PHP Create Arrays

[❮ Previous](https://www.w3schools.com/php/php_arrays_associative.asp)[Next ❯](https://www.w3schools.com/php/php_arrays_access.asp)

Create Array

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

Example[Get your own PHP Server](https://www.w3schools.com/php/php_server.asp)

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

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

Example

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

Multiple Lines

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

Example

$cars = [

"Volvo",

"BMW",

"Toyota"

];

Trailing Comma

A comma after the last item is allowed:

Example

$cars = [

"Volvo",

"BMW",

"Toyota",

];

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"

];

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

];

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";

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 »](https://www.w3schools.com/php/phptryit.asp?filename=tryphp_arrays_create3)

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

[❮ Previous](https://www.w3schools.com/php/php_arrays_create.asp)[Next ❯](https://www.w3schools.com/php/php_arrays_update.asp)

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.

Example

Access an item by referring to its index number:

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

echo $cars[2];

**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"];

Double or Single Quotes

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

Example

echo $cars["model"];

echo $cars['model'];

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]();

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"]();

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>";

}

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

[❮ Previous](https://www.w3schools.com/php/php_arrays_access.asp)[Next ❯](https://www.w3schools.com/php/php_arrays_remove.asp)

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.

Example

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

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

$cars[1] = "Ford";

**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 »](https://www.w3schools.com/php/phptryit.asp?filename=tryphp_arrays_update2)

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);

**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 Delete Array Items

[❮ Previous](https://www.w3schools.com/php/php_arrays_update.asp)[Next ❯](https://www.w3schools.com/php/php_arrays_sort.asp)

Remove Array Item

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

The unset() function deletes specified variables, and can therefor be used to delete array items:

Example

Remove the second item:

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

unset($cars[1]);

Remove Multiple Array Items

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]);

**Note:**The unset() function will NOT re-arrange the indexes, and the examples above will no longer contain the missing indexes.

Using the array\_splice Function

If you want the array to re-arrange the indexes, 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.

Example

Remove the second item:

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

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

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

Remove Item From an Associative Array

To remove items from an associative array, you can use unset() function like before, but referring to the key name instead of index.

Example

Remove the "model":

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

unset($cars["model"]);

PHP Sorting Arrays

[❮ Previous](https://www.w3schools.com/php/php_arrays_remove.asp)[Next ❯](https://www.w3schools.com/php/php_arrays_multidimensional.asp)

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

PHP - Sort Functions For Arrays

In this chapter, we will go through the following PHP array sort functions:

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

Sort Array in Ascending Order - sort()

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

Example

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

sort($cars);

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

Example

$numbers = array(4, 6, 2, 22, 11);

sort($numbers);

Sort Array in Descending Order - rsort()

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

Example

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

rsort($cars);

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

Example

$numbers = array(4, 6, 2, 22, 11);

rsort($numbers);

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

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

Example

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

asort($age);

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

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

Example

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

ksort($age);

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

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

Example

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

arsort($age);

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

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

Example

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

krsort($age);