



- [Downloads](#)
- [Documentation](#)
- [Get Involved](#)
- [Help](#)
- 

[Dutch PHP Conference 2024](#)

[Getting Started](#)

[Introduction](#)

[A simple tutorial](#)

[Language Reference](#)

[Basic syntax](#)

[Types](#)

[Variables](#)

[Constants](#)

[Expressions](#)

[Operators](#)

[Control Structures](#)

[Functions](#)

[Classes and Objects](#)

[Namespaces](#)

[Enumerations](#)

[Errors](#)

[Exceptions](#)

[Fibers](#)

[Generators](#)

[Attributes](#)

[References Explained](#)

[Predefined Variables](#)

[Predefined Exceptions](#)

[Predefined Interfaces and Classes](#)

[Predefined Attributes](#)

[Context options and parameters](#)

[Supported Protocols and Wrappers](#)

[Security](#)

[Introduction](#)

[General considerations](#)

[Installed as CGI binary](#)

[Installed as an Apache module](#)

[Session Security](#)

[Filesystem Security](#)

[Database Security](#)

[Error Reporting](#)

[User Submitted Data](#)

[Hiding PHP](#)

[Keeping Current](#)

[Features](#)

[HTTP authentication with PHP](#)

[Cookies](#)

[Sessions](#)

[Dealing with XForms](#)

[Handling file uploads](#)

[Using remote files](#)

[Connection handling](#)

[Persistent Database Connections](#)

[Command line usage](#)

[Garbage Collection](#)
[DTrace Dynamic Tracing](#)

[Function Reference](#)

[Affecting PHP's Behaviour](#)
[Audio Formats Manipulation](#)
[Authentication Services](#)
[Command Line Specific Extensions](#)
[Compression and Archive Extensions](#)
[Cryptography Extensions](#)
[Database Extensions](#)
[Date and Time Related Extensions](#)
[File System Related Extensions](#)
[Human Language and Character Encoding Support](#)
[Image Processing and Generation](#)
[Mail Related Extensions](#)
[Mathematical Extensions](#)
[Non-Text MIME Output](#)
[Process Control Extensions](#)
[Other Basic Extensions](#)
[Other Services](#)
[Search Engine Extensions](#)
[Server Specific Extensions](#)
[Session Extensions](#)
[Text Processing](#)
[Variable and Type Related Extensions](#)
[Web Services](#)
[Windows Only Extensions](#)
[XML Manipulation](#)
[GUI Extensions](#)

Keyboard Shortcuts

?	This help
j	Next menu item
k	Previous menu item
g p	Previous man page
g n	Next man page
G	Scroll to bottom
g g	Scroll to top
g h	Goto homepage
g s	Goto search (current page)
/	Focus search box

[Проверка типа »](#)

[« Строки](#)

- [Руководство по PHP](#)
- [Справочник языка](#)
- [Операторы](#)

Change language: Russian

Операторы, работающие с массивами

Операторы, работающие с массивами		
Пример	Название	Результат
<code>\$a + \$b</code>	Объединение	Объединение массива <i>\$a</i> с массивом <i>\$b</i> .
<code>\$a == \$b</code>	Равно	Возвращает true , если массив в переменной <i>\$a</i> и массив в переменной <i>\$b</i> содержат одни и те же пары ключ/значение.
<code>\$a === \$b</code>	Тождественно равно	Возвращает true , если массив в переменной <i>\$a</i> и массив в переменной <i>\$b</i> содержат одни и те же пары ключ/значение в том же самом порядке и того же типа.
<code>\$a != \$b</code>	Не равно	Возвращает true , если массив в переменной <i>\$a</i> не равен массиву в переменной <i>\$b</i> .
<code>\$a <> \$b</code>	Не равно	Возвращает true , если массив в переменной <i>\$a</i> не равен массиву в переменной <i>\$b</i> .
<code>\$a !== \$b</code>	Тождественно не равно	Возвращает true , если массив в переменной <i>\$a</i> не равен тождественно массиву в переменной <i>\$b</i> .

Оператор `+` возвращает левый массив, к которому был присоединён правый массив. Для ключей, которые существуют в обоих массивах, будут выбраны значения из левого массива, а элементы из правого массива, которые им соответствуют, будут проигнорированы.

<?php

```
$a = array("a" => "яблоко", "b" => "банан");
$b = array("a" => "груша", "b" => "клубника", "c" => "вишня");

$c = $a + $b; // Объединение $a и $b
echo "Объединение \a и \b: \n";
var_dump($c);

$c = $b + $a; // Объединение $b и $a
echo "Объединение \b и \a: \n";
var_dump($c);

$a += $b; // Объединение $a += $b — это $a и $b
echo "Объединение \a += \b: \n";
var_dump($a);
```

После выполнения скрипт напечатает следующее:

```
Объединение $a и $b:
array(3) {
  ["a"]=>
  string(5) "яблоко"
  ["b"]=>
  string(6) "банан"
  ["c"]=>
  string(6) "вишня"
}
Объединение $b и $a:
array(3) {
  ["a"]=>
  string(4) "груша"
  ["b"]=>
  string(10) "клубника"
  ["c"]=>
  string(6) "вишня"
}
Объединение $a += $b:
array(3) {
  ["a"]=>
  string(5) "яблоко"
  ["b"]=>
  string(6) "банан"
  ["c"]=>
  string(6) "вишня"
}
```

При сравнении элементы массива признаются идентичными, если совпадает и ключ, и его значение.

Пример #1 Comparing arrays

```
<?php
$a = array("apple", "banana");
$b = array(1 => "banana", "0" => "apple");

var_dump($a == $b); // bool(true)
var_dump($a === $b); // bool(false)
?>
```

Смотрите также

- [Массивы](#)
- [Функции для работы с массивами](#)

[+add a note](#)

User Contributed Notes 7 notes

[up](#)

[down](#)

226

[cb at netalyst dot com ¶](#)

15 years ago

The union operator did not behave as I thought it would on first glance. It implements a union (of sorts) based on the keys of the array, not on the values.

For instance:

```
<?php
$a = array('one','two');
$b=array('three','four','five');

//not a union of arrays' values
echo '$a + $b : ';
print_r ($a + $b);

//a union of arrays' values
echo "array_unique(array_merge($a,$b)):";
// cribbed from http://oreilly.com/catalog/progphp/chapter/ch05.html
print_r (array_unique(array_merge($a,$b)));
?>
```

//output

```
$a + $b : Array
(
    [0] => one
    [1] => two
    [2] => five
)
array_unique(array_merge(Array,Array)):Array
(
    [0] => one
    [1] => two
    [2] => three
    [3] => four
    [4] => five
)
```

[up](#)

[down](#)

2

[Anonymous ¶](#)

1 year ago

Merge two arrays and retain only unique values.

Append values from second array.

Do not care about keys.

```
<?php
$array1 = [
0 => 'apple',
1 => 'orange',
2 => 'pear',
];

$array2 = [
0 => 'melon',
1 => 'orange',
2 => 'banana',
];

$result = array_keys(
array_flip($array1) + array_flip($array2)
);
?>
```

Result:

```
[
[0] => "apple",
[1] => "orange",
[2] => "pear",
[3] => "melon",
[4] => "banana",
]
```

[up](#)

[down](#)

40

[Q1712 at online dot ms ¶](#)

16 years ago

The example may get u into thinking that the identical operator returns true because the key of apple is a string but that is not the case, cause if a string array key is the standart representation of a integer it's gets a numeral key automaticly.

The identical operator just requires that the keys are in the same order in both arrays:

```
<?php
$a = array (0 => "apple", 1 => "banana");
$b = array (1 => "banana", 0 => "apple");

var_dump($a === $b); // prints bool(false) as well

$b = array ("0" => "apple", "1" => "banana");

var_dump($a === $b); // prints bool(true)
?>
```

[up](#)

[down](#)

22

[dfranklin at fen dot com ¶](#)

19 years ago

Note that + will not renumber numeric array keys. If you have two numeric arrays, and their indices overlap, + will use the first array's values for each numeric key, adding the 2nd array's values only where the first doesn't already have a value for that index. Example:

```
$a = array('red', 'orange');
$b = array('yellow', 'green', 'blue');
```

```
$both = $a + $b;  
var_dump($both);
```

Produces the output:

```
array(3) { [0]=> string(3) "red" [1]=> string(6) "orange" [2]=> string(4) "blue" }
```

To get a 5-element array, use array_merge.

Dan

[up](#)

[down](#)

16

[Dan Patrick ¶](#)

11 years ago

It should be mentioned that the array union operator functions almost identically to array_replace with the exception that precedence of arguments is reversed.

[up](#)

[down](#)

5

[xtpeqii at Hotmail dot com ¶](#)

6 years ago

```
$a=[ 3, 2, 1];  
$b=[ 6, 5, 4];  
var_dump( $a + $b );
```

output:

```
array(3) {  
[0]=>  
int(3)  
[1]=>  
int(2)  
[2]=>  
int(1)  
}
```

The reason for the above output is that EVERY array in PHP is an associative one.

Since the 3 elements in \$b have the same keys(or numeric indices) as those in \$a, those elements in \$b are ignored by the union operator.

[up](#)

[down](#)

15

[amirlaher AT yahoo DOT co SPOT uk ¶](#)

21 years ago

[]= could be considered an Array Operator (in the same way that .= is a String Operator).

[]= pushes an element onto the end of an array, similar to array_push:

```
<?
```

```
$array= array(0=>"Amir",1=>"needs");
```

```
$array[]= "job";
```

```
print_r($array);
```

```
?>
```

Prints: Array ([0] => Amir [1] => needs [2] => job)

[+add a note](#)

- [Операторы](#)
 - [Приоритет](#)
 - [Арифметика](#)
 - [Инкремент и декремент](#)
 - [Присваивание](#)
 - [Побитовые операторы](#)
 - [Сравнение](#)
 - [Управление ошибками](#)

- [Исполнение](#)
- [Логика](#)
- [Строки](#)
- [Массивы](#)
- [Проверка типа](#)
- [Copyright © 2001-2024 The PHP Group](#)
- [My PHP.net](#)
- [Contact](#)
- [Other PHP.net sites](#)
- [Privacy policy](#)

