



**нетология**  
университет интернет-профессий

# СТРОКИ, МАССИВЫ И ОБЪЕКТЫ



**СЕРГЕЙ ГЕРАСИМЕНКО** / ОАО РЖД



# СЕРГЕЙ ГЕРАСИМЕНКО

Программист-разработчик в ОАО РЖД



[gerasimenkosv@bk.ru](mailto:gerasimenkosv@bk.ru)



# ПЛАН ЗАНЯТИЯ

1. [Строки](#)
2. [Списки](#)
3. [Массивы](#)
4. [NULL](#)
5. [Ресурсы](#)

LOREM  
IPSUM

DOLOR SIT AMET

CONSECTETUR • ADIPISCING • ELIT



**СТРОКИ**

# ОБЪЯВЛЕНИЕ СТРОК

```
1 $a = 'Строка в одинарных кавычках';  
2 $b = "В двойных кавычках появляется возможность использовать  
3 переменные вот так: $a";  
4 $c = "Или так: {$b}";  
5 $d = <<<PHP  
6 Длинную строку удобно объявить вот таким способом,  
7 особенно, если в ней используются оба типа кавычек PHP;
```

# ОПЕРАЦИИ НАД СТРОКАМИ

## — Склеивание (конкатенация)

```
$a = "38" . 'попугаев'; // 38 попугаев
```

## — Сравнение

```
$result = "a" > "Я"; // true
```

```
$result = "a" > "я"; // false
```

---

При нестрогом (==) сравнении интерпретатор пытается привести строки к числам!

При произведении арифметических операций первые символы строки будут приведены, по возможности, к числу, остальные — отброшены.

```
$sum = "38попугаев" + "1 крылышко 3333"; // 39
```





# ПРЕОБРАЗОВАНИЕ СТРОК

## SUBSTR/MB\_SUBSTR

```
string substr (string $string, int $start [, int $length ])
```

```
string mb_substr (string $str, int $start [, int $length = NULL [, string  
$encoding = mb_internal_encoding() ]])
```

Если `start` неотрицателен, возвращаемая подстрока начинается с позиции `start` от начала строки, считая от нуля. Например, в строке `'abcdef'`, в позиции 0 находится символ `'a'`, в позиции 2 — символ `'c'`, и т.д. Если `start` отрицательный, возвращаемая подстрока начинается с позиции, отстоящей на `start` символов от конца строки `string`.

## STR\_PAD

```
string str_pad (string $input, int $pad_length [, string $pad_string = " " [, int  
$pad_type = STR_PAD_RIGHT ]])
```

Эта функция возвращает строку input, дополненную слева, справа или с обеих сторон до заданной длины. Если необязательный аргумент pad\_string не передан, то input будет дополнен пробелами, иначе он будет дополнен символами из pad\_string до нужной длины.

# STR\_REPEAT

```
string str_repeat (string $input, int $multiplier)
```

Возвращает строку input, повторенную multiplier раз. multiplier должен быть больше или равен нулю. Если он равен нулю, возвращается пустая строка.



# STR\_SHUFFLE

`string str_shuffle (string $str)`

`str_shuffle()` перемешивает символы в строке. Выбирается одна возможная перестановка из всех возможных.



# STRREV

`string strrev (string $string)`

Возвращает строку `string`, перевернутую задом наперед.



## **РАБОТА С ПРОБЕЛЬНЫМИ СИМВОЛАМИ**

# TRIM

```
string trim (string $str [, string $character_mask = "\t\n\r\0\x0B" ])
```

Эта функция возвращает строку `str` с удаленными из начала и конца строки пробелами. Если второй параметр не передан, `trim()` удаляет следующие символы:

- " " (ASCII 32 (0x20)), обычный пробел
- "\t" (ASCII 9 (0x09)), символ табуляции
- "\n" (ASCII 10 (0x0A)), символ перевода строки
- "\r" (ASCII 13 (0x0D)), символ возврата каретки
- "\0" (ASCII 0 (0x00)), NUL-байт
- "\x0B" (ASCII 11 (0x0B)), вертикальная табуляция





## RTRIM/LTRIM

```
string rtrim (string $str [, string $character_mask ])
```

Эта функция возвращает строку `str` с удаленными из конца строки пробелами.

```
string ltrim (string $str [, string $character_mask ])
```

Удаляет пробелы (или другие символы) из начала строки.

# NL2BR

```
string nl2br (string $string [, bool $is_xhtml = true ])
```

Возвращает строку string, в которой перед каждым переводом строки (\r\n, \n\r, \n и \r) вставлен `<br />` или `<br>`.



## **РАБОТА С РЕГИСТРОМ СИМВОЛОВ**

# STRTOLOWER/MB\_STRTOLOWER

```
string strtolower (string $string)
```

Возвращает строку `string`, в которой все буквенные символы переведены в нижний регистр.

```
string mb_strtolower (string $str [, string $encoding = mb_internal_encoding()])
```

В отличие от `strtolower()`, то что символ является буквой определяется на основании свойств символа Юникода. Таким образом на поведение функции не влияют региональные настройки системы, а также функция может преобразовывать символы, имеющие буквенные особенности, такие как А-умляут (Ä).

# STRTOUPPER/MB\_STRTOUPPER

```
string strtoupper (string $string)
```

Возвращает строку string, в которой все буквенные символы переведены в верхний регистр.

```
string mb_strtoupper (string $str [, string $encoding =  
mb_internal_encoding() ])
```

В отличие от `strtoupper()`, то что символ является буквой определяется на основании свойств символа Юникода. Таким образом на поведение функции не влияют региональные настройки системы, а также функция может преобразовывать символы, имеющие буквенные особенности, такие как А-умляут (Ä).



# LCFIRST/UCFIRST

`string lcfirst (string $str)`

Возвращает строку `str`, первый символ которой был преобразован в нижний регистр, если этот символ является буквой.

`string ucfirst (string $str)`

Возвращает строку `str`, в которой первый символ переведен в верхний регистр, если этот символ является буквой.



# UCWORDS

`string ucwords (string $str)`

Возвращает строку `str`, в которой первый символ каждого слова переведен в верхний регистр, если этот символ является буквой. Эта функция считает словами последовательности символов, разделенных пробельными символами, которыми являются пробел, разрыв строки, перевод строки, возврат каретки, горизонтальная и вертикальная табуляция.

## MB\_CONVERT\_CASE

```
string mb_convert_case (string $str, int $mode [, string $encoding =  
mb_internal_encoding() ])
```

Производит смену регистра символов в строке `string` в соответствии с режимом `mode`. Это может быть одна из констант `MB_CASE_UPPER`, `MB_CASE_LOWER` или `MB_CASE_TITLE`. В отличие от стандартных функций смены регистра, вроде `strtolower()` и `strtoupper()`, смена регистра осуществляется на основе свойств символа Юникода.



# РАБОТА С КОДИРОВКОЙ СТРОК – ICONV

```
string iconv (string $in_charset, string $out_charset, string $str)
```

Преобразует набор символов строки `str` из кодировки `in_charset` в `out_charset`.



## **ПОИСК И ЗАМЕНА СИМВОЛОВ В СТРОКЕ**

# STRPOS

```
mixed strpos (string $haystack, mixed $needle [, int $offset = 0 ])  
int mb_strpos ( string $haystack, string $needle [, int $offset = 0 [, string  
$encoding = mb_internal_encoding() ]])
```

Ищет позицию первого вхождения подстроки `needle` в строку `haystack`. Если `needle` не является строкой, он приводится к целому и трактуется как код символа. Если `offset` указан, то поиск будет начат с указанного количества символов с начала строки. В отличие от `strrpos()` и `stripos()` данный параметр не может быть отрицательным.

# STRRPOS

```
int strrpos (string $haystack, string $needle [, int $offset = 0 ])
```

```
int mb_strrpos (string $haystack, string $needle [, int $offset = 0 [, string  
$encoding = mb_internal_encoding() ]])
```

Ищет позицию последнего вхождения подстроки `needle` в строку `haystack`. Если указан `offset`, то поиск начнется с данного количества символов с начала строки. Если передано отрицательное значение, поиск начнется с указанного количества символов от конца строки, но по-прежнему будет производиться поиск последнего вхождения.

# STR\_REPLACE

mixed str\_replace (mixed \$search, mixed \$replace, mixed \$subject [, int &\$count ])

Эта функция возвращает строку или массив, в котором все вхождения search в subject заменены на replace. Если search и replace — массивы, то str\_replace() использует каждое значение из соответствующего массива для поиска и замены в subject. Если в массиве replace меньше элементов, чем в search, в качестве строки замены для оставшихся значений будет использована пустая строка. Если search — массив, а replace — строка, то эта строка замены будет использована для каждого элемента массива search.

# STRLEN/MB\_STRLEN

```
int strlen (string $string)
```

```
mixed mb_strlen (string $str [, string $encoding = mb_internal_encoding() ])
```

Возвращает длину строки `string`. `mb_strlen()` Возвращает `FALSE`, если передан недопустимый параметр `encoding`.



# ПОЛНЫЙ СПИСОК ФУНКЦИЙ ДЛЯ РАБОТЫ СО СТРОКАМИ

<http://php.net/manual/ru/ref.strings.php>

## 100 Things

1. move to Washington D.C.
2. go to graduate school
3. become a professional
4. get married
5. have children
6. take my children to Disney world
7. visit India
8. visit Norway
9. get on the Seattle ferry
10. become a senator
11. work in the white house
12. write a cook book
13. meet Derek Zener
14. vacation in Hawaii
15. learn Italian
16. visit Egypt
17. go skydiving
18. learn to skateboard
19. catch a squirrel
20. have a pet duck
21. live in New York
- 22.
- 23.
- 24.
- 25.





# СПИСКИ

---

**Список — это ловкий способ сделать несколько присваиваний за один раз**

```
$row = mysqli_fetch_row($link, $result);  
list($id, $title) = $row; // $id = $row[0], $title = $row[1]
```



variable



array



# МАССИВЫ

# ОБЪЯВЛЕНИЕ МАССИВА

— «Старый» способ

```
$eggs = array("egg1", "egg2", "egg3");  
// $eggs[0] == "egg1", $eggs[1] == "egg2", etc...
```

— «Новый» способ

```
$data = [10, 11, 3, 14, 8]; // $data[0] == 10, $data[1] == 11, etc...
```

- С принудительным присваиванием ключей

```
$eggs = array(1 => "egg1", 2 => "egg2", 40 => "o_0");  
$data = [100 => 2, 200 => 4];
```

- Ассоциативный массив

```
$array = array("name" => "Виктор", "city" => "Москва");  
echo $array['name']; // Виктор
```

# РАБОТА С МАССИВОМ

## — Добавление элементов в массив

```
$eggs[] = "новое яйцо"; // ключ назначается автоматически  
$eggs[99] = "уейн грецки"; // принудительное создание элемента
```

## — Удаление элемента из массива

```
unset($eggs[99]); // завершил карьеру
```

# ОБРАБОТКА В ЦИКЛЕ

**важно!!** работает в том случае, если все индексы назначены автоматически и идут от нуля (поведение по умолчанию).

```
1 for ($i=0; $i < count($eggs); $i++) {  
2     echo $eggs[$i];  
3 }
```



# ОБРАБОТКА АССОЦИАТИВНОГО МАССИВА И МАССИВА С ИНДЕКСАМИ НЕ ПО ПОРЯДКУ

```
1 foreach($array as $k => $v) {  
2     echo "Номер: {$k}, значение: $v";  
3 }  
4  
5 $new_array = array();  
6 foreach($array as $v) {  
7     $new_array[] = $v; // "перепаковываем"  ключи  по  порядку  от 0  
8 }
```



## **ФУНКЦИИ ДЛЯ ПРЕОБРАЗОВАНИЯ СТРОК В МАССИВЫ**

# EXPLODE/IMplode

```
array explode (string $delimiter, string $string [, int $limit ])
```

Возвращает массив строк, полученных разбиением строки `string` с использованием `delimiter` в качестве разделителя. Если аргумент `limit` является положительным, возвращаемый массив будет содержать максимум `limit` элементов, при этом последний элемент будет содержать остаток строки `string`.

```
string implode (string $glue, array $pieces)
```

```
string implode (array $pieces)
```

Объединяет элементы массива с помощью строки `glue`.

# STR\_SPLIT

```
array str_split (string $string [, int $split_length = 1 ])
```

Преобразует строку в массив. Если указан необязательный аргумент `split_length`, возвращаемый массив будет содержать части исходной строки длиной `split_length` каждая, иначе каждый элемент будет содержать один символ. Если `split_length` меньше 1, возвращается FALSE. Если `split_length` больше длины строки `string`, то вся строка будет возвращена в первом и единственном элементе массива.



## **ДОБАВЛЕНИЕ / УДАЛЕНИЕ ЭЛЕМЕНТОВ**



# ARRAY\_FILL

```
array array_fill (int $start_index, int $num, mixed $value)
```

Заполняет массив num элементами со значением value, начиная с ключа start\_index.

Если start\_index отрицателен, первым индексом возвращаемого массива будет start\_index, а последующие индексы будут начинаться с нуля.

# ARRAY\_MERGE

```
array array_merge (array $array1 [, array $... ])
```

Сливает элементы одного или большего количества массивов таким образом, что значения одного массива присоединяются к концу предыдущего. Результатом работы функции является новый массив. Если входные массивы имеют одинаковые строковые ключи, тогда каждое последующее значение будет заменять предыдущее. Однако, если массивы имеют одинаковые числовые ключи, значение, упомянутое последним, не заменит исходное значение, а будет добавлено в конец массива.

# ARRAY\_PAD

```
array array_pad (array $array, int $size, mixed $value)
```

Функция `array_pad()` возвращает копию массива `array`, дополненного до размера `size` элементами со значением `value`. Если параметр `size` положителен, то массив увеличивается вправо, если отрицателен — влево. Если абсолютное значение параметра `size` меньше или равно размеру массива `array`, функция не производит никаких операций. За один раз возможно добавить максимум 1048576 элементов.





# ARRAY\_POP

mixed array\_pop (array &\$array)

array\_pop() извлекает и возвращает последнее значение параметра array, уменьшая размер array на один элемент.



# ARRAY\_SHIFT

mixed array\_shift (array &\$array)

array\_shift() извлекает первое значение массива array и возвращает его, сокращая размер array на один элемент. Все числовые ключи будут изменены таким образом, что нумерация массива начнётся с нуля, в то время как строковые ключи останутся прежними.

# ARRAY\_PUSH

```
int array_push (array &$array, mixed $value1 [, mixed $... ])
```

`array_push()` использует `array` как стек, и добавляет переданные значения в конец массива `array`. Длина `array` увеличивается на количество переданных значений. Имеет тот же эффект, что и выражение: `$array[] = $var;` повторенное для каждого переданного значения.

# ARRAY\_UNSHIFT

```
int array_unshift (array &$array, mixed $value1 [, mixed $... ])
```

`array_unshift()` добавляет переданные в качестве аргументов элементы в начало массива `array`. Обратите внимание, что список элементов добавляется целиком, то есть порядок элементов сохраняется. Все числовые ключи будут изменены таким образом, что нумерация массива будет начинаться с нуля, в то время как строковые ключи останутся прежними.



## **ФУНКЦИИ ДЛЯ СОРТИРОВКИ МАССИВОВ**



## SORT/RSORT

```
bool sort (array &$array [, int $sort_flags = SORT_REGULAR ])
```

Эта функция сортирует массив. После завершения работы функции элементы массива будут расположены в порядке возрастания. Дополнительный второй параметр `sort_flags` можно использовать для изменения поведения сортировки.

```
bool rsort (array &$array [, int $sort_flags = SORT_REGULAR ])
```

Эта функция сортирует массив в обратном порядке (от большего к меньшему).

# ASORT/ARSORT

```
bool asort (array &$array [, int $sort_flags = SORT_REGULAR ])
```

Эта функция сортирует массив таким образом, что сохраняются отношения между ключами и значениями. Она полезна, в основном, при сортировке ассоциативных массивов, когда важно сохранить отношение ключ => значение.

```
bool arsort (array &$array [, int $sort_flags = SORT_REGULAR ])
```

Эта функция сортирует массив в обратном порядке таким образом, что сохраняются отношения между ключами и значениями.

# KSORT/KRSORT

```
bool ksort (array &$array [, int $sort_flags = SORT_REGULAR ])
```

Сортирует массив по ключам, сохраняя отношения между ключами и значениями. Эта функция полезна, в основном, для работы с ассоциативными массивами.

```
bool krsort (array &$array [, int $sort_flags = SORT_REGULAR ])
```

Сортирует массив по ключам в обратном порядке, сохраняя отношения между ключами и значениями. Эта функция полезна, в основном, для работы с ассоциативными массивами.



---

## **ФУНКЦИИ ДЛЯ ПОИСКА В МАССИВЕ**

# IN\_ARRAY

```
bool in_array (mixed $needle , array $haystack [, bool $strict = FALSE ])
```

Ищет в haystack значение needle. Если strict не установлен, то при поиске будет использовано нестрогое сравнение.

# ARRAY\_SEARCH

```
mixed array_search (mixed $needle, array $haystack [, bool $strict = false ])
```

Осуществляет поиск данного значения в массиве и возвращает соответствующий ключ в случае удачи. Эта функция может возвращать как boolean FALSE, так и не-boolean значение (ключ), которое приводится к FALSE.



# ARRAY\_KEY\_EXISTS

`bool array_key_exists (mixed $key, array $array)`

Функция `array_key_exists()` возвращает `TRUE`, если в массиве присутствует указанный ключ `key`. Параметр `key` может быть любым значением, которое подходит для индекса массива.

---

## ПОДСЧЕТ ЭЛЕМЕНТОВ В МАССИВЕ



# COUNT

```
int count (mixed $array_or_countable [, int $mode = COUNT_NORMAL ])
```

Подсчитывает количество элементов массива. Если необязательный параметр `mode` установлен в `COUNT_RECURSIVE` (или `1`), `count()` будет рекурсивно подсчитывать количество элементов массива. Это полезно для подсчёта всех элементов многомерных массивов.



## ARRAY\_SUM/ARRAY\_PRODUCT

number array\_sum (array \$array)

array\_sum() возвращает сумму значений массива.

number array\_product (array \$array)

array\_product() возвращает произведение значений массива.



## **ФУНКЦИИ ДЛЯ СРАВНЕНИЯ МАССИВОВ**



# ARRAY\_DIFF

```
array array_diff (array $array1, array $array2 [, array $... ])
```

Сравнивает array1 с одним или несколькими другими массивами и возвращает значения из array1, которые отсутствуют во всех других массивах.

```
1 | $array1 = array("a" => "green", "red", "blue", "red");
2 | $array2 = array("b" => "green", "yellow", "red");
3 | $result = array_diff($array1, $array2);
4 | // Array([1] => blue)
```

# ARRAY\_INTERSECT

```
array array_intersect (array $array1, array $array2 [, array $... ])
```

Функция `array_intersect()` возвращает массив, содержащий все значения массива `array1`, которые содержатся во всех аргументах. Обратите внимание, что ключи сохраняются.



## **ФУНКЦИИ ДЛЯ ПРЕОБРАЗОВАНИЯ МАССИВОВ**

# ARRAY\_FLIP

```
array array_flip (array $array)
```

Функция `array_flip()` возвращает `array` наоборот, то есть ключи массива `array` становятся значениями, а значения массива `array` становятся ключами.

Обратите внимание, что значения массива `array` должны быть корректными ключами, то есть они должны иметь тип `integer` или `string`. Если значение имеет неверный тип, будет выдано предупреждение и данная пара ключ/значение не будет включена в результат.

# ARRAY\_KEYS

```
array array_keys (array $array [, mixed $search_value [, bool $strict = false ]] )
```

Функция `array_keys()` возвращает числовые и строковые ключи, содержащиеся в массиве `array`.

Если указан необязательный параметр `search_value`, функция возвращает только ключи, совпадающие с этим параметром. В обратном случае, функция возвращает все ключи массива `array`.

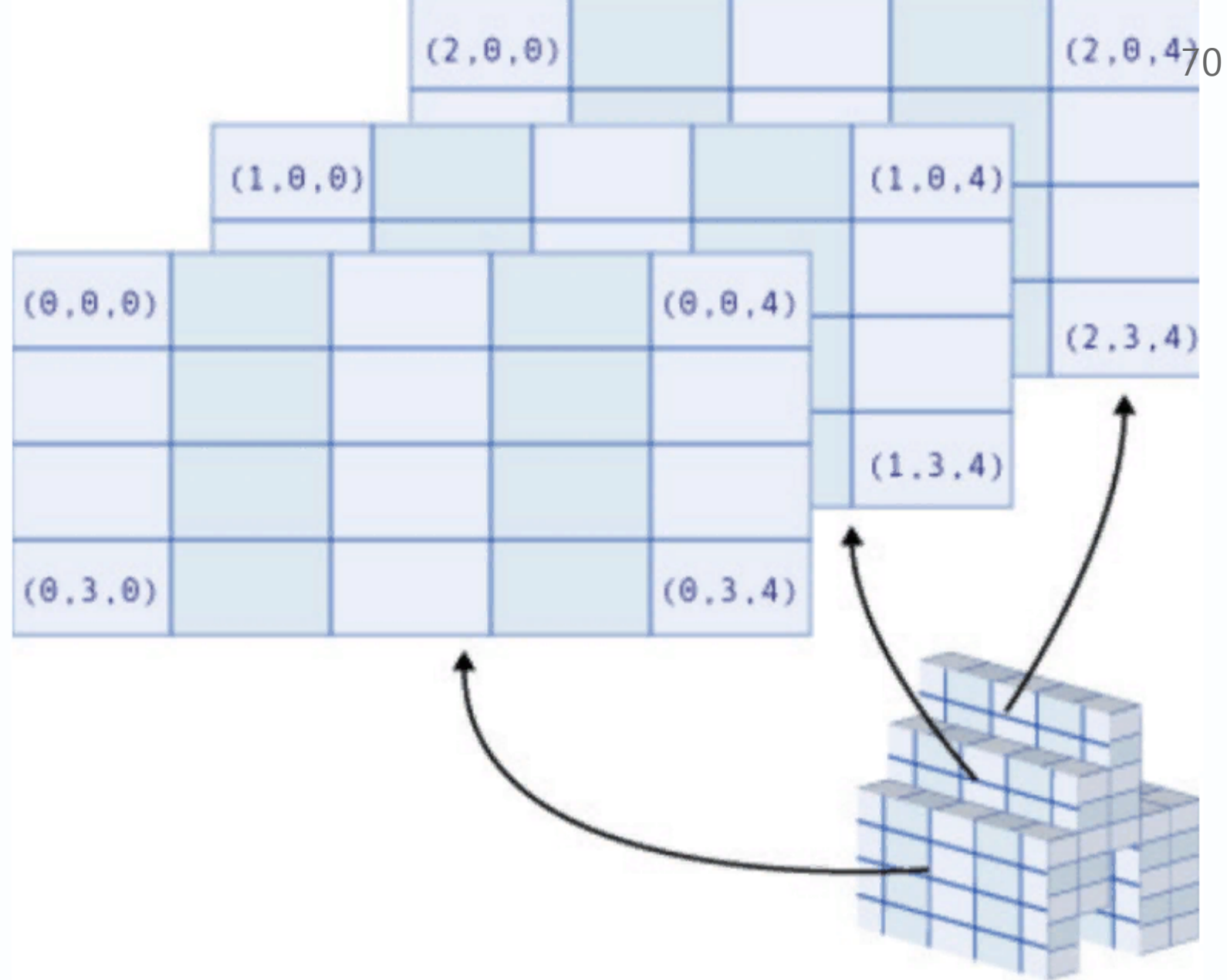


# ARRAY\_CHUNK

```
array array_chunk (array $array, int $size [, bool $preserve_keys = false ])
```

Разбивает массив на несколько массивов размером в `size` элементов. Последний массив из полученных может содержать меньшее количество значений, чем указано в `size`.

Если `size` меньше 1, будет сгенерирована ошибка уровня `E_WARNING` и возвращен `NULL`.





# МНОГОМЕРНЫЙ МАССИВ



## Многомерный массив — это

массив, в котором значениями являются также массивы.

```
1 $a = array(  
2     array("Москва", "Санкт-Петербург", "Нижний Новгород"),  
3     array("Берлин", "Франкфурт", "Мюнхен"),  
4     array("Париж", "Марсель")  
5 );  
6  
7 echo $a[1][2]; // Мюнхен
```

## Многомерный ассоциативный массив

Ключи многомерного массива также могут быть заданы

```
1 $a = array(  
2     "Россия" => array("Москва", "Санкт-Петербург", 6 => "Нижний Новгород"),  
3     "Германия" => array("Берлин", 3 => "Франкфурт", 1 => "Мюнхен"),  
4     "Франция" => array("Париж", "Марсель")  
5 );  
6  
7 echo $a["Россия"][6]; // Н. Новгород
```

# ОБРАБОТКА МНОГОМЕРНОГО МАССИВА

— Вложенные циклы

```
1 foreach($a as $k => $v) {  
2     $country = $k; // страна — ключ  
3     foreach($v as $city) {  
4         // в $city будут попадать города  
5     }  
6 }
```

- Если индексы многомерного массива (и всех внутренних элементов) идут по порядку, то

```
1  for($i = 0; $i < count($a); $i++) {  
2      $row = $a[$i];  
3      for($j = 0; $j < count($row); $j++) {  
4          $item = $row[$j];  
5      }  
6  }
```



## ARRAY\_COLUMN

```
array array_column (array $input, mixed $column_key [, mixed $index_key =  
null ])
```

Берет данные из одной колонки многомерного массива и формирует из них одномерный массив.

В качестве индексов итогового массива могут использоваться данные из колонки, указанной в третьем параметре.

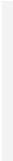
N O T H I N G



**NULL**



## NULL — это



значение для отражения состояния переменной без какого-либо значения, не установленной, не объявленной, удаленной и т.п.

Это единственное возможное значение типа данных null.





# NULL

Переменная считается null, если:

- ей была присвоена соответствующая константа (NULL)
- ей еще не было присвоено никакого значения
- она была удалена с помощью unset

NULL — хороший способ сообщить о том, что что-то пошло не так.



999



999



999



999





# РЕСУРСЫ

## Ресурсы — это

специальный тип данных, содержащий в себе указатель на внешний ресурс (открытый файл, соединение с БД, картинку и т.п.)

В своих приложениях мы получаем ресурсы как результаты тех или иных функций, записываем их в переменную, и затем используем эту переменную, когда требуется обращение к внешним данным.

## ПОЛУЧАЕМ И ИСПОЛЬЗУЕМ РЕСУРСЫ

```
1  try {
2      $dbh = new PDO($dsn, $user, $password);
3  } catch (PDOException $e) {
4      echo 'Подключение не удалось: ' . $e->getMessage();
5  }
6  $sth = $dbh->prepare("SELECT name, colour FROM fruit");
7  $sth->execute();
8  $result = $sth->fetch(PDO::FETCH_ASSOC);
9  print_r($result);
10 print("\n");
```



## **В БУДУЩИХ ЛЕКЦИЯХ РЕСУРСЫ БУДУТ РАССМОТРЕНЫ ПОДРОБНЕЕ**

Список ресурсов (для смелых): <http://php.net/manual/ru/resource.php>



**нетология**  
университет интернет-профессий

**Задавайте вопросы и напишите отзыв о лекции!**

**СЕРГЕЙ ГЕРАСИМЕНКО**



[gerasimenkosv@bk.ru](mailto:gerasimenkosv@bk.ru)