Search

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

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
<?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
<?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

User Contributed Notes 7 notes

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
)
```


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