Работа с функциями для массивов в PHP

Изучите методы и функции: count, in_array, array_sum, array_product, range, array_merge, array_slice,array_splice, array_keys, array_values, array_combine, array_flip, array_reverse, array_search,array_replace, array_count_values, функции для сортировки, array_rand, shuffle, array_unique,array_shift, array_pop, array_unshift, array_push, array_pad, array_fill, array_fill_keys, array_chunk,array_count_values, array_map, array_intersect, array_diff, array_map.

Функция count

Функция **count** подсчитывает количество элементов массива.

Синтаксис

```
count(массив);
```

Примеры

Пример

В данном примере функция вернула количество элементов массива:

```
<?php
        $arr = ['a', 'b', 'c', 'd'];
        echo count($arr);
?>
```

Результат выполнения кода:

4

Функция in_array

Функция **in_array** проверяет наличие заданного элемента в массиве.

См. также функцию array_key_exists, которая проверяет наличие указанного ключа в массиве.

Синтаксис

```
in_[что искать, в каком массиве];
```

Примеры

Пример

Давайте проверим, есть ли в массиве **$arr** элемент со значением **3**:

```
<?php
$arr = [1, 2, 3, 4, 5];
var_dump(in_[3, $arr]);
?>
```

Результат выполнения кода:

```
true
```

Функция array_sum

Функция **array_sum** вычисляет сумму элементов массива.

Синтаксис

```
array_sum(массив);
```

Примеры

Пример

Давайте найдем сумму элементов массива:

```
<?php
        $arr = [1, 2, 3, 4, 5];  echo array_sum($arr);?>
```

Результат выполнения кода:

```
15
```

Пример . Применение

Давайте найдем сумму цифр числа. Для этого разобьем число в массив с помощью str_split и сложим элементы этого массива с помощью **array_sum**:

```
<?php
        $num = 12345;    echo array_sum(str_split($num, 1));?>
```

Результат выполнения кода:

```
15
```

Функция array_product

Функция **array_product** вычисляет произведение (умножение) элементов массива.

Синтаксис

```
array_product(массив);
```

Примеры

Пример

Давайте найдем произведение элементов массива:

```
<?php
        $arr = [1, 2, 3, 4, 5];
        echo array_product($arr);
?>
```

Результат выполнения кода:

```
120
```

Функция range

Функция **range** создает массив с диапазоном элементов. К примеру, можно создать массив, заполненный числами от 1 до 100 или буквами от 'a' до 'z'. Диапазон, который сгенерирует функция, задается параметрами: первый параметр откуда генерировать, а второй - докуда.

Третий необязательный параметр функции задает шаг. К примеру, можно сделать ряд 1, 3, 5, 7, если задать шаг 2, или ряд 1, 4, 7, 10 если задать шаг 3.

См. также функцию array_fill, которая заполняет массив заданным значением.

См. также функцию array_pad, которая дополняет массив заданным значением до нужного размера.

Синтаксис

```
range(откуда, докуда, [шаг]);
```

Примеры

Пример

Давайте создадим массив, заполненный числами от 1 до 5:

```
<?php
        $arr = range(1, 5);
        var_dump($arr);
?>
```

Результат выполнения кода:

```
[1, 2, 3, 4, 5]
```

Пример

Давайте создадим массив, заполненный числами от 5 до 1:

```
<?php
        $arr = range(5, 1);
        var_dump($arr);
?>
```

Результат выполнения кода:

[5, 4, 3, 2, 1]

Пример

Давайте создадим массив, заполненный числами от 0 до 10 с шагом 2:

```
<?php
        $arr = range(0, 10, 2);
        var_dump($arr);
?>
```

Результат выполнения кода:

[0, 2, 4, 6, 8, 10]

Пример

Давайте создадим массив, заполненный буквами от 'a' до 'e':

```
<?php
        $arr = range('a', 'e');
        var_dump($arr);
?>
```

Результат выполнения кода:

['a', 'b', 'c', 'd', 'e']

Функция array_merge

Функция **array_merge** сливает два и более массивов вместе.

Если в сливаемых массивах встречаются одинаковые ключи - останется только один из таких элементов.

Если вам нужно, чтобы остались все элементы с одинаковыми ключами - используйте функцию array_merge_recursive.

См. также функцию array_combine, которая сливает два массива в один ассоциативный.

Синтаксис

```
array_merge(первый массив, второй массив...);
```

Примеры

Пример

В данном примере функция слила массивы:

```php
<?php
        $arr1 = [1, 2, 3];
        $arr2 = ['a', 'b', 'c'];
        $res = array_merge($arr1, $arr2);
        var_dump($res);
?>
```

Результат выполнения кода:

```
[1, 2, 3, 'a', 'b', 'c']
```

Функция array_slice

Функция **array_slice** отрезает и возвращает часть массива. При этом сам массив не меняется. Первым параметром указывается массив для разрезания. Вторым параметром указывается, с какого элемента начинать отрезание, а третьим - сколько элементов отрезать. Третий параметр может быть отрицательным - в этом случае отсчет начнется с конца (-1 - последний элемент, -2 - предпоследний и так далее). Третий параметр можно вообще не указывать - в этом случае массив отрежется до самого конца.

Последний необязательный параметр регулирует сохранять ли ключи при отрезании, true - сохранять, false (по умолчанию) - не сохранять.

См. также функцию array_splice, которая отрезает часть массива, изменяя при этом сам массив.

Синтаксис

```
array_slice(массив, откуда отрезать, [сколько], [сохранять ключи = false]);
```

Примеры

Пример

Давайте вырежем элементы с первого (имеет номер 0), 3 штуки:

```php
<?php
        $arr = ['a', 'b', 'c', 'd', 'e'];
        var_dump(array_slice($arr, 0, 3));
?>
```

Результат выполнения кода:

```
['a', 'b', 'c']
```

Пример

Давайте вырежем элементы со второго (имеет номер 1), 3 штуки:

```php
<?php
        $arr = ['a', 'b', 'c', 'd', 'e'];
        var_dump(array_slice($arr, 1, 3));
?>
```

Результат выполнения кода:

['b', 'c', 'd']

Пример

Давайте вырежем элементы со второго (имеет номер 1) до конца массива. Для этого третий параметр не пишем:

```php
<?php
        $arr = ['a', 'b', 'c', 'd', 'e'];
        var_dump(array_slice($arr, 1));
?>
```

Результат выполнения кода:

['b', 'c', 'd', 'e']

Пример

Давайте вырежем элементы с предпоследнего, 2 штуки. Для этого второй параметр установим в -2 (позиция предпоследнего элемента):

```php
<?php
        $arr = ['a', 'b', 'c', 'd', 'e'];
        var_dump(array_slice($arr, -2, 2));
?>
```

Результат выполнения кода:

['d', 'e']

Пример

По умолчанию массив не сохраняет ключи при вырезании:

```php
<?php
        $arr = ['a'=>1, 'b'=>2, 'c'=>3, 'd'=>4, 'e'=>5];
        var_dump(array_slice($arr, 0, 3));
?>
```

Результат выполнения кода:

[1, 2, 3]

Пример

Давайте сделаем так, чтобы ключи сохранялись. Для этого последний параметр установим как true:

```
<?php
        $arr = ['a'=>1, 'b'=>2, 'c'=>3, 'd'=>4, 'e'=>5];
        var_dump(array_slice($arr, 0, 3, true));
?>
```

Результат выполнения кода:

['a'=>1, 'b'=>2, 'c'=>3]

Функция array_splice

Функция **array_splice** отрезает и возвращает часть массива. При этом отрезанная часть исчезает из массива. Вместо отрезанной части можно вставлять новые элементы.

Первым параметром указывается массив для разрезания. Вторым параметром указывается, с какого элемента начинать отрезание, а третьим - сколько элементов отрезать. Третий параметр может быть отрицательным - в этом случае отсчет начнется с конца (-1 - последний элемент, -2 - предпоследний и так далее). Третий параметр можно вообще не указывать - в этом случае массив отрежется до самого конца.

В последнем необязательным параметре можно задавать массив элементов, которые будут вставлены взамен удаленных.

См. также функцию array_slice, которая отрезает часть массива, не изменяя при этом сам массив.

Синтаксис

```
array_splice(массив, откуда отрезать, [сколько], [вставить взамен]);
```

Примеры

Пример

Давайте вырежем элементы с первого (имеет номер 0), 3 штуки:

```
<?php
        $arr = ['a', 'b', 'c', 'd', 'e'];
        var_dump(array_splice($arr, 0, 3));
?>
```

Результат выполнения кода:

['a', 'b', 'c']

При этом массив $arr примет следующий вид:

['d', 'e']

Пример

Давайте вырежем элементы со второго (имеет номер 1), 3 штуки:

```
<?php
        $arr = ['a', 'b', 'c', 'd', 'e'];
        var_dump(array_splice($arr, 1, 3));
?>
```

Результат выполнения кода:

['b', 'c', 'd']

При этом массив $arr примет следующий вид:

['a', 'e']

Пример

Давайте вырежем элементы со второго (имеет номер 1) до конца массива. Для этого третий параметр не пишем:

```
<?php
        $arr = ['a', 'b', 'c', 'd', 'e'];
        var_dump(array_splice($arr, 1));
?>
```

Результат выполнения кода:

['b', 'c', 'd', 'e']

При этом массив $arr примет следующий вид:

['a']

Пример

Давайте вырежем элементы с предпоследнего, 2 штуки. Для этого второй параметр установим в -2 (позиция предпоследнего элемента):

```
<?php
        $arr = ['a', 'b', 'c', 'd', 'e'];
        var_dump(array_splice($arr, -2, 2));
?>
```

Результат выполнения кода:

['d', 'e']

При этом массив $arr примет следующий вид:

['a', 'b', 'c']

Пример

Давайте вырежем элементы со второго (имеет номер 1), 2 штуки, а в замен вставим элементы 1, 2, 3:

```
<?php
        $arr = ['a', 'b', 'c', 'd', 'e'];
```

```
        var_dump(array_splice($arr, 1, 2, [1, 2, 3]));
?>
```

Результат выполнения кода:

['b', 'c']

При этом массив $arr примет следующий вид:

['a', 1, 2, 3, 'd', 'e']

Пример

Давайте вообще ничего не будем вырезать, а просто вставим элементы 1, 2, 3 начиная с позиции 1. Для этого третий параметр ставится в ноль:

```
<?php
        $arr = ['a', 'b', 'c', 'd', 'e'];
        var_dump(array_splice($arr, 1, 0, [1, 2, 3]));
?>
```

Результат выполнения кода:

[]

При этом массив $arr примет следующий вид:

['a', 1, 2, 3, 'b', 'c', 'd', 'e']

Функция array_keys

Функция **array_keys** получает ключи массива и записывает их в новый массив.

См. также функцию array_values, которая получает элементы массива.

Синтаксис

```
array_keys(массив);
```

Примеры

Пример

В данном примере функция получила ключи из массива:

```
 <?php
        $arr = ['a'=>1, 'b'=>2, 'c'=>3, 'd'=>4, 'e'=>5];
        var_dump(array_keys($arr));
?>
```

Результат выполнения кода:

['a', 'b', 'c', 'd', 'e']

Функция array_values

Функция **array_values** выбирает все значения из массива.

См. также функцию array_keys, которая возвращает ключи массива.

Синтаксис

```
array_values(массив);
```

Примеры

Пример

Давайте получим все значения массива:

```
<?php
        $arr = ['a'=>1, 'b'=>2, 'c'=>3, 'd'=>4, 'e'=>5];
        var_dump(array_values($arr));
?>
```

Результат выполнения кода:

[1, 2, 3, 4, 5]

Функция array_combine

Функция **array_combine** осуществляет слияние двух массивов в один ассоциативный.

Первым параметром функция принимает массив будущих ключей, а вторым - массив будущих значений.

См. также функцию array_merge, которая сливает несколько массивов в один.

См. также функции array_keys и array_values которые которые позволяют извлечь ключи и значения массива.

Синтаксис

```
array_combine(массив ключей, массив значений);
```

Примеры

Пример

В данном примере функция сольет два массива в один ассоциативный. При этом соответствующие элементы из первого массива станут ключами элементов из второго массива:

```
<?php
        $keys = ['green' ,'blue' ,'red'];
        $elems = ['зеленый', 'голубой', 'красный'];
```

```
        $arr = array_combine($keys, $elems);
        var_dump($arr);
?>
```

Результат выполнения кода:

['green'=>'зеленый', 'blue'=>'голубой', 'red'=>'красный']

Функция array_flip

Функция **array_flip** производит обмен местами ключей и значений массива.

См. также функцию array_reverse, которая переворачивает массив.

Синтаксис

```
array_flip(массив);
```

Примеры

Пример

Давайте поменяем местами ключи и значения массива:

```
<?php
        $arr = ['a'=>1, 'b'=>2, 'c'=>3];
        $result = array_flip($arr);
        var_dump($result);
?>
```

Результат выполнения кода:

[1=>'a', 2=>'b', 3=>'c']

Функция array_reverse

Функция **array_reverse** переворачивает массив в обратном порядке.

Первым параметром передается массив, а вторым - сохранять ключи при перестановке элементов или нет (true - да, false - нет).

Второй параметр указывать необязательно. В таком случае по умолчанию вторым параметром является false.

См. также функцию array_flip, которая меняет местами ключи и значения.

Синтаксис

```
array_reverse(массив, [сохранять ли ключи]);
```

Примеры

Пример

Давайте перевернем массив:

```
<?php
        $arr = [1, 2, 3, 4, 5];
        $arr = array_reverse($arr);
        var_dump($arr);
?>
```

Результат выполнения кода:

[5, 4, 3, 2, 1]

Пример

Учтите, что по умолчанию ключи оторвутся от элементов:

```
<?php
        $arr = ['a'=>1, 'b'=>2, 'c'=>3, 'd'=>4, 'e'=>5];
        $arr = array_reverse($arr);
        var_dump($arr);
?>
```

Результат выполнения кода:

['a'=>5, 'b'=>4, 'c'=>3, 'd'=>2, 'e'=>1]

Пример

Чтобы ключи переворачивались вместе с массивом - нужно передать второй параметр:

```
<?php
        $arr = ['a'=>1, 'b'=>2, 'c'=>3, 'd'=>4, 'e'=>5];
        $arr = array_reverse($arr, true);
        var_dump($arr);
?>
```

Результат выполнения кода:

['e'=>5, 'd'=>4, 'c'=>3, 'b'=>2, 'a'=>1]

Функция array_search

Функция **array_search** осуществляет поиск значения в массиве и возвращает ключ первого найденного элемента. Если такой элемент не найдет - вернет false. Третий параметр задает строгое сравнение по типу (как по ===). Если поставить true - он будет сравнивать строго, а если false (по умолчанию) - то нет.

Синтаксис

```
array_search(что ищем, где ищем, [сравнивать по типу = false]);
```

Примеры

Пример

Найдем в массиве элемент со значением 'c' - в результате получим его ключ (он равен 2):

```
<?php
        $arr = ['a', 'b', 'c', 'd', 'e'];
        echo array_search('c', $arr));
?>
```

Результат выполнения кода:

```
2
```

Функция array_replace

Функция **array_replace** заменяет значения первого массива значениями с такими же ключами из других переданных массивов. Если ключ из первого массива присутствует во втором массиве, его значение заменяется на значение из второго массива. Если ключ есть во втором массиве, но отсутствует в первом - он будет создан в первом массиве. Если ключ присутствует только в первом массиве, то сохранится как есть.

Если для замены передано несколько массивов, они будут обработаны в порядке передачи и более поздние массивы будут перезаписывать значения из предыдущих.

Синтаксис

```
array_replace(массив, массив, массив...);
```

Примеры

Пример

Давайте заменим элемент с ключом 0 на '!', а элемент с ключом 2 - на '!!':

```
<?php
        $arr = ['a', 'b', 'c', 'd', 'e'];
        $result = array_replace($arr, [0=>'!', 2=>'!!']));
        var_dump($result);
?>
```

Результат выполнения кода:

```
['!', 'b', '!!', 'd', 'e']
```

Функция array_count_values

Функция **array_count_values** производит подсчет количества всех значений массива. Возвращает ассоциативный массив, в котором ключами будут элементы массива, а значениями - их количество в массиве.

См. также функцию count, которая подсчитывает количество элементов в массиве.

Синтаксис

```
array_count_values(массив);
```

Примеры

Пример

Подсчитаем, сколько раз встречается каждый из элементов:

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

Результат выполнения кода:

```
['a'=>3, 'b'=>2, 'c'=>1]
```

Сортировка массивов

Для сортировки массивов в PHP существует несколько функций: **sort** - по возрастанию элементов,**rsort** - по убыванию элементов, **asort** - по возрастанию элементов с сохранением ключей, **arsort** - по убыванию элементов с сохранением ключей, **ksort** - по возрастанию ключей, **krsort** - по убыванию ключей, **usort** - по функции по элементам, **uasort** - по функции по элементам с сохранением ключей,**uksort** - по функции по ключам, **natsort** - натуральная сортировка.

Все эти функции изменяют сам массив - это значит, что результат не нужно никуда присваивать: поменяется сам массив.

См. так функцию array_multisort.

Синтаксис

```
sort(массив);
```

Примеры

Пример

В данном примере функция отсортирует массив по возрастанию элементов:

```
<?php
        $arr = [1, 3, 2, 5, 4];
        sort($arr);
        var_dump($arr);
```

```
?>
```

Результат выполнения кода:

[1, 2, 3, 4, 5]

Функция array_rand

Функция **array_rand** возвращает случайный ключ из массива.

Второй необязательный параметр указывает, сколько случайных ключей следует вернуть.

Если он не указан - возвращается один ключ, а если указан - заданное количество ключей в виде массива.

См. также функцию shuffle, которая перемешивает массив.

См. также функцию mt_rand, которая возвращает случайное число в заданном диапазоне.

Синтаксис

```
array_rand(массив, [сколько ключей выбрать]);
```

Примеры

Пример

В данном примере функция вернет случайный ключ из массива:

```
<?php
        $arr = ['a'=>1, 'b'=>2, 'c'=>3, 'd'=>4, 'e'=>5];
        $key = array_rand($arr);
        var_dump($key);
?>
```

Результат выполнения кода:

c

Пример

Давайте вернем случайный элемент из массива, зная случайный ключ:

```
<?php
        $arr = ['a'=>1, 'b'=>2, 'c'=>3, 'd'=>4, 'e'=>5];
        $key = array_rand($arr);
        var_dump($arr[$key]);
?>
```

Результат выполнения кода:

3

Пример

Сейчас мы задали второй параметр и теперь функция вернет массив из 3-х случайных ключей (3 ключа - так как второй параметр равен 3):

```php
<?php
        $arr = ['a'=>1, 'b'=>2, 'c'=>3, 'd'=>4, 'e'=>5];
        $keys = array_rand($arr, 3);
        var_dump($keys);
?>
```

Результат выполнения кода:

['a', 'b', 'e']

Функция array_unique

Функция **array_unique** осуществляет удаление повторяющихся элементов (дублей) из массива.

См. также функцию array_count_values, которая подсчитывает количество всех значений массива.

Синтаксис

```php
array_unique(массив);
```

Примеры

Пример

Давайте удалим дубли из массива:

```php
<?php
        $arr = [1, 1, 1, 2, 2, 3];
        $arr = array_unique($arr);
        var_dump($arr);
?>
```

Результат выполнения кода:

[1, 2, 3]

Функция array_shift

Функция **array_shift** вырезает и возвращает первый элемент массива. При этом этот элемент исчезает из массива.

См. также функцию array_pop, которая вырезает последний элемент массива.

Синтаксис

```php
array_shift(массив);
```

Примеры

Пример

В данном примере функция вырезала первый элемент массива:

```
<?php
        $arr = [1, 2, 3, 4, 5];
        echo array_shift($arr);
?>
```

1

Массив $arr станет выглядеть так:

[2, 3, 4, 5]

Функция array_pop

Функция **array_pop** вырезает и возвращает последний элемент массива. При этом этот элемент исчезает из массива.

См. также функцию array_shift, которая вырезает первый элемент массива.

Синтаксис

```
array_pop(массив);
```

Примеры

Пример

В данном примере функция вырезала и вернула последний элемент массива:

```
<?php
        $arr = [1, 2, 3, 4, 5];
        echo array_pop($arr);
?>
```

Результат выполнения кода:

5

Массив $arr станет выглядеть так:

[1, 2, 3, 4]

Функция array_unshift

Функция **array_unshift** добавляет элементы в начало массива.

Функция изменяет переданный массив, а возвращает новое количество элементов в массиве.

См. также функцию array_shift, которая извлекает первый элемент массива.

Синтаксис

```
array_unshift(массив, какие элементы добавить);
```

Примеры

Пример

Давайте добавим в массив еще 2 элемента:

```
<?php
        $arr = [1, 2, 3];
        $num = array_unshift($arr, 'a', 'b');
        var_dump($arr);
?>
```

Результат выполнения кода:

['a', 'b', 1, 2, 3]

В переменной $num будет лежать новое количество элементов массива:

5

Функция array_push

Функция **array_push** добавляет элементы в конец массива.

Функция изменяет переданный массив, а возвращает новое количество элементов в массиве.

См. также функцию array_pop, которая вырезает последний элемент массива.

Синтаксис

```
array_push(массив к которому добавить элементы, какие элементы добавить);
```

Примеры

Пример

Давайте добавим элементы в конец массива:

```
<?php
        $arr = [1, 2, 3];
        $num = array_push($arr, 4, 5);
        var_dump($arr);
?>
```

Результат выполнения кода:

[1, 2, 3, 4, 5]

В переменной $num будет лежать новое количество элементов массива:

5

Функция array_pad

Функция **array_pad** дополняет массив определенным значением до заданного размера.

См. также функцию array_fill, которая заполняет массив заданным значением.

См. также функцию range, которая создает массив с диапазоном элементов.

Синтаксис

```
array_pad(массив, до какого размера заполнить, чем заполнять);
```

Второй параметр можно делать отрицательным - в этом случае массив будет дополнятся элементами не с конца, а с начала.

Примеры

Пример

Давайте заполним массив нулями так, чтобы его размер стал 5 элементов:

```
<?php
        $arr = [1, 2, 3];
        $arr = array_pad($arr, 5, 0);
        var_dump($arr);
?>
```

Результат выполнения кода:

[1, 2, 3, 0, 0]

Пример

А теперь в массиве уже есть 5 элементов - поэтому он ничем не заполнится:

```
<?php
        $arr = [1, 2, 3, 4, 5, 6];
        $arr = array_pad($arr, 5, 0);
        var_dump($arr);
?>
```

Результат выполнения кода:

[1, 2, 3, 4, 5, 6]

Пример

Давайте второй параметр сделаем отрицательным. В этом случае массив заполнится сначала, а не с конца:

```php
<?php
        $arr = [1, 2, 3];
        $arr = array_pad($arr, -5, 0);
        var_dump($arr);
?>
```

Результат выполнения кода:

```
[0, 0, 1, 2, 3]
```

Функция array_fill

Функция **array_fill** создает массив, заполненный элементами с определенным значением.

См. также функцию array_pad, которая дополняет массив заданными элементами до определенного значения.

См. также функцию range, которая создает массив с диапазоном элементов.

Синтаксис

```
array_fill(ключ первого элемента, сколько элементов, чем заполнять);
```

Примеры

Пример

Заполним массив 5-ю элементами с текстом **'x'**. Так как первый параметр 0, то ключи начнут свою нумерацию с нуля:

```php
<?php
        $arr = array_fill(0, 5, 'x');
        var_dump($arr);
?>
```

Результат выполнения кода:

```
[0=>'x', 1=>'x', 2=>'x', 3=>'x', 4=>'x']
```

Пример

Заполним массив 5-ю элементами с текстом **'x'**. Так как первый параметр 3, то ключи начнут свою нумерацию с трех:

```php
<?php
        $arr = array_fill(3, 5, 'x');
        var_dump($arr);
?>
```

Результат выполнения кода:

[3=>'x', 4=>'x', 5=>'x', 6=>'x', 7=>'x']

Пример

Давайте сделаем массив [['x', 'x', 'x'], ['x', 'x', 'x'], ['x', 'x', 'x']]:

```
<?php
        $arr = array_fill(0, 3, array_fill(0, 5, 'x'));
        var_dump($arr);
?>
```

Результат выполнения кода:

[['x', 'x', 'x'], ['x', 'x', 'x'], ['x', 'x', 'x']]

Функция array_fill_keys

Функция **array_fill_keys** создает массив и заполняет массив элементами с определенным значением так, чтобы весь массив был с одинаковыми элементами, но разными ключами. Ключи берутся из массива, передаваемого первым параметром.

См. также функцию array_fill, которая заполняет массив значениями без заданных ключей.

Синтаксис

```
array_fill_keys(ключ первого элемента, сколько элементов, чем заполнять);
```

Примеры

Пример

Заполним массив 5-ю элементами с текстом **'x'**. Так как первый параметр 0, то ключи начнут свою нумерацию с нуля:

```
<?php
        $arr = array_fill_keys(0, 5, 'x');
        var_dump($arr);
?>
```

Результат выполнения кода:

[0=>'x', 1=>'x', 2=>'x', 3=>'x', 4=>'x']

Пример

Заполним массив 5-ю элементами с текстом **'x'** с ключами из массива ['a', 'b', 'c', 'd', 'e']:

```
<?php
        $arr = array_fill_keys(['a', 'b', 'c', 'd', 'e'], 'x');
        var_dump($arr);
```

```
?>
```

Результат выполнения кода:

```
['a'=>'x', 'b'=>'x', 'c'=>'x', 'd'=>'x', 'e'=>'x']
```

Функция array_chunk

Функция **array_chunk** разбивает одномерный массив в двухмерный. Первым параметром она принимает массив, а вторым - количество элементов в каждом подмассиве.

Синтаксис

```
array_chunk(массив, по сколько элементов);
```

Примеры

Пример

Давайте разобьем массив по два элемента в подмассиве:

```
<?php
        $arr = ['a', 'b', 'c', 'd'];
        var_dump(array_chunk($arr, 2));
?>
```

Результат выполнения кода:

```
[    ['a', 'b'],
   ['c', 'd']
];
```

Пример

Давайте разобьем массив по 3 элемента в подмассиве. Обратите внимание на то, что в последнем подмассиве два элемента, а не 3, так как ему не хватило элементов:

```
<?php
        $arr = ['a', 'b', 'c', 'd', 'e'];
        var_dump(array_chunk($arr, 2));
?>
```

Результат выполнения кода:

```
[    ['a', 'b', 'c'],
   ['d', 'e']
];
```

Функция array_count_values

Функция **array_count_values** производит подсчет количества всех значений массива. Возвращает ассоциативный массив, в котором ключами будут элементы массива, а значениями - их количество в массиве.

См. также функцию count, которая подсчитывает количество элементов в массиве.

Синтаксис

```
array_count_values(массив);
```

Примеры

Пример

Подсчитаем, сколько раз встречается каждый из элементов:

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

Результат выполнения кода:

```
['a'=>3, 'b'=>2, 'c'=>1]
```

Функция array_map

Функция **array_map** применяет заданную функцию ко всем элементам массива и возвращает измененный массив.

Первым параметром функция принимает имя функции, а вторым - массив. Можно передавать дополнительные массивы третьим и так далее параметрами.

Синтаксис

```
array_map(имя функции в кавычках, массив, [еще массивы через запятую]);
```

Примеры

Пример

В данном примере функция **array_map** извлекла квадратный корень из каждого элемента массива (с помощью функции sqrt) и записала в новый массив:

```
<?php
        $arr = [1, 4, 9];
        $result = array_map('sqrt', $arr);
        var_dump($result);
?>
```

Результат выполнения кода:

```
[1, 2, 3]
```

Функция array_intersect

Функция **array_intersect** вычисляет пересечение массивов - возвращает массив из элементов, которые есть во всех массивах, переданных в функцию.

См. также функцию array_diff, которая вычисляет разность массивов.

Синтаксис

```
array_intersect(массив, массив, массив...);
```

Примеры

Пример

Давайте найдем, какие элементы есть и в одном, и в другом массиве (это 3, 4, 5):

```
<?php
        $arr1 = array (1, 2, 3, 4, 5);
        $arr2 = array (3, 4, 5, 6, 7);
        $result = array_intersect($arr1, $arr2);
        var_dump($result);
?>
```

Результат выполнения кода:

[3, 4, 5]

Функция array_diff

Функция **array_diff** возвращает массив из элементов, которые не являются общими для всех массивов, переданных в функцию.

См. также функцию array_intersect, которая вычисляет пересечение массивов.

Синтаксис

```
array_diff(массив, массив, массив...);
```

Примеры

Пример

Давайте найдем, какие элементы есть в одном массиве и при этом отсутствуют в другом (это 1, 2, 6, 7):

```
<?php
        $arr1 = array (1, 2, 3, 4, 5);
        $arr2 = array (3, 4, 5, 6, 7);
```

```
        $result = array_intersect($arr1, $arr2);
        var_dump($result);
?>
```

Результат выполнения кода:

[1, 2, 6, 7]

Функция array_map

Функция **array_map** применяет заданную функцию ко всем элементам массива и возвращает измененный массив.

Первым параметром функция принимает имя функции, а вторым - массив. Можно передавать дополнительные массивы третьим и так далее параметрами.

Синтаксис

```
array_map(имя функции в кавычках, массив, [еще массивы через запятую]);
```

Примеры

Пример

В данном примере функция **array_map** извлекла квадратный корень из каждого элемента массива (с помощью функции sqrt) и записала в новый массив:

```
<?php
        $arr = [1, 4, 9];
        $result = array_map('sqrt', $arr);
        var_dump($result);
?>
```

Результат выполнения кода:

[1, 2, 3]

Задачи на функции работы с массивами в PHP

Примеры решения задач

Задача

Задача. Создайте массив, заполненный числами от **1** до **100**. Найдите **сумму** элементов данного массива.

Решение: для начала создадим массив с числами от **1** до **100**. Вручную это сделать будет очень сложно, поэтому воспользуемся функцией range:

```php
<?php
        $arr = range(1, 100);
?>
```

Сумму элементов полученного массива найдем с помощью функции array_sum (можно использовать цикл **foreach**, как мы это делали раньше, но **array_sum** в данном случае гораздо удобнее и проще):

```php
<?php
        //Найдем сумму элементов нашего массива:
        echo array_sum(range(1, 100));
?>
```

Задача . Функция array_map

Задача. Дан массив с элементами **'a', 'b', 'c', 'd', 'e'**. С помощью функции array_map сделайте из него массив **'A', 'B', 'C', 'D', 'E'**.

Решение: с помощью функции **array_map** задача решается в одну строчку - первым параметром передадим ей функцию strtoupper, а вторым - массив, к каждому элементу которого мы хотим применить **strtoupper**:

```php
<?php
        $arr = ['a', 'b', 'c', 'd', 'e'];
        $arr = array_map('strtoupper', $arr);
        var_dump($arr);
?>
```

Задачи для решения

Работа с count

Для решения задач данного блока вам понадобятся следующие функции: count.

1 Дан массив **$arr**. Подсчитайте количество элементов этого массива.

2 Дан массив **$arr**. С помощью функции **count** выведите последний элемент данного массива.

Работа с in_array

Для решения задач данного блока вам понадобятся следующие функции: in_array.

3 Дан массив с числами. Проверьте, что в нем есть элемент со значением **3**.

Работа с array_sum и array_product

Для решения задач данного блока вам понадобятся следующие функции: array_sum, array_product.

4 Дан массив **[1, 2, 3, 4, 5]**. Найдите сумму элементов данного массива.

5 Дан массив **[1, 2, 3, 4, 5]**. Найдите произведение (умножение) элементов данного массива.

6 Дан массив **$arr**. С помощью функций array_sum и count найдите среднее арифметическое элементов (сумма элементов делить на их количество) данного массива.

Работа с range

Для решения задач данного блока вам понадобятся следующие функции: range.

7 Создайте массив, заполненный числами от **1** до **100**.

8 Создайте массив, заполненный буквами от **'a'** до **'z'**.

9 Создайте строку **'1-2-3-4-5-6-7-8-9'** не используя цикл. Показать подсказку.

10 Найдите сумму чисел от **1** до **100** не используя цикл.

11 Найдите произведение чисел от **1** до **10** не используя цикл.

Работа с array_merge

Для решения задач данного блока вам понадобятся следующие функции: array_merge.

12 Даны два массива: первый с элементами **1, 2, 3**, второй с элементами **'a', 'b', 'c'**. Сделайте из них массив с элементами **1, 2, 3, 'a', 'b', 'c'**.

Работа с array_slice

Для решения задач данного блока вам понадобятся следующие функции: array_slice.

13 Дан массив с элементами **1, 2, 3, 4, 5**. С помощью функции **array_slice** создайте из него массив **$result** с элементами **2, 3, 4**.

Работа с array_splice

Для решения задач данного блока вам понадобятся следующие функции: array_splice.

14 Дан массив **[1, 2, 3, 4, 5]**. С помощью функции **array_splice** преобразуйте массив в **[1, 4, 5]**.

15 Дан массив **[1, 2, 3, 4, 5]**. С помощью функции **array_splice** запишите в новый массив элементы **[2, 3, 4]**.

16 Дан массив **[1, 2, 3, 4, 5]**. С помощью функции **array_splice** сделайте из него массив **[1, 2, 3, 'a', 'b', 'c', 4, 5]**.

17 Дан массив **[1, 2, 3, 4, 5]**. С помощью функции **array_splice** сделайте из него массив **[1, 'a', 'b', 2, 3, 4, 'c', 5, 'e']**.

Работа с array_keys, array_values, array_combine

Для решения задач данного блока вам понадобятся следующие функции: array_keys, array_values,array_combine.

18 Дан массив **'a'=>1, 'b'=>2, 'c'=>3'**. Запишите в массив **$keys** ключи из этого массива, а в **$values** – значения.

19 Даны два массива: **['a', 'b', 'c']** и **[1, 2, 3]**. Создайте с их помощью массив **'a'=>1, 'b'=>2, 'c'=>3'**.

Работа с array_flip, array_reverse

Для решения задач данного блока вам понадобятся следующие функции: array_flip, array_reverse.

20 Дан массив **'a'=>1, 'b'=>2, 'c'=>3**. Поменяйте в нем местами ключи и значения.

21 Дан массив с элементами **1, 2, 3, 4, 5**. Сделайте из него массив с элементами **5, 4, 3, 2, 1**.

Работа с array_search

Для решения задач данного блока вам понадобятся следующие функции: array_search.

22 Дан массив ['a', '-', 'b', '-', 'c', '-', 'd']. Найдите позицию первого элемента '-'.

23 Дан массив ['a', '-', 'b', '-', 'c', '-', 'd']. Найдите позицию первого элемента '-' и удалите его с помощью функции **array_splice**.

Работа с array_replace

Для решения задач данного блока вам понадобятся следующие функции: array_replace.

24 Дан массив ['a', 'b', 'c', 'd', 'e']. Поменяйте элемент с ключом **0** на '!', а элемент с ключом **3** - на '!!'.

Работа с сортировку

Для решения задач данного блока вам понадобятся следующие функции: функции для сортировки,

25 Дан массив **'3'=>'a', '1'=>'c', '2'=>'e', '4'=>'b'**. Попробуйте на нем различные типы сортировок.

Работа с array_rand

Для решения задач данного блока вам понадобятся следующие функции: array_rand.

26 Дан массив с элементами **'a'=>1, 'b'=>2, 'c'=>3**. Выведите на экран случайный **ключ** из данного массива.

27 Дан массив с элементами **'a'=>1, 'b'=>2, 'c'=>3**. Выведите на экран случайный **элемент** данного массива.

Работа с shuffle

Для решения задач данного блока вам понадобятся следующие функции: shuffle.

28 Дан массив **$arr**. Перемешайте его элементы в случайном порядке.

29 Заполните массив числами от **1** до **25** с помощью range, а затем перемешайте его элементы в случайном порядке.

30 Создайте массив, заполненный буквами от **'a'** до **'z'** так, чтобы буквы шли в случайном порядке и не повторялись.

31 Сделайте строку длиной **6** символов, состоящую из маленьких английских букв, расположенных в случайном порядке. Буквы не должны повторяться.

Работа с array_unique

Для решения задач данного блока вам понадобятся следующие функции: array_unique.

32 Дан массив с элементами **'a', 'b', 'c', 'b', 'a'**. Удалите из него повторяющиеся элементы.

Работа с array_shift, array_pop, array_unshift, array_push

Для решения задач данного блока вам понадобятся следующие функции: array_shift, array_pop,array_unshift, array_push.

33 Дан массив с элементами **1, 2, 3, 4, 5**. Выведите на экран его первый и последний элемент, причем так, чтобы в исходном массиве они исчезли.

34 Дан массив с элементами **1, 2, 3, 4, 5**. Добавьте ему в начало элемент 0, а в конец - элемент 6.

35 Дан массив с элементами **1, 2, 3, 4, 5, 6, 7, 8**. С помощью цикла и функций **array_shift** и **array_pop**выведите на экран его элементы в следующем порядке: **18273645**. Показать решение.

Работа с array_pad, array_fill, array_fill_keys, array_chunk

Для решения задач данного блока вам понадобятся следующие функции: array_pad, array_fill, array_fill_keys,array_chunk.

36 Дан массив с элементами **'a', 'b', 'c'**. Сделайте из него массив с элементами **'a', 'b', 'c', '-', '-', '-'**.

37 Заполните массив 10-ю буквами **'x'**.

38 Создайте массив, заполненный целыми числами от **1** до **20**. С помощью функции array_chunkразбейте этот массив на **5** подмассивов (**[1, 2, 3, 4]; [5, 6, 7, 8]** и т.д.).

Работа с array_count_values

Для решения задач данного блока вам понадобятся следующие функции: array_count_values.

39 Дан массив с элементами **'a', 'b', 'c', 'b', 'a'**. Подсчитайте сколько раз встречается каждая из букв.

Работа с array_map

Для решения задач данного блока вам понадобятся следующие функции: array_map.

40 Дан массив с элементами **1, 2, 3, 4, 5**. Создайте новый массив, в котором будут лежать квадратные корни данных элементов.

41 Дан массив с элементами **'php'**, **'html'**. Создайте новый массив, в котором из элементов будут удалены теги.

42 Дан массив с элементами **' a ', ' b ', ' c '**. Создайте новый массив, в котором будут данные элементы без концевых пробелов.

Работа с array_intersect, array_diff

Для решения задач данного блока вам понадобятся следующие функции: array_intersect, array_diff.

43 Дан массив с элементами **1, 2, 3, 4, 5** и массив с элементами **3, 4, 5, 6, 7**. Запишите в новый массив элементы, которые есть и в том, и в другом массиве.

44 Дан массив с элементами **1, 2, 3, 4, 5** и массив с элементами **3, 4, 5, 6, 7**. Запишите в новый массив элементы, которые не присутствуют в обоих массивах одновременно.

Задачи

45 Дана строка **'1234567890'**. Найдите сумму цифр из этой строки не используя цикл.

46 Создайте массив **['a'=>1, 'b'=2... 'z'=>26]** не используя цикл. Показать подсказку.

47 Создайте массив вида **[[1, 2, 3], [4, 5, 6], [7, 8, 9]]** не используя цикл. Показать подсказку.

48 Дан массив с элементами **1, 2, 4, 5, 5**. Найдите второй по величине элемент. В нашем случае это будет **4**.