »

You will learn more about arrays in the PHP Arrays chapter.

ADVERTISEMENT

The foreach Loop on Objects

The foreach loop can also be used to loop through properties of an object:

Example

Print the property names and values of the $myCar object:

class Car {

public $color;

public $model;

public function \_\_construct($color, $model) {

$this->color = $color;

$this->model = $model;

}

}

$myCar = new Car("red", "Volvo");

foreach ($myCar as $x => $y) {

echo "$x: $y <br>";

}

Try it Yourself »

You will learn more about objects in the PHP Objects and Classes chapter.

The break Statement

With the break statement we can stop the loop even if it has not reached the end:

Example

Stop the loop if $x is "blue":

$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $x) {

if ($x == "blue") break;

echo "$x <br>";

}

Try it Yourself »

The continue Statement

With the continue statement we can stop the current iteration, and continue with the next:

Example

Stop, and jump to the next iteration if $x is "blue":

$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $x) {

if ($x == "blue") continue;

echo "$x <br>";

}

Try it Yourself »

Foreach Byref

When looping through the array items, any changes done to the array item will, by default, NOT affect the original array:

Example

By default, changing an array item will not affect the original array:

$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $x) {

if ($x == "blue") $x = "pink";

}

var\_dump($colors);

Try it Yourself »

BUT, by using the & character in the foreach declaration, the array item is assigned *by reference*, which results in any changes done to the array item will also be done to the original array:

Example

By assigning the array items *by reference*, changes will affect the original array:

$colors = array("red", "green", "blue", "yellow");

foreach ($colors as &$x) {

if ($x == "blue") $x = "pink";

}

var\_dump($colors);

Try it Yourself »

Alternative Syntax

The foreach loop syntax can also be written with the endforeach statement like this

Example

Loop through the items of an indexed array:

$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $x) :

echo "$x <br>";

endforeach;

PHP Break

❮ PreviousNext ❯

The break statement can be used to jump out of different kind of loops.

Break in For loop

The break statement can be used to jump out of a for loop.

ExampleGet your own PHP Server

Jump out of the loop when $x is 4:

for ($x = 0; $x < 10; $x++) {

if ($x == 4) {

break;

}

echo "The number is: $x <br>";

}

Try it Yourself »

Break in While Loop

The break statement can be used to jump out of a while loop.

Break Example

$x = 0;

while($x < 10) {

if ($x == 4) {

break;

}

echo "The number is: $x <br>";

$x++;

}

Try it Yourself »

ADVERTISEMENT

Break in Do While Loop

The break statement can be used to jump out of a do...while loop.

Example

Stop the loop when $i is 3:

$i = 1;

do {

if ($i == 3) break;

echo $i;

$i++;

} while ($i < 6);

Try it Yourself »

Break in For Each Loop

The break statement can be used to jump out of a foreach loop.

Example

Stop the loop if $x is "blue":

$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $x) {

if ($x == "blue") break;

echo "$x <br>";

}

PHP Continue

❮ PreviousNext ❯

The continue statement can be used to jump out of the current iteration of a loop, and continue with the next.

Continue in For Loops

The continue statement stops the current iteration in the for loop and continue with the next.

ExampleGet your own PHP Server

Move to next iteration if $x = 4:

for ($x = 0; $x < 10; $x++) {

if ($x == 4) {

continue;

}

echo "The number is: $x <br>";

}

Try it Yourself »

Continue in While Loop

The continue statement stops the current iteration in the while loop and continue with the next.

Continue Example

Move to next iteration if $x = 4:

$x = 0;

while($x < 10) {

if ($x == 4) {

continue;

}

echo "The number is: $x <br>";

$x++;

}

Try it Yourself »

ADVERTISEMENT

Continue in Do While Loop

The continue statement stops the current iteration in the do...while loop and continue with the next.

Example

Stop, and jump to the next iteration if $i is 3:

$i = 0;

do {

$i++;

if ($i == 3) continue;

echo $i;

} while ($i < 6);

Try it Yourself »

Continue in For Each Loop

The continue statement stops the current iteration in the foreach loop and continue with the next.

Example

Stop, and jump to the next iteration if $x is "blue":

$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $x) {

if ($x == "blue") continue;

echo "$x <br>";

}

# PHP Functions

❮ PreviousNext ❯

The real power of PHP comes from its functions.

PHP has more than 1000 built-in functions, and in addition you can create your own custom functions.

## PHP Built-in Functions

PHP has over 1000 built-in functions that can be called directly, from within a script, to perform a specific task.

Please check out our PHP reference for a complete overview of the PHP built-in functions.

## PHP User Defined Functions

Besides the built-in PHP functions, it is possible to create your own functions.

* A function is a block of statements that can be used repeatedly in a program.
* A function will not execute automatically when a page loads.
* A function will be executed by a call to the function.

## Create a Function

A user-defined function declaration starts with the keyword function, followed by the name of the function:

### ExampleGet your own PHP Server

function myMessage() {

echo "Hello world!";

}

**Note:** A function name must start with a letter or an underscore. Function names are NOT case-sensitive.

**Tip:** Give the function a name that reflects what the function does!

## Call a Function

To call the function, just write its name followed by parentheses ():

### Example

function myMessage() {

echo "Hello world!";

}

myMessage();

Try it Yourself »

In our example, we create a function named myMessage().

The opening curly brace { indicates the beginning of the function code, and the closing curly brace } indicates the end of the function.

The function outputs "Hello world!".

ADVERTISEMENT

## PHP Function Arguments

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

Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

The following example has a function with one argument ($fname). When the familyName() function is called, we also pass along a name, e.g. ("Jani"), and the name is used inside the function, which outputs several different first names, but an equal last name:

### Example

function familyName($fname) {

echo "$fname Refsnes.<br>";

}

familyName("Jani");

familyName("Hege");

familyName("Stale");

familyName("Kai Jim");

familyName("Borge");

Try it Yourself »

The following example has a function with two arguments ($fname, $year):

### Example

function familyName($fname, $year) {

echo "$fname Refsnes. Born in $year <br>";

}

familyName("Hege", "1975");

familyName("Stale", "1978");

familyName("Kai Jim", "1983");

Try it Yourself »

## PHP 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:

### Example

function setHeight($minheight = 50) {

echo "The height is : $minheight <br>";

}

setHeight(350);

setHeight(); // will use the default value of 50

setHeight(135);

setHeight(80);

Try it Yourself »

## PHP Functions - Returning values

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

### Example

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);

Try it Yourself »

## Passing Arguments by Reference

In PHP, arguments are usually passed by value, which means that a copy of the value is used in the function and the variable that was passed into the function cannot be changed.

When a function argument is passed by reference, changes to the argument also change the variable that was passed in. To turn a function argument into a reference, the & operator is used:

### Example

Use a pass-by-reference argument to update a variable:

function add\_five(&$value) {

$value += 5;

}

$num = 2;

add\_five($num);

echo $num;

Try it Yourself »

## Variable Number of Arguments

By using the ... operator in front of the function parameter, the function accepts an unknown number of arguments. This is also called a variadic function.

The variadic function argument becomes an array.

### Example

A function that do not know how many arguments it will get:

function sumMyNumbers(...$x) {

$n = 0;

$len = count($x);

for($i = 0; $i < $len; $i++) {

$n += $x[$i];

}

return $n;

}

$a = sumMyNumbers(5, 2, 6, 2, 7, 7);

echo $a;

Try it Yourself »

You can only have one argument with variable length, and it has to be the last argument.

### Example

The variadic argument must be the last argument:

function myFamily($lastname, ...$firstname) {

txt = "";

$len = count($firstname);

for($i = 0; $i < $len; $i++) {

$txt = $txt."Hi, $firstname[$i] $lastname.<br>";

}

return $txt;

}

$a = myFamily("Doe", "Jane", "John", "Joey");

echo $a;

Try it Yourself »

If the variadic argument is not the last argument, you will get an error.

### Example

Having the ... operator on the first of two arguments, will raise an error:

function myFamily(...$firstname, $lastname) {

$txt = "";

$len = count($firstname);

for($i = 0; $i < $len; $i++) {

$txt = $txt."Hi, $firstname[$i] $lastname.<br>";

}

return $txt;

}

$a = myFamily("Doe", "Jane", "John", "Joey");

echo $a;

Try it Yourself »

## PHP is a Loosely Typed Language

In the examples above, notice that we did not have to tell PHP which data type the variable is.

PHP automatically associates a data type to the variable, depending on its value. Since the data types are not set in a strict sense, you can do things like adding a string to an integer without causing an error.

In PHP 7, type declarations were added. This gives us an option to specify the expected data type when declaring a function, and by adding the strict declaration, it will throw a "Fatal Error" if the data type mismatches.

In the following example we try to send both a number and a string to the function without using strict:

### Example

function addNumbers(int $a, int $b) {

return $a + $b;

}

echo addNumbers(5, "5 days");

// since strict is NOT enabled "5 days" is changed to int(5), and it will return 10

Try it Yourself »

To specify strict we need to set declare(strict\_types=1);. This must be on the very first line of the PHP file.

In the following example we try to send both a number and a string to the function, but here we have added the strict declaration:

### Example

**<?php** declare(strict\_types=1); // strict requirement

function addNumbers(int $a, int $b) {

return $a + $b;

}

echo addNumbers(5, "5 days");

// since strict is enabled and "5 days" is not an integer, an error will be thrown

**?>**

Try it Yourself »

The strict declaration forces things to be used in the intended way.

## PHP Return Type Declarations

PHP 7 also supports Type Declarations for the return statement. Like with the type declaration for function arguments, by enabling the strict requirement, it will throw a "Fatal Error" on a type mismatch.

To declare a type for the function return, add a colon ( : ) and the type right before the opening curly ( { )bracket when declaring the function.

In the following example we specify the return type for the function:

### Example

**<?php** declare(strict\_types=1); // strict requirement

function addNumbers(float $a, float $b) : float {

return $a + $b;

}

echo addNumbers(1.2, 5.2);

**?>**

Try it Yourself »

You can specify a different return type, than the argument types, but make sure the return is the correct type:

### Example

**<?php** declare(strict\_types=1); // strict requirement

function addNumbers(float $a, float $b) : int {

return (int)($a + $b);

}

echo addNumbers(1.2, 5.2);

# PHP Arrays

❮ PreviousNext ❯

An array stores multiple values in one single variable:

### ExampleGet your own PHP Server

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

Try it Yourself »

## What is an Array?

An array is a special variable that can hold many values under a single name, and you can access the values by referring to an index number or name.

## PHP Array Types

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

## Working With Arrays

In this tutorial you will learn how to work with arrays, including:

* Create Arrays
* Access Arrays
* Update Arrays
* Add Array Items
* Remove Array Items
* Sort Arrays

## Array Items

Array items can be of any data type.

The most common are strings and numbers (int, float), but array items can also be objects, functions or even arrays.

You can have different data types in the same array.

### Example

Array items of four different data types:

$myArr = array("Volvo", 15, ["apples", "bananas"], myFunction);

Try it Yourself »

## Array Functions

The real strength of PHP arrays are the built-in array functions, like the count() function for counting array items:

### Example

How many items are in the $cars array:

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

echo count($cars);

Try it Yourself »

For a complete reference of all array functions, go to our complete PHP Array Reference.

PHP Array Functions

❮ PreviousNext ❯

PHP Array Introduction

The array functions allow you to access and manipulate arrays.

Simple and multi-dimensional arrays are supported.

Installation

The array functions are part of the PHP core. There is no installation needed to use these functions.

PHP Array Functions

|  |  |
| --- | --- |
| **Function** | **Description** |
| array() | Creates an array |
| array\_change\_key\_case() | Changes all keys in an array to lowercase or uppercase |
| array\_chunk() | Splits an array into chunks of arrays |
| array\_column() | Returns the values from a single column in the input array |
| array\_combine() | Creates an array by using the elements from one "keys" array and one "values" array |
| array\_count\_values() | Counts all the values of an array |
| array\_diff() | Compare arrays, and returns the differences (compare values only) |
| array\_diff\_assoc() | Compare arrays, and returns the differences (compare keys and values) |
| array\_diff\_key() | Compare arrays, and returns the differences (compare keys only) |
| array\_diff\_uassoc() | Compare arrays, and returns the differences (compare keys and values, using a user-defined key comparison function) |
| array\_diff\_ukey() | Compare arrays, and returns the differences (compare keys only, using a user-defined key comparison function) |
| array\_fill() | Fills an array with values |
| array\_fill\_keys() | Fills an array with values, specifying keys |
| array\_filter() | Filters the values of an array using a callback function |
| array\_flip() | Flips/Exchanges all keys with their associated values in an array |
| array\_intersect() | Compare arrays, and returns the matches (compare values only) |
| array\_intersect\_assoc() | Compare arrays and returns the matches (compare keys and values) |
| array\_intersect\_key() | Compare arrays, and returns the matches (compare keys only) |
| array\_intersect\_uassoc() | Compare arrays, and returns the matches (compare keys and values, using a user-defined key comparison function) |
| array\_intersect\_ukey() | Compare arrays, and returns the matches (compare keys only, using a user-defined key comparison function) |
| array\_key\_exists() | Checks if the specified key exists in the array |
| array\_keys() | Returns all the keys of an array |
| array\_map() | Sends each value of an array to a user-made function, which returns new values |
| array\_merge() | Merges one or more arrays into one array |
| array\_merge\_recursive() | Merges one or more arrays into one array recursively |
| array\_multisort() | Sorts multiple or multi-dimensional arrays |
| array\_pad() | Inserts a specified number of items, with a specified value, to an array |
| array\_pop() | Deletes the last element of an array |
| array\_product() | Calculates the product of the values in an array |
| array\_push() | Inserts one or more elements to the end of an array |
| array\_rand() | Returns one or more random keys from an array |
| array\_reduce() | Returns an array as a string, using a user-defined function |
| array\_replace() | Replaces the values of the first array with the values from following arrays |
| array\_replace\_recursive() | Replaces the values of the first array with the values from following arrays recursively |
| array\_reverse() | Returns an array in the reverse order |
| array\_search() | Searches an array for a given value and returns the key |
| array\_shift() | Removes the first element from an array, and returns the value of the removed element |
| array\_slice() | Returns selected parts of an array |
| array\_splice() | Removes and replaces specified elements of an array |
| array\_sum() | Returns the sum of the values in an array |
| array\_udiff() | Compare arrays, and returns the differences (compare values only, using a user-defined key comparison function) |
| array\_udiff\_assoc() | Compare arrays, and returns the differences (compare keys and values, using a built-in function to compare the keys and a user-defined function to compare the values) |
| array\_udiff\_uassoc() | Compare arrays, and returns the differences (compare keys and values, using two user-defined key comparison functions) |
| array\_uintersect() | Compare arrays, and returns the matches (compare values only, using a user-defined key comparison function) |
| array\_uintersect\_assoc() | Compare arrays, and returns the matches (compare keys and values, using a built-in function to compare the keys and a user-defined function to compare the values) |
| array\_uintersect\_uassoc() | Compare arrays, and returns the matches (compare keys and values, using two user-defined key comparison functions) |
| array\_unique() | Removes duplicate values from an array |
| array\_unshift() | Adds one or more elements to the beginning of an array |
| array\_values() | Returns all the values of an array |
| array\_walk() | Applies a user function to every member of an array |
| array\_walk\_recursive() | Applies a user function recursively to every member of an array |
| arsort() | Sorts an associative array in descending order, according to the value |
| asort() | Sorts an associative array in ascending order, according to the value |
| compact() | Create array containing variables and their values |
| count() | Returns the number of elements in an array |
| current() | Returns the current element in an array |
| each() | Deprecated from PHP 7.2. Returns the current key and value pair from an array |
| end() | Sets the internal pointer of an array to its last element |
| extract() | Imports variables into the current symbol table from an array |
| in\_array() | Checks if a specified value exists in an array |
| key() | Fetches a key from an array |
| krsort() | Sorts an associative array in descending order, according to the key |
| ksort() | Sorts an associative array in ascending order, according to the key |
| list() | Assigns variables as if they were an array |
| natcasesort() | Sorts an array using a case insensitive "natural order" algorithm |
| natsort() | Sorts an array using a "natural order" algorithm |
| next() | Advance the internal array pointer of an array |
| pos() | Alias of current() |
| prev() | Rewinds the internal array pointer |
| range() | Creates an array containing a range of elements |
| reset() | Sets the internal pointer of an array to its first element |
| rsort() | Sorts an indexed array in descending order |
| shuffle() | Shuffles an array |
| sizeof() | Alias of count() |
| sort() | Sorts an indexed array in ascending order |
| uasort() | Sorts an array by values using a user-defined comparison function and maintains the index association |
| uksort() | Sorts an array by keys using a user-defined comparison function |
| usort() | Sorts an array by values using a user-defined comparison function |

PHP Indexed Arrays

❮ PreviousNext ❯

PHP Indexed Arrays

In indexed arrays each item has an index number.

By default, the first item has index 0, the second item has item 1, etc.

ExampleGet your own PHP Server

Create and display an indexed array:

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

var\_dump($cars);

Try it Yourself »

Access Indexed Arrays

To access an array item you can refer to the index number.

Example

Display the first array item:

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

echo $cars[0];

Try it Yourself »

Change Value

To change the value of an array item, use the index number:

Example

Change the value of the second item:

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

$cars[1] = "Ford";

var\_dump($cars);

Try it Yourself »

Loop Through an Indexed Array

To loop through and print all the values of an indexed array, you could use a foreach loop, like this:

Example

Display all array items:

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

foreach ($cars as $x) {

echo "$x <br>";

}

Try it Yourself »

For a complete reference of all array functions, go to our complete PHP Array Reference.

ADVERTISEMENT

Index Number

The key of an indexed array is a number, by default the first item is 0 and the second is 1 etc., but there are exceptions.

New items get the next index number, meaning one higher than the *highest existing index*.

So if you have an array like this:

$cars[0] = "Volvo";

$cars[1] = "BMW";

$cars[2] = "Toyota";

And if you use the array\_push() function to add a new item, the new item will get the index 3:

Example

array\_push($cars, "Ford");

var\_dump($cars);

Try it Yourself »

But if you have an array with random index numbers, like this:

$cars[5] = "Volvo";

$cars[7] = "BMW";

$cars[14] = "Toyota";

And if you use the array\_push() function to add a new item, what will be the index number of the new item?

Example

array\_push($cars, "Ford");

var\_dump($cars);

PHP Associative Arrays

❮ PreviousNext ❯

PHP Associative Arrays

Associative arrays are arrays that use named keys that you assign to them.

ExampleGet your own PHP Server

$car = array("brand"=>"Ford", "model"=>"Mustang", "year"=>1964);

var\_dump($car);

Try it Yourself »

Access Associative Arrays

To access an array item you can refer to the key name.

Example

Display the model of the car:

$car = array("brand"=>"Ford", "model"=>"Mustang", "year"=>1964);

echo $car["model"];

Try it Yourself »

Change Value

To change the value of an array item, use the key name:

Example

Change the year item:

$car = array("brand"=>"Ford", "model"=>"Mustang", "year"=>1964);

$car["year"] = 2024;

var\_dump($car);

Try it Yourself »

ADVERTISEMENT

Loop Through an Associative Array

To loop through and print all the values of an associative array, you could use a foreach loop, like this:

Example

Display all array items, keys and values:

$car = array("brand"=>"Ford", "model"=>"Mustang", "year"=>1964);

foreach ($car as $x => $y) {

echo "$x: $y <br>";

}

Try it Yourself »

For a complete reference of all array functions, go to our complete PHP Array Reference.

PHP Create Arrays

❮ PreviousNext ❯

Create Array

You can create arrays by using the array() function:

ExampleGet your own PHP Server

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

Try it Yourself »

You can also use a shorter syntax by using the [] brackets:

Example

$cars = ["Volvo", "BMW", "Toyota"];

Try it Yourself »

Multiple Lines

Line breaks are not important, so an array declaration can span multiple lines:

Example

$cars = [

"Volvo",

"BMW",

"Toyota"

];

Try it Yourself »

Trailing Comma

A comma after the last item is allowed:

Example

$cars = [

"Volvo",

"BMW",

"Toyota",

];

Try it Yourself »

Array Keys

When creating indexed arrays the keys are given automatically, starting at 0 and increased by 1 for each item, so the array above could also be created with keys:

Example

$cars = [

0 => "Volvo",

1 => "BMW",

2 =>"Toyota"

];

Try it Yourself »

As you can see, indexed arrays are the same as associative arrays, but associative arrays have names instead of numbers:

Example

$myCar = [

"brand" => "Ford",

"model" => "Mustang",

"year" => 1964

];

Try it Yourself »

Declare Empty Array

You can declare an empty array first, and add items to it later:

Example

$cars = [];

$cars[0] = "Volvo";

$cars[1] = "BMW";

$cars[2] = "Toyota";

Try it Yourself »

The same goes for associative arrays, you can declare the array first, and then add items to it:

Example

$myCar = [];

$myCar["brand"] = "Ford";

$myCar["model"] = "Mustang";

$myCar["year"] = 1964;

Try it Yourself »

Mixing Array Keys

You can have arrays with both indexed and named keys:

Example

$myArr = [];

$myArr[0] = "apples";

$myArr[1] = "bananas";

$myArr["fruit"] = "cherries";

PHP Access Arrays

❮ PreviousNext ❯

Access Array Item

To access an array item, you can refer to the index number for indexed arrays, and the key name for associative arrays.

ExampleGet your own PHP Server

Access an item by referring to its index number:

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

echo $cars[2];

Try it Yourself »

**Note:**The first item has index 0.

To access items from an **associative array**, use the key name:

Example

Access an item by referring to its key name:

$cars = array("brand" => "Ford", "model" => "Mustang", "year" => 1964);

echo $cars["year"];

Try it Yourself »

Double or Single Quotes

You can use both double and single quotes when accessing an array:

Example

echo $cars["model"];

echo $cars['model'];

Try it Yourself »

Excecute a Function Item

Array items can be of any data type, including function.

To execute such a function, use the index number followed by parentheses ():

Example

Execute a function item:

function myFunction() {

echo "I come from a function!";

}

$myArr = array("Volvo", 15, myFunction);

$myArr[2]();

Try it Yourself »

Use the key name when the function is an item in a associative array:

Example

Execute function by referring to the key name:

function myFunction() {

echo "I come from a function!";

}

$myArr = array("car" => "Volvo", "age" => 15, "message" => myFunction);

$myArr["message"]();

Try it Yourself »

Loop Through an Associative Array

To loop through and print all the values of an associative array, you can use a foreach loop, like this:

Example

Display all array items, keys and values:

$car = array("brand"=>"Ford", "model"=>"Mustang", "year"=>1964);

foreach ($car as $x => $y) {

echo "$x: $y <br>";

}

Try it Yourself »

Loop Through an Indexed Array

To loop through and print all the values of an indexed array, you can use a foreach loop, like this:

Example

Display all array items:

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

foreach ($cars as $x) {

echo "$x <br>";

}

PHP Update Array Items

❮ PreviousNext ❯

Update Array Item

To update an existing array item, you can refer to the index number for indexed arrays, and the key name for associative arrays.

ExampleGet your own PHP Server

Change the second array item from "BMW" to "Ford":

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

$cars[1] = "Ford";

Try it Yourself »

**Note:**The first item has index 0.

To update items from an **associative array**, use the key name:

Example

Update the year to 2024:

$cars = array("brand" => "Ford", "model" => "Mustang", "year" => 1964);

$cars["year"] = 2024;

Try it Yourself »

Update Array Items in a Foreach Loop

There are different techniques to use when changing item values in a foreach loop.

One way is to insert the & character in the assignment to assign the item value by reference, and thereby making sure that any changes done with the array item inside the loop will be done to the original array:

Example

Change ALL item values to "Ford":

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

foreach ($cars as &$x) {

$x = "Ford";

}

unset($x);

var\_dump($cars);

Try it Yourself »

**Note:**Remember to add the unset() function after the loop.

Without the unset($x) function, the $x variable will remain as a reference to the last array item.

To demonstrate this, see what happens when we change the value of $x after the foreach loop:

Example

Demonstrate the consequence of forgetting the unset() function:

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

foreach ($cars as &$x) {

$x = "Ford";

}

$x = "ice cream";

var\_dump($cars);

PHP Add Array Items

❮ PreviousNext ❯

Add Array Item

To add items to an existing array, you can use the bracket [] syntax.

ExampleGet your own PHP Server

Add one more item to the fruits array:

$fruits = array("Apple", "Banana", "Cherry");

$fruits = ["Orange"];

Try it Yourself »

Associative Arrays

To add items to an associative array, or key/value array, use brackets [] for the key, and assign value with the = operator.

Example

Add one item to the car array:

$cars = array("brand" => "Ford", "model" => "Mustang");

$cars["color"] = "Red";

Try it Yourself »

Add Multiple Array Items

To add multiple items to an existing array, use the array\_push() function.

Example

Add three item to the fruits array:

$fruits = array("Apple", "Banana", "Cherry");

array\_push($fruits, "Orange", "Kiwi", "Lemon");

Try it Yourself »

Add Multiple Items to Associative Arrays

To add multiple items to an existing array, you can use the += operator.

Example

Add two items to the cars array:

$cars = array("brand" => "Ford", "model" => "Mustang");

$cars += ["color" => "red", "year" => 1964];

PHP Delete Array Items

❮ PreviousNext ❯

Remove Array Item

To remove an existing item from an array, you can use the array\_splice() function.

With the array\_splice() function you specify the index (where to start) and how many items you want to delete.

ExampleGet your own PHP Server

Remove the second item:

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

array\_splice($cars, 1, 1);

Try it Yourself »

After the deletion, the array gets reindexed automtically, starting at index 0.

Using the unset Function

You can also use the unset() function to delete existing array items.

**Note:**The unset() function does not re-arrange the indexes, meaning that after deletion the array will no longer contain the missing indexes.

Example

Remove the second item:

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

unset($cars[1]);

Try it Yourself »

Remove Multiple Array Items

To remove multiple items, the array\_splice() function takes a length parameter that allows you to specify the number of items to delete.

Example

Remove 2 items, starting a the second item (index 1):

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

array\_splice($cars, 1, 2);

Try it Yourself »

The unset() function takes a unlimited number of arguments, and can therefor be used to delete multiple array items:

Example

Remove the first and the second item:

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

unset($cars[0], $cars[1]);

Try it Yourself »

Remove Item From an Associative Array

To remove items from an associative array, you can use the unset() function.

Specify the key of the item you want to delete.

Example

Remove the "model":

$cars = array("brand" => "Ford", "model" => "Mustang", "year" => 1964);

unset($cars["model"]);

Try it Yourself »

Using the array\_diff Function

You can also use the array\_diff() function to remove items from an associative array.

This function returns a new array, without the specified items.

Example

Create a new array, without "Mustang" and "1964":

$cars = array("brand" => "Ford", "model" => "Mustang", "year" => 1964);

$newarray = array\_diff($cars, ["Mustang", 1964]);

Try it Yourself »

**Note:**The array\_diff() function takes *values*as parameters, and not *keys*.

Remove the Last Item

The array\_pop() function removes the last item of an array.

Example

Remove the last item:

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

array\_pop($cars);

Try it Yourself »

Remove the First Item

The array\_shift() function removes the first item of an array.

Example

Remove the first item:

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

array\_shift($cars);