PHP

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.

▼ PHP Syntax

```
A PHP script starts with <?php and ends with ?>:
<?php
// PHP code goes here
?>
----------------------------------------------------------------
<!DOCTYPE html>
<html>
<body>
// In PHP, keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-d
efined functions are not case-sensitive
<?php
ECHO "Hello World!<br>";
echo "Hello World!<br>";
EcHo "Hello World!<br>";
?>
-------------------------------------------------------------------
```

```
//Note: However; all variable names are case-sensitive!
<?php
$color = "red";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?>
// output
My car is red
My house is
My boat is

</body>
</html>
```

▼ Comments in PHP

```
<!DOCTYPE html>
<html>
<body>

<?php
// This is a single-line comment

# This is also a single-line comment

/*
This is a multiple-lines comment block
that spans over multiple
lines
*/

// You can also use comments to leave out parts of a code line
$x = 5 /* + 15 */ + 5;
echo $x;
// output : 10
?>

</body>
</html>
```

▼ Creating (Declaring) PHP Variables

```
<?php
//In PHP, a variable starts with the $ sign, followed by the name of the variable:
$txt = "Hello world!";
$x = 5;
```

```
$y = 10.5;
?>
-----------------------------------------------------
/*
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, a
nd _ )
Variable names are case-sensitive ($age and $AGE are two different variables)
*/
-----------------------------------------------
<?php
$txt = "W3Schools.com";
echo "I love $txt!";
//output; I love W3Schools.com
?>
=
<?php
$txt = "W3Schools.com";
echo "I love " . $txt . "!";
?>
=
<?php
$x = 5;
$y = 5;
echo $x + $y;
?>
```

▼ 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 scope
------------------
<?php
$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>";
--------------------
/* output :
Variable x inside function is:

Variable x outside function is: 5
*/
?>
-----------------------------------------------
Local Scope
------------
<?php
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>";
--------------------
output ;
Variable x inside function is: 5

Variable x outside function is:
?>
-------------------------------------------------------------
PHP The global Keyword
---------------
<?php
$x = 5;
$y = 10;

function myTest() {
  global $x, $y;
  $y = $x + $y;
}

myTest();
echo $y; // outputs 15
?>
=
<?php
$x = 5;
$y = 10;

function myTest() {
```

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

myTest();
echo $y; // outputs 15
?>
-------------------------------------------------------
PHP The static Keyword
------------------
<?php
function myTest() {
  static $x = 0;
  echo $x;
  $x++;
}

myTest();
myTest();
myTest();
// output 012
?>
```

▼ 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`.

```
echo statement
---------------
The echo statement can be used with or without parentheses: echo or echo().
<?php
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.";
?>
--------------------------
<?php
$txt1 = "Learn PHP";
$txt2 = "W3Schools.com";
$x = 5;
$y = 4;
```

```
echo "<h2>" . $txt1 . "</h2>";
echo "Study PHP at " . $txt2 . "<br>";
echo $x + $y;
?>
output
## Learn PHP

Study PHP at W3Schools.com
9
-----------------------------------------------------
The PHP print Statement
-----------------------------
<?php
print "<h2>PHP is Fun!</h2>";
print "Hello world!<br>";
print "I'm about to learn PHP!";
?>
---------------------------
<?php
$txt1 = "Learn PHP";
$txt2 = "W3Schools.com";
$x = 5;
$y = 4;

print "<h2>" . $txt1 . "</h2>";
print "Study PHP at " . $txt2 . "<br>";
print $x + $y;
?>
--------------------------
```

▼ PHP Data Types

PHP supports the following data types:

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

```
//  PHP String  //
---------------------------
<?php
$x = "Hello world!";
$y = 'Hello world!';

echo $x;
echo "<br>";
echo $y;
?>
-------------------------
//   PHP Integer   //

# An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,64
7.
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
# the following example $x is an integer. The PHP var_dump() function returns the data
type and value:
<?php
$x = 5985;
var_dump($x);
?>
-------------------------------------------------
// PHP Float //
------------------
A float (floating point number) is a number with a decimal point or a number in expone
ntial form.
<?php
$x = 10.365;
var_dump($x);
?>
---------------------------------------------------------
// PHP Boolean //
---------------------------
A Boolean represents two possible states: TRUE or FALSE.
<?
$x = true;
$y = false;
?>
-----------------------------------------------------------
// PHP Array //
-------------------------
An array stores multiple values in one single variable.
<?php
$cars = array("Volvo","BMW","Toyota");
var_dump($cars);
```

```
?>
------------------------------------------------------------
// 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.

<?php
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("black", "Volvo");
echo $myCar -> message();
echo "<br>";
$myCar = new Car("red", "Toyota");
echo $myCar -> message();
?>
output ;
My car is a black Volvo!
My car is a red Toyota!
---------------------------------------------------------------
// PHP NULL Value //
----------------------------
<?php
$x = "Hello world!";
$x = null;
var_dump($x); //output NULL
?>
-----------------------------------------------
```

▼ PHP String Functions

```
// strlen() - Return the Length of a String //
<?php
echo strlen("Hello world!"); // outputs 12
?>
--------------------------------
// str_word_count() - Count Words in a String //
<?php
echo str_word_count("Hello world!"); // outputs 2
```

```
?>
----------------------------
// strrev() - Reverse a String //
<?php
echo strrev("Hello world!"); // outputs !dlrow olleH
?>
-----------------------------------
// strpos() - Search For a Text Within a String //
<?php
echo strpos("Hello world!", "world"); // outputs 6
?>
----------------------------------------
// str_replace() - Replace Text Within a String //
<?php
echo str_replace("world", "Dolly", "Hello world!"); // outputs Hello Dolly!
?>
-------------------------------------------
```

▼ PHP Numbers

One thing to notice about PHP is that it provides automatic data type conversion.

So, if you assign an integer value to a variable, the type of that variable will automatically be an integer. Then, if you assign a string to the same variable, the type will change to a string.

This automatic conversion can sometimes break your code.

```
// PHP Integers //
-------------------------
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 :
<?php
$x = 5985;
var_dump(is_int($x));

$x = 59.85;
var_dump(is_int($x));
```

```
?>
----------------------------
// PHP Floats //
--------------
is_float()
is_double() - alias of is_float()
example :
<?php
$x = 10.365;
var_dump(is_float($x));
?>
----------------------------------------
// PHP Infinity //
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 :
<?php
$x = 1.9e411;
var_dump($x);
?>
---------------------------
//  PHP NaN  //
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 :
<?php
$x = acos(8);
var_dump($x);
?>
---------------------------------------
//  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 other
wise.
Example :
<?php
$x = 5985;
var_dump(is_numeric($x));

$x = "5985";
var_dump(is_numeric($x));

$x = "59.85" + 100;
var_dump(is_numeric($x));

$x = "Hello";
var_dump(is_numeric($x));
?>
-----------------------------------------------
```

```
//  PHP Casting Strings and Floats to Integers  //

The (int), (integer), or intval() function are often used to convert a value to an int
eger.

<?php
// 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;
?>
---------------------------------------------
```

▼ PHP Math

```
//  PHP pi() Function  //
The pi() function returns the value of PI:
<?php
echo(pi()); // returns 3.1415926535898
?>
-----------------------------------------------
//  PHP min() and max() Functions  //
The min() and max() functions can be used to find the lowest or highest value in a lis
t of arguments:
<?php
echo(min(0, 150, 30, 20, -8, -200));  // returns -200
echo(max(0, 150, 30, 20, -8, -200));  // returns 150
?>
-----------------------------------------------
//  PHP abs() Function  //
The abs() function returns the absolute (positive) value of a number:
<?php
echo(abs(-6.7));  // returns 6.7
?>
--------------------------------------
// PHP sqrt() Function  //
The sqrt() function returns the square root of a number:
<?php
echo(sqrt(64));  // returns 8
?>
-----------------------------------------------
// PHP round() Function  //
The round() function rounds a floating-point number to its nearest integer:
```

```
<?php
echo(round(0.60));  // returns 1
echo(round(0.49));  // returns 0
?>
-------------------------------------------------
//  Random Numbers  //
---------------------
The rand() function generates a random number:
<?php
echo(rand());
echo(rand(10, 100));
?>
```

▼ PHP Constants

Note:

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

```
// 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. Defa
ult is false
<?php
define("GREETING", "Welcome to W3Schools.com!");
echo GREETING;
?>
<?php
define("GREETING", "Welcome to W3Schools.com!", true);
echo greeting;
?>
-------------------------------------------------
//  PHP Constant Arrays  //
<?php
define("cars", [
  "Alfa Romeo",
  "BMW",
  "Toyota"
]);
echo cars[0];
?>
----------------------------------------
// Constants are Global  //
<?php
define("GREETING", "Welcome to W3Schools.com!");
```

```
function myTest() {
  echo GREETING;
}

myTest();
?>
-----------------------------------------------
```

▼ PHP Operators

PHP Conditional Assignment Operators

PHP Array Operators

PHP String Operators

PHP Logical Operators

PHP Increment / Decrement Operators

PHP Comparison Operators

PHP Assignment Operators

PHP Arithmetic Operators

▼ PHP if...else...elsef Statements

```
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  //
Syntax
if (condition) {
  code to be executed if condition is true;
}
----------------------
Example :

$t = date("H");
```

```
if ($t < "20") {
  echo "Have a good day!";
}
-------------------------------------------------------
PHP - The if...else Statement
Syntax :

if (condition) {
  code to be executed if condition is true;
} else {
  code to be executed if condition is false;
}
-----------------------------------------------
PHP - The if...elseif...else Statement
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;
}
```

▼ PHP switch Statement

```
                Syntax :
switch (n) {
  case label1:
    code to be executed if n=label1;
    break;
  case label2:
    code to be executed if n=label2;
    break;
  case label3:
    code to be executed if n=label3;
    break;
    ...
  default:
    code to be executed if n is different from all labels;
}

               Example :

$favcolor = "red";

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

```
    echo "Your favorite color is blue!";
    break;
  case "green":
    echo "Your favorite color is green!";
    break;
  default:
    echo "Your favorite color is neither red, blue, nor green!";
}
```

▼ PHP Loops

In PHP, we have the following loop types:

- `while` - loops through a block of code as long as the specified condition is true
- `do...while` - loops through a block of code once, and then repeats the loop as long as the specified condition is true
- `for` - loops through a block of code a specified number of times
- `foreach` - loops through a block of code for each element in an array

```
      while
            Syntax
while (condition is true) {
  code to be executed;
}
            Examble :
$x = 1;

while($x <= 5) {
  echo "The number is: $x <br>";
  $x++;
}
--------------------------------------------------
The PHP do...while Loop
              Syntax
do {
  code to be executed;
} while (condition is true);
            Examble :
$x = 1;

do {
  echo "The number is: $x <br>";
  $x++;
} while ($x <= 5);
-------------------------------------------
The PHP for Loop
```

```
              Syntax
for (init counter; test counter; increment counter) {
  code to be executed for each iteration;
}
              Examble
for ($x = 0; $x <= 10; $x++) {
  echo "The number is: $x <br>";
}
-------------------------------------------------------
The PHP foreach Loop
              Syntax
foreach ($array as $value) {
  code to be executed;
}
            Examble :
$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $value) {
  echo "$value <br>";
}
            Examble :
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

foreach($age as $x => $val) {
  echo "$x = $val<br>";
}
-------------------------------------------------------------
PHP Break and Continue
                PHP Break
            Examble:
for ($x = 0; $x < 10; $x++) {
  if ($x == 4) {
    break;
  }
  echo "The number is: $x <br>";
}
                PHP Continue
            Examble:
for ($x = 0; $x < 10; $x++) {
  if ($x == 4) {
    continue;
  }
  echo "The number is: $x <br>";
}
```

▼ PHP Functions

```
PHP Built-in Functions
PHP has over 1000 built-in functions that can be called directly, from within a scrip
t, to perform a specific task.
```

```
            Syntax
function functionName() {
  code to be executed;
}
            Examble:
function writeMsg() {
  echo "Hello world!";
}

writeMsg(); // call the function
----------------------------------------

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
---------------------------------------------------
declare(strict_types=1); // strict requirement
function setHeight(int $minheight = 50) {
  echo "The height is : $minheight <br>";
}

setHeight(350);
setHeight(); // will use the default value of 50
setHeight(135);
setHeight(80);
---------------------------------------------------------------
declare(strict_types=1); // strict requirement
function addNumbers(float $a, float $b) : float {
  return $a + $b;
}
echo addNumbers(1.2, 5.2);
-------------------------------------------------------
            Passing Arguments by Reference
function add_five(&$value) {
  $value += 5;
}

$num = 2;
add_five($num);
echo $num;
```

▼ PHP Arrays

```
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
----------------------------------
Create an Array in PHP
```

```
In PHP, the array() function is used to create an array:

array();
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
-----------------------------------------------------------------------------------------
---
PHP Associative Arrays

$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
-----------------------
          PHP - Two-dimensional Arrays
echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2].".<br>";
echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2].".<br>";
echo $cars[2][0].": In stock: ".$cars[2][1].", sold: ".$cars[2][2].".<br>";
echo $cars[3][0].": In stock: ".$cars[3][1].", sold: ".$cars[3][2].".<br>";

for ($row = 0; $row < 4; $row++) {
  echo "<p><b>Row number $row</b></p>";
  echo "<ul>";
  for ($col = 0; $col < 3; $col++) {
    echo "<li>".$cars[$row][$col]."</li>";
  }
  echo "</ul>";
}
--------------------------------------------
PHP - Sort Functions For Arrays
In this chapter, we will go through the following PHP array sort functions:

sort() - sort arrays in ascending order
rsort() - sort arrays in descending order
asort() - sort associative arrays in ascending order, according to the value
ksort() - sort associative arrays in ascending order, according to the key
arsort() - sort associative arrays in descending order, according to the value
krsort() - sort associative arrays in descending order, according to the key
```

PHP Sorting Arrays

PHP - Two-dimensional Arrays

A two-dimensional array is an array of arrays (a three-dimensional array is an array of arrays of arrays).

First, take a look at the following table:

Name	Stock	Sold
Volvo	22	18
BMW	15	13
Saab	5	2
Land Rover	17	15

We can store the data from the table above in a two-dimensional array, like this:

```
$cars = array (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
);
```

▼ PHP Global Variables - Superglobals

The PHP superglobal variables are:

- $GLOBALS
- $_SERVER
- $_REQUEST
- $_POST
- $_GET
- $_FILES
- $_ENV
- $_COOKIE
- $_SESSION

```
                    PHP $GLOBALS
$x = 75;
$y = 25;

function addition() {
  $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
}

addition();
echo $z;
```

```
-----------------------------------===========
              PHP $_SERVER
$_SERVER is a PHP super global variable which holds information about headers, paths,
 and script locations.


echo $_SERVER['PHP_SELF'];
echo "<br>";
echo $_SERVER['SERVER_NAME'];
echo "<br>";
echo $_SERVER['HTTP_HOST'];
echo "<br>";
echo $_SERVER['HTTP_REFERER'];
echo "<br>";
echo $_SERVER['HTTP_USER_AGENT'];
echo "<br>";
echo $_SERVER['SCRIPT_NAME'];
----------------------------------------------
```

$_SERVER

```
              PHP $_REQUEST
<html>
<body>

<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  // collect value of input field
  $name = $_REQUEST['fname'];
  if (empty($name)) {
    echo "Name is empty";
  } else {
    echo $name;
  }
}
?>

</body>
</html>
PHP $_REQUEST is a PHP super global variable which is used to collect data after submi
tting an HTML form.
-----------------------------------------------------
                  PHP $_POST
PHP $_POST is a PHP super global variable which is used to collect form data after sub
mitting an HTML form with method="post". $_POST is also widely used to pass variables.
```

```
<html>
<body>

<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  // collect value of input field
  $name = $_POST['fname'];
  if (empty($name)) {
    echo "Name is empty";
  } else {
    echo $name;
  }
}
?>

</body>
</html>
-------------------------------------------------------------------------------------
--------
                          PHP $_GET
Super global variables are built-in variables that are always available in all scope
PHP $_GET is a PHP super global variable which is used to collect form data after subm
itting an HTML form with method="get".


<html>
<body>

<a href="test_get.php?subject=PHP&web=W3schools.com">Test $GET</a>

</body>
</html>
----------------------
<html>
<body>

<?php
echo "Study " . $_GET['subject'] . " at " . $_GET['web'];
?>

</body>
</html>
```

▼ PHP Regular Expressions

Regular Expression Functions

PHP provides a variety of functions that allow you to use regular expressions. The `preg_match()`, `preg_match_all()` and `preg_replace()` functions are some of the most commonly used ones:

Function	Description
preg_match()	Returns 1 if the pattern was found in the string and 0 if not
preg_match_all()	Returns the number of times the pattern was found in the string, which may also be 0
preg_replace()	Returns a new string where matched patterns have been replaced with another string

```
Syntax
$exp = "/w3schools/i";
                Using preg_match()
<?php
$str = "Visit W3Schools";
$pattern = "/w3schools/i";
echo preg_match($pattern, $str); // Outputs 1
?>

                Using preg_match_all()
<?php
$str = "The rain in SPAIN falls mainly on the plains.";
$pattern = "/ain/i";
echo preg_match_all($pattern, $str); // Outputs 4
?>
                  Using preg_replace()
<?php
$str = "Visit Microsoft!";
$pattern = "/microsoft/i";
echo preg_replace($pattern, "W3Schools", $str); // Outputs "Visit W3Schools!"
?>
```

Regular Expression Modifiers

Modifiers can change how a search is performed.

Modifier	Description
i	Performs a case-insensitive search
m	Performs a multiline search (patterns that search for the beginning or end of a string will match the beginning or end of each line)
u	Enables correct matching of UTF-8 encoded patterns

Regular Expression Patterns

Brackets are used to find a range of characters:

Expression	Description
[abc]	Find one character from the options between the brackets
[^abc]	Find any character NOT between the brackets
[0-9]	Find one character from the range 0 to 9

Metacharacters

Metacharacters are characters with a special meaning:

Metacharacter	Description
|	Find a match for any one of the patterns separated by | as in: cat|dog|fish
.	Find just one instance of any character
^	Finds a match as the beginning of a string as in: ^Hello
$	Finds a match at the end of the string as in: World$
\d	Find a digit
\s	Find a whitespace character
\b	Find a match at the beginning of a word like this: \bWORD, or at the end of a word like this: WORD\b
\uxxxx	Find the Unicode character specified by the hexadecimal number xxxx

Quantifiers

Quantifiers define quantities:

Quantifier	Description
n+	Matches any string that contains at least one *n*
n*	Matches any string that contains zero or more occurrences of *n*
n?	Matches any string that contains zero or one occurrences of *n*
n{x}	Matches any string that contains a sequence of *X* *n*'s
n{x,y}	Matches any string that contains a sequence of X to Y *n*'s
n{x,}	Matches any string that contains a sequence of at least X *n*'s