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

break »
« for

- Руководство по PHP
- Справочник языка
- Управляющие конструкции

Change language: Russian

foreach

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

Языковая конструкция `foreach` предлагает простой способ перебора массивов. Конструкция `foreach` работает только с массивами и объектами, и будет выдавать ошибку при попытке использовать её с переменными других типов данных или неинициализированными переменными. Разработчику доступны два вида синтаксиса:

```
foreach (iterable_expression as $value)
    statement
foreach (iterable_expression as $key => $value)
    statement
```

Первая форма обходит доступные для перебора данные, заданные выражением `iterable_expression`. На каждой итерации значение текущего элемента присваивается переменной `$value`.

Вторая форма дополнительно будет присваивать ключ текущего элемента переменной `$key` на каждой итерации.

Обратите внимание, что конструкция `foreach` не изменяет внутренний указатель массива, с которым, например, работают функции current() и key().

Разработчику доступна настройка итерации объектов.

Чтобы непосредственно изменять элементы массива внутри цикла, перед переменной `$value` указывают знак &. Тогда значение будет присвоено по ссылке.

```php
<?php

$arr = array(1, 2, 3, 4);
foreach ($arr as &$value) {
$value = $value * 2;
}
// массив $arr теперь выглядит так: array(2, 4, 6, 8)
unset($value); // разорвать ссылку на последний элемент
?>
```

Внимание

Ссылка переменной `$value` на последний элемент массива останется даже после окончания цикла `foreach`. Рекомендовано уничтожать её языковой конструкцией unset(). В противном случае разработчик столкнётся с таким поведением:

```php
<?php

$arr = array(1, 2, 3, 4);
foreach ($arr as &$value) {
$value = $value * 2;
}
// $arr = array(2, 4, 6, 8)

// Без вызова конструкции unset($value), перменная $value всё ещё ссылается на последний элемент: $arr[3]

foreach ($arr as $key => $value) {
// Значение элемента $arr[3] будет обновляться значениями массива $arr при каждой итерации цикла...
echo "{$key} => {$value} ";
print_r($arr);
}
// ...до тех пор, пока предпоследнее значение не будет скопировано в последнее значение

// вывод:
// 0 => 2 Array ( [0] => 2, [1] => 4, [2] => 6, [3] => 2 )
// 1 => 4 Array ( [0] => 2, [1] => 4, [2] => 6, [3] => 4 )
// 2 => 6 Array ( [0] => 2, [1] => 4, [2] => 6, [3] => 6 )
// 3 => 6 Array ( [0] => 2, [1] => 4, [2] => 6, [3] => 6 )
```

```
?>
```

Разрешено перебирать значение константного массива по ссылке:

```
<?php

foreach (array(1, 2, 3, 4) as &$value) {
$value = $value * 2;
}
?>
```

Замечание:

Языковая конструкция `foreach` не поддерживает подавление сообщений об ошибках через оператор `@`.

Ещё примеры, которые показывают работу конструкции:

```
<?php

/* Пример 1: только значение */

$a = array(1, 2, 3, 17);

foreach ($a as $v) {
echo "Текущее значение переменной \$a: $v.\n";
}

/* Пример 2: значение (для иллюстрации массив выводится в виде значения с ключом) */

$a = array(1, 2, 3, 17);

$i = 0; /* только для пояснения */

foreach ($a as $v) {
echo "\$a[$i] => $v.\n";
$i++;
}

/* Пример 3: ключ и значение */

$a = array(
"one" => 1,
"two" => 2,
"three" => 3,
"seventeen" => 17
);

foreach ($a as $k => $v) {
echo "\$a[$k] => $v.\n";
}

/* Пример 4: многомерные массивы */
$a = array();
$a[0][0] = "a";
$a[0][1] = "b";
$a[1][0] = "y";
$a[1][1] = "z";

foreach ($a as $v1) {
foreach ($v1 as $v2) {
echo "$v2\n";
}
}
```

```
/* Пример 5: динамические массивы */

foreach (array(1, 2, 3, 4, 5) as $v) {
echo "$v\n";
}
?>
```

Распаковка вложенных массивов языковой конструкцией list()

(PHP 5 >= 5.5.0, PHP 7, PHP 8)

Доступен перебор массива массивов и распаковка вложенного массива в переменные цикла путём передачи конструкции list() в качестве значения.

Например:

```
<?php

$array = [
[1, 2],
[3, 4],
];

foreach ($array as list($a, $b)) {
// Переменная $a содержит первый элемент вложенного массива,
// а переменная $b — второй.
echo "A: $a; B: $b\n";
}
?>
```

Результат выполнения приведённого примера:

```
A: 1; B: 2
A: 3; B: 4
```

В конструкцию list() разрешено передавать меньшее количество элементов, чем содержится во вложенном массиве, тогда оставшиеся значения массива будут проигнорированы:

```
<?php

$array = [
[1, 2],
[3, 4],
];

foreach ($array as list($a)) {
// Обратите внимание, переменной $b здесь нет.
echo "$a\n";
}
?>
```

Результат выполнения приведённого примера:

```
1
3
```

Если массив содержит недостаточно элементов для заполнения переменных в конструкции list(), то будет сгенерировано уведомление об ошибке:

```
<?php

$array = [
[1, 2],
[3, 4],
];
```

```
foreach ($array as list($a, $b, $c)) {
echo "A: $a; B: $b; C: $c\n";
}
?>
```

Результат выполнения приведённого примера:

```
Notice: Undefined offset: 2 in example.php on line 7
A: 1; B: 2; C:

Notice: Undefined offset: 2 in example.php on line 7
A: 3; B: 4; C:
```


User Contributed Notes 2 notes

24
***Okafor Chiagozie* ¶**
1 year ago

```
An easier way to unpack nested array elements

$array = [
[1, 2],
[3, 4],
];

foreach ($array as [$a, $b]) {
echo "A: $a; B: $b\n";
}
```


1
***Sanusi Hassan* ¶**
1 year ago

```
destructure array elements

you can unpac nested array elements using the following

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

foreach ($array as $v) {
[$a, $b] = $v;
echo "A: $a; B: $b\n";
}
?>
```

- for
- foreach
- break
- continue
- switch
- match
- declare
- return
- require
- include
- require_once
- include_once
- goto