**Конспект «Массивы и функции в PHP»**

**Функции**

**Функция** — это фрагмент кода, который можно написать один раз, а затем использовать, когда потребуется.

Прежде чем использовать функцию, её *объявляют* с помощью ключевого слова function. После него указывают имя функции. Оно должно начинаться с латинской буквы или знака подчёркивания.

После имени функции пишут круглые скобки, а затем — тело функции в фигурных скобках. В теле функции находится тот код, который функция будет выполнять.

function my\_func() {

keks\_log('Я функция my\_func');

}

Когда функцию объявляют, PHP запоминает её, но при этом код внутри фигурных скобок не выполняется. Чтобы его выполнить, функцию надо *вызвать*.

Чтобы вызвать функцию, пишем её имя и круглые скобки:

my\_func(); // Выведет: "Я функция my\_func"

Функцию можно вызывать любое число раз.

Переменные, объявленные внутри функции, доступны только внутри этой функции. А внешние переменные, наоборот, внутри функции не видны. Это называется областью видимости. Вы можете узнать о ней больше в [документации](https://www.php.net/manual/ru/language.variables.scope.php).

Мы можем вызывать одну функцию внутри другой.

**Аргументы функции**

Аргументы позволяют передавать функции разные данные и влиять на результат её работы. Аргументы указывают при объявлении функции в круглых скобках. Их может быть сколько угодно. Если аргументов несколько, они указываются через запятую.

Аргументы похожи на переменные. Они тоже хранят значения, и их имена должны начинаться со знака доллара. Аргументы можно использовать только в теле функции:

function my\_func($number\_1, $number\_2) {

keks\_log($number\_1 - $number\_2);

}

Значения аргументов указывают в круглых скобках при вызове функции. Их перечисляют через запятую **в том же порядке**, что и аргументы при объявлении функции.

my\_func(3, 1); // Выведет: "2"

my\_func(1, 3); // Выведет: "-2"

**Функция возвращает значение**

Функции могут не только принимать данные, но и **возвращать** их. Чтобы указать, какое значение должна вернуть функция, используют ключевое слово return:

function my\_func($number\_1, $number\_2) {

return $number\_1 - $number\_2;

}

Когда функцию вызовут, на её место в коде подставится то значение, которое она вернёт.

<!-- Вызываем функцию в шаблоне -->

<p><?= my\_func(3, 1) ?></p>

<!-- Результат -->

<p>2</p>

Вернув значение, функция прекратит работу. Последующий код **в теле** функции PHP проигнорирует.

function my\_func($number\_1, $number\_2) {

return $number\_1 - $number\_2;

// Код на следующей строке никогда не будет исполнен

keks\_log('Есть ли жизнь после return?');

}

Функция может ничего не возвращать или вернуть **одно** любое значение: число, строку, массив и так далее.

**Встроенные функции**

**Встроенные функции** — это функции, которые написали создатели PHP. Это часть языка, поэтому их не нужно объявлять, прежде чем использовать. В PHP существуют тысячи готовых функций. Нет нужды заучивать их наизусть, ведь всегда можно заглянуть в [документацию](https://www.php.net/manual/ru/funcref.php).

**Колбэком** называют функцию, которую передают в качестве аргумента другой функции. Колбэки не являются встроенными функциями, их пишут сами разработчики.

uasort

Встроенная функция uasort сортирует массив. Эта функция принимает два аргумента: массив, который нужно отсортировать, и имя функции-колбэка.

uasort($array, 'callback');

Функция-колбэк, которую мы передаём uasort, должна принимать два аргумента и возвращать число: положительное число, если первый аргумент больше, отрицательное число, если больше второй аргумент, и ноль, если аргументы равны.

Во время вызова uasort передаёт в колбэк по двое элементы массива, узнаёт, какой из них больше, и сортирует массив по возрастанию: от меньшего к большему.

array\_slice

Функция array\_slice принимает массив, копирует его часть и возвращает её в виде нового массива. Массив, который передали функции, при этом не изменяется.

Функция принимает четыре аргумента:

* исходный массив,
* порядковый номер элемента, с которого начнётся копирование,
* количество элементов, которые нужно скопировать,
* булево значение — нужно ли сохранять ключи исходного массива.

$new\_array = array\_slice($array, $first\_element, $length, $preserve\_keys);

array\_filter

В PHP есть встроенная функция фильтрации — array\_filter. Она принимает два аргумента: исходный массив и функцию-колбэк — и возвращает новый массив. Исходный массив при этом не изменяется.

$filtered\_array = array\_filter($array, 'callback');

Согласно [документации](https://www.php.net/manual/ru/function.array-filter.php), колбэк должен принимать элемент исходного массива и возвращать булево значение: true, если этот элемент нужно включить в новый массив, и false, если не нужно.

array\_rand

Для получения случайных элементов из массива, используем функцию array\_rand. Она принимает исходный массив и количество случайных элементов, которые мы хотим получить.

$random = array\_rand($array, $number\_of\_elements);

Если второй аргумент — единица, функция вернёт один случайный ключ. Если второй аргумент больше единицы, функция вернёт массив со случайными ключами.

Функция array\_rand выбирает случайные элементы, но возвращает их в том порядке, в каком они идут в исходном массиве.

shuffle

Встроенная функция shuffle перемешивает массив. Функция принимает массив и выстраивает его элементы в случайном порядке. Будьте осторожны, функция shuffle изменяет переданный ей массив и не сохраняет ключи. С ассоциативными массивами её лучше не использовать.

shuffle($array);

**Объединение массивов**

В PHP массивы можно объединить с помощью знака плюс:

$big\_array = $array\_1 + $array\_2;

В результате получится один массив, который включает элементы как из первого, так и из второго массива. При этом порядок элементов и их ключи сохранятся.

Если у нескольких элементов окажутся одинаковые ключи, то значение возьмётся из того массива, который идёт раньше. Перед объединением убедитесь, что ключи в массивах не совпадают или что одинаковым ключам соответствуют одинаковые значения.