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

natcasesort »
« ksort

- Руководство по PHP
- Справочник функций
- Модули, относящиеся к переменным и типам
- Массивы
- Функции для работы с массивами

list

(PHP 4, PHP 5, PHP 7, PHP 8)

list — Присваивает переменным значения похожим на массивы синтаксисом

Описание

list(mixed `$var`, mixed `...$vars` = ?): array

Как и конструкция array(), это не функция, а языковая конструкция. Конструкцией **list()** пользуются, чтобы присваивать списку переменных значения за одну операцию. Строки нельзя распаковать, а выражения **list()** нельзя вызывать без аргументов.

> **Замечание**:
>
> До PHP 7.1.0 конструкция **list()** работала только с индексированными массивами и принимала числовые индексы начиная с 0.

Список параметров

`var`

> Переменная.

`vars`

> Дополнительные переменные.

Возвращаемые значения

Возвращает присвоенный массив.

Список изменений

Версия	Описание
7.3.0	Добавлена поддержка присвоения по ссылкам при деструктурировании массива.
7.1.0	Теперь в конструкции **list()** разрешено задавать ключи. Это разрешает разыменовывать ассоциативные массивы и массивы с индексами не по порядку.

Примеры

Пример #1 Примеры использования конструкции list()

```php
<?php

$info = array('кофе', 'коричневый', 'кофеин');

// Составить список всех переменных
list($drink, $color, $power) = $info;
echo "$drink - $color, а $power делает его особенным.\n";

// Составить список только некоторых из них
list($drink, , $power) = $info;
echo "В $drink есть $power.\n";

// Или пропустить все, кроме третьей
list( , , $power) = $info;
echo "Мне нужен $power!\n";
```

```php
// Конструкция list() не работает со строками
list($bar) = "abcde";
var_dump($bar); // NULL
?>
```

Пример #2 Пример использования конструкции list()

```php
<?php

$result = $pdo->query("SELECT id, name FROM employees");
while (list($id, $name) = $result->fetch(PDO::FETCH_NUM)) {
echo "id: $id, name: $name\n";
}
?>
```

Пример #3 Использование list() с индексами массивов

```php
<?php

list($a, list($b, $c)) = array(1, array(2, 3));

var_dump($a, $b, $c);

?>
```

```
int(1)
int(2)
int(3)
```

Пример #4 Конструкция list() и порядок указания индексов

Порядок определения индексов в массиве, с которым будет работать конструкция **list()**, неважен.

```php
<?php

$foo = array(2 => 'a', 'foo' => 'b', 0 => 'c');
$foo[1] = 'd';
list($x, $y, $z) = $foo;
var_dump($foo, $x, $y, $z);
```

Даёт такой вывод (обратите внимание на порядок, в котором элементы сравнивались, и в каком порядке элементы записаны в синтаксисе **list()**):

```
array(4) {
  [2]=>
  string(1) "a"
  ["foo"]=>
  string(1) "b"
  [0]=>
  string(1) "c"
  [1]=>
  string(1) "d"
}
string(1) "c"
string(1) "d"
string(1) "a"
```

Пример #5 list() с ключами

Начиная с PHP 7.1.0 конструкции **list()** разрешено содержать явные ключи, которые указывают как произвольные выражения. Допустимо смешивать строковые и целочисленные ключи; однако элементы с ключами и без ключей смешивать нельзя.

```php
<?php

$data = [
["id" => 1, "name" => 'Tom'],
["id" => 2, "name" => 'Fred'],
];
```

```php
foreach ($data as ["id" => $id, "name" => $name]) {
echo "id: $id, name: $name\n";
}
echo PHP_EOL;
list(1 => $second, 3 => $fourth) = [1, 2, 3, 4];
echo "$second, $fourth\n";
```

Результат выполнения приведённого примера:

```
id: 1, name: Tom
id: 2, name: Fred

2, 4
```

Смотрите также

- each() - Возвращает текущую пару ключа и значения массива и сдвигает указатель на одну позицию вперёд
- array() - Создаёт массив
- extract() - Импортирует переменные массива в текущую таблицу символов

User Contributed Notes 25 notes

142
Rhamnia Mohamed ¶
6 years ago
Since PHP 7.1, keys can be specified

exemple :

```php
<?php
$array = ['locality' => 'Tunis', 'postal_code' => '1110'];

list('postal_code' => $zipCode, 'locality' => $locality) = $array;

print $zipCode; // will output 1110
print $locality; // will output Tunis
?>
```


128
carlosv775 at gmail dot com ¶
7 years ago
In PHP 7.1 we can do the following:

```php
<?php
[$a, $b, $c] = ['a', 'b', 'c'];
?>
```

Before, we had to do:

```php
<?php
list($a, $b, $c) = ['a', 'b', 'c'];
?>
```


91
grzeniufication ¶
9 years ago
The example showing that:

```php
$info = array('kawa', 'brązowa', 'kofeina');
```

```
list($a[0], $a[1], $a[2]) = $info;
var_dump($a);

outputs:
array(3) {
[2]=>
string(8) "kofeina"
[1]=>
string(5) "brązowa"
[0]=>
string(6) "kawa"
}

One thing to note here is that if you define the array earlier, e.g.:
$a = [0, 0, 0];

the indexes will be kept in the correct order:

array(3) {
[0]=>
string(4) "kawa"
[1]=>
string(8) "brązowa"
[2]=>
string(7) "kofeina"
}

Thought that it was worth mentioning.
```

up
down
71
megan at voices dot com ¶
10 years ago

```
As noted, list() will give an error if the input array is too short. This can be avoided by array_merge()'ing in some
default values. For example:

<?php
$parameter = 'name';
list( $a, $b ) = array_merge( explode( '=', $parameter ), array( true ) );
?>

However, you will have to array_merge with an array long enough to ensure there are enough elements (if $parameter is
empty, the code above would still error).

An alternate approach would be to use array_pad on the array to ensure its length (if all the defaults you need to add are
the same).

<?php
$parameter = 'bob-12345';
list( $name, $id, $fav_color, $age ) = array_pad( explode( '-', $parameter ), 4, '' );
var_dump($name, $id, $fav_color, $age);
/* outputs
string(3) "bob"
string(5) "12345"
string(0) ""
string(0) ""
*/
?>
```

up
down
76
chris at chlab dot ch ¶

11 years ago

The example states the following:

```
<?php
// list() doesn't work with strings
list($bar) = "abcde";
var_dump($bar);
// output: NULL
?>
```

If the string is in a variable however, it seems using list() will treat the string as an array:

```
<?php
$string = "abcde";
list($foo) = $string;
var_dump($foo);
// output: string(1) "a"
?>
```

up
down
51

grzeniufication ¶

6 years ago

```
<?php
/**
* It seems you can skip listed values.
* Here's an example to show what I mean.
*
* FYI works just as well with PHP 7.1 shorthand list syntax.
* Tested against PHP 5.6.30, 7.1.5
*/
$a = [ 1, 2, 3, 4 ];

// this is quite normal use case for list
echo "Unpack all values\n";
list($v1, $v2, $v3, $v4) = $a;
echo "$v1, $v2, $v3, $v4\n";
unset($v1, $v2, $v3, $v4);

// this is what I mean:
echo "Skip middle\n";
list($v1, , , $v4) = $a;
echo "$v1, $v2, $v3, $v4\n";
unset($v1, $v2, $v3, $v4);

echo "Skip beginning\n";
list( , , $v3, $v4) = $a;
echo "$v1, $v2, $v3, $v4\n";
unset($v1, $v2, $v3, $v4);

echo "Skip end\n";
list($v1, $v2, , ) = $a;
echo "$v1, $v2, $v3, $v4\n";
unset($v1, $v2, $v3, $v4);

echo "Leave middle\n";
list( , $v2, $v3, ) = $a;
echo "$v1, $v2, $v3, $v4\n";
unset($v1, $v2, $v3, $v4);
```

up
down
36

pemapmodder1970 at gmail dot com ¶

8 years ago

```
list() can be used with foreach

<?php
$array = [[1, 2], [3, 4], [5, 6]];

foreach($array as list($odd, $even)){
echo "$odd is odd; $even is even", PHP_EOL;
}
?>

The output:
===
1 is odd; 2 is even
3 is odd; 4 is even
5 is odd; 6 is even
```

up
down
26

john at jbwalker dot com ¶

10 years ago

```
The list construct seems to look for a sequential list of indexes rather taking elements in sequence. What that obscure statement means is that if you unset an element, list will not simply jump to the next element and assign that to the variable but will treat the missing element as a null or empty variable:

$test = array("a","b","c","d");
unset($test[1]);
list($a,$b,$c)=$test;
print "\$a='$a' \$b='$b' \$c='$c'<BR>";

results in:
$a='a' $b='' $c='c'

not:
$a='a' $b='c' $c='d'
```

up
down
47

svennd ¶

11 years ago

```
The list() definition won't throw an error if your array is longer then defined list.
<?php

list($a, $b, $c) = array("a", "b", "c", "d");

var_dump($a); // a
var_dump($b); // b
var_dump($c); // c
?>
```

up
down
6

diyor024 at gmail dot com ¶

2 years ago

```
Don't miss simple array pattern matching since php 7

<?php

[$a] = ['hello!'];
var_dump($a); // 'hello!'

$arr = [4 => 50];
[4 => $fifty] = $arr;
```

```
var_dump($fifty); // 50

$multidimensionalArray = [['id' => 15, 'email' => 'diyor024@gmail.com']];
[['id' => $id, 'email' => $email]] = $multidimensionalArray;
var_dump($id, $email); // 15 diyor024@gmail.com

?>
```

up
down
3

nek dot dev at gmail dot com ¶

1 year ago

It can be convenient to specify a default value in case an element is missing in the list. You can use operator + for this:

```
<?php
$someArray = ['color' => 'orange'];
['color' => $color, 'size' => $size] = $someArray + ['color' => null, 'size' => null];
?>
```

This will avoid the warning `Undefined array key "size"` you would encounter otherwise.

up
down
5

vike2000 at gmail dot com ¶

4 years ago

Setting it like <?php list($var1,$varN) = null ?> does _not_ raise an E_NOTICE (or other error) and afaics effectively equals an https://php.net/function.unset of $var1,$varN.

I note this as contrasting with the fact that PHP triggers an E_NOTICE about "Undefined offset" "if there aren't enough array elements to fill the list()", as attow documented for https://php.net/control-structures.foreach#control-structures.foreach.list and here only noted in https://php.net/function.list#122951 by Mardaneus.

For completeness, a bash(1) (v5.0 or 4.3 on macos10.13) cli test producing the same result for all my PHP-versions (installed via macports.org) follows. It's also tested with php7.3 using bash5.0 on Debian10:

```
bash --noprofile --norc -c 'for php in php{{53..56},{70..73}};do for literal in "array()" null;do echo -n $php …
=$literal:&&$php -n -d error_reporting=E_ALL -r "var_dump(list(\$var)=$literal);";done;done'

# Above produces the same result pairs per version from:
php53 …=array():
Notice: Undefined offset: 0 in Command line code on line 1
array(0) {
}
# ... to:
php73 …=null:NULL
```

up
down
5

mark at manngo dot net ¶

3 years ago

For PHP 7.1 on, the documentation states that integer and string keys can be mixed, but that elements with and without keys cannot. Here is an example, using data from getimagesize() with mixed keys:

```
<?php
$data=[
0=> 160,
1 => 120,
2 => 2,
3 => 'width="160" height="120"',
'mime' => 'image/jpeg'
];
list(0=>$width,1=>$height,2=>$type,3=>$dimensions,'mime'=>$mime)=$data;
```

```
?>
```

Here, the numeric keys also need to be specified, as if the whole array is treated as an associative array.

As noted elsewhere, the list() operator can be written in array format:

```
<?php
[0=>$width,1=>$height,2=>$type,3=>$dimensions,'mime'=>$mime]=$data;
?>
```

up
down
7
petru at fuxspam dot xtremeweb dot ro ¶
5 years ago

This is something I haven't seen in documentation.

Since PHP 7.1, you can use short-hand list unpacking using square brackets, just like short-hand array declaration:

```
<?php

$foo = ['a', 'b', 'c'];

// short-hand array definition
[$a, $b, $c] = $foo;
echo $a; // displays "a"

// it's same like:
list($x, $y, $z) = $foo;
echo $x; // displays "a"

?>
```

up
down
3
samraskul at gmail dot com ¶
2 years ago

```
list($a, $b, $c) = ["blue", "money", 32];
```

shortcut:

```
[$a, $b, $c] = ["blue", "money", 32];
```

up
down
5
Paul Marti ¶
4 years ago

Since 7.1.0, you can use an array directly without list():

```
<?php
[$test, $test2] = explode(",", "hello, world");
echo $test . $test2; // hello, world
?>
```

up
down
5
blazej ¶
6 years ago

From PHP Version 7.1 you can specify keys in list(), or its new shorthand [] syntax. This enables destructuring of arrays with non-integer or non-sequential keys.

```
<?php
$data = [
```

```
["id" => 1, "name" => 'Tom'],
["id" => 2, "name" => 'Fred'],
];

// list() style
list("id" => $id1, "name" => $name1) = $data[0];

// [] style
["id" => $id1, "name" => $name1] = $data[0];

// list() style
foreach ($data as list("id" => $id, "name" => $name)) {
// logic here with $id and $name
}

// [] style
foreach ($data as ["id" => $id, "name" => $name]) {
// logic here with $id and $name
}
```

up
down
4
anthony dot ossent at live dot fr ¶
7 years ago

a simple example of use to swap two variables :

```
$a = 'hello';
$b = 'world';

list($a, $b) = [$b, $a];

echo $a . ' ' . $b; //display "world hello"
```

another example :

```
function getPosition($x, $y, $z)
{
// ... some operations like $x++...
return [$x, $y, $z];
}

list($x, $y, $z) = getPosition($x ,$y, $z);
```

up
down
1
contato at tobias dot ws ¶
5 years ago

Since PHP 7.1 the [] may now be used as an alternative to the existing list() syntax:

```
<?php
[$number, $message] = explode('|', '123|Hello World!');
?>
```

up
down
0
Colin Guthrie ¶
8 years ago

If you want use the undefined behaviour as you might expect it e.g. if you want:

```
$b = ['a','b']; list($a, $b) = $b;
```

to result in $a=='a' and $b=='b', then you can just cast $b to an array (even although it already is) to create a copy.

e.g.

```
$b = ['a','b']; list($a, $b) = (array)$b;
```

and get the expected results.

up
down
-2

Dean ¶

8 years ago

UNDOCUMENTED BEHAVIOR:

```
list($a,$b,$c) = null;
```

in fact works like:

```
$a = null; $b = null; $c = null;
```

...So correspondingly:

```
list($rows[]) = null;
```

Will increment count($rows), just as if you had executed $rows[] = null;

Watch out for this (for example) when retrieving entire tables from a database, e.g.

```
while (list($rows[]) = $mysqlresult->fetch_row());
```

This will leave an extra 'null' entry as the last element of $rows.

up
down
-2

Mardaneus ¶

5 years ago

Unless you specify keys when using list() it expects the array being fed into it to start at 0.

So having the following code will result in a notice level warning "Undefined offset: 0" and variables not filling as expected

```
<?php
list($c1, $c2, $c3) = array [1 =>'a', 2 => 'b', 3 => 'c'];

var_dump($c1); // NULL
var_dump($c2); // string(1) "a"
var_dump($c3); // string(1) "b"

?>
```

up
down
-3

fredsaavedra at hotmail dot com ¶

3 years ago

Easy way to get actual date and time values in variables.

```
list($day,$month,$year,$hour,$minute,$second) = explode('-',date('d-m-Y-G-i-s'));
echo "$day-$month-$year $hour".":".$minute.":".$second;
```

up
down
-6

JD ¶

4 years ago

As of PHP 7.3, lists now support array destructuring - see here: https://www.php.net/manual/en/migration73.new-

features.php
up
down
-8
sergey dot leonidovich dot shevchenko at gmail dot com ¶
3 years ago

```
['a' => $x, 'b' => $y] = ['a' => 1, 'b' => 2];

echo $x . '-' . $y; // 1-2
```

＋add a note

- array_values
- array_walk_recursive
- array_walk
- array
- arsort
- asort
- compact
- count
- current
- end
- extract
- in_array
- key_exists
- key
- krsort
- ksort
- list
- natcasesort
- natsort
- next
- pos
- prev
- range
- reset
- rsort
- shuffle
- sizeof
- sort
- uasort
- uksort
- usort

- Deprecated
  - each