Функции обработки массивов

Массивы идеально подходят для хранения, изменения и работы с наборами переменных.

Поддерживаются одно- и многомерные массивы, как созданные пользователем, так и возвращённые в качестве результата какой-либо функцией. Используя специальные функции можно записать в массив результаты запроса к базам данных. Также существуют функции, возвращающие массивы в качестве результата.

Массив в PHP - это упорядоченное отображение, которое устанавливает соответствие между *значением* и *ключом*. Этот тип оптимизирован в нескольких направлениях, поэтому вы можете использовать его как собственно массив, список (вектор), хеш-таблицу (являющуюся реализацией карты), словарь, коллекцию, стек, очередь и, возможно, что-то ещё. Так как значением массива может быть другой массив PHP, можно также создавать деревья и многомерные массивы.

Массив (тип array) может быть создан языковой конструкцией array(). В качестве параметров она принимает любое количество разделённых запятыми пар key => value (ключ => значение).

```
array(
    key  => value,
    key2 => value2,
    key3 => value3,
    ...
)
```

Запятая после последнего элемента массива необязательна и может быть опущена. Обычно это делается для однострочных массивов, то есть array(1, 2) предпочтительней array(1, 2,). Для многострочных массивов с другой стороны обычно используется завершающая запятая, так как позволяет легче добавлять новые элементы в конец массива.

Так же существует короткий синтаксис массива, который заменяет array() на [].

Пример: Простой массив

```
<?php
$array = array(
    "foo" => "bar",
    "bar" => "foo",
);

Использование синтаксиса короткого массива
$array = [
    "foo" => "bar",
    "bar" => "foo",
];
?>
```

Где key может быть либо типа int, либо типа string. value может быть любого типа.

Дополнительно с ключом key будут сделаны следующие преобразования:

- Строки (string), содержащие целое число (int) (исключая случаи, когда число предваряется знаком +) будут преобразованы к типу int. Например, ключ со значением "8" будет в действительности сохранён со значением 8. С другой

стороны, значение "08" не будет преобразовано, так как оно не является корректным десятичным целым.

- Числа с плавающей точкой (float) также будут преобразованы к типу int, то есть дробная часть будет отброшена. Например, ключ со значением 8.7 будет в действительности сохранён со значением 8.
- Тип bool также преобразовываются к типу int. Например, ключ со значением true будет сохранён со значением 1 и ключ со значением false будет сохранён со значением 0.
- Тип null будет преобразован к пустой строке. Например, ключ со значением null будет в действительности сохранён со значением "".
- Массивы (array) и объекты (object) *не могут* использоваться в качестве ключей. При подобном использовании будет генерироваться предупреждение: Недопустимый тип смещения (Illegal offset type).

Если несколько элементов в объявлении массива используют одинаковый ключ, то только последний будет использоваться, а все другие будут перезаписаны.

Пример: Пример преобразования типов и перезаписи элементов

```
<?php
$array = array(
    1    => "a",
    "1"  => "b",
    1.5  => "c",
    true => "d",
);
var_dump($array);
?>
```

Где var_dump - это дамп информации о переменной.

Так как все ключи в вышеприведённом примере преобразуются к 1, значение будет перезаписано на каждый новый элемент и останется только последнее присвоенное значение "d".

Массивы в PHP могут содержать ключи типов int и string одновременно, так как PHP не делает различия между индексированными и ассоциативными массивами.

Пример #3 Смешанные ключи типов int и string

```
<?php
$array = array(
    "foo" => "bar",
    "bar" => "foo",
    100   => -100,
    -100  => 100,
);
var_dump($array);
?>
```

Параметр key является необязательным. Если он не указан, PHP будет использовать предыдущее наибольшее значение ключа типа int, увеличенное на 1.

Пример: Индексированные массивы без ключа

```
<?php
$array = array("foo", "bar", "hallo", "world");
var_dump($array);
?>
```

Возможно указать ключ только для некоторых элементов и пропустить для других:

Пример: Ключи для некоторых элементов

```
<?php
$array = array(
        "a",
        "b",
    6 => "c",
        "d",
);
var_dump($array);
?>
```

Как выше видно последнее значение "d" было присвоено ключу 7. Это произошло потому, что самое большое значение ключа целого типа перед этим было 6.

Пример: Расширенный пример преобразования типов и перезаписи элементов

```
<?php
$array = array(
    1    => 'a',
    '1'  => 'b', // значение "b" перезапишет значение "a"
    1.5  => 'c', // значение "c" перезапишет значение "b"
    -1 => 'd',
    '01'  => 'e', // поскольку это не целочисленная строка, она НЕ перезапишет клю
ч для 1
    '1.5' => 'f', // поскольку это не целочисленная строка, она НЕ перезапишет клю
ч для 1
    true => 'g', // значение "g" перезапишет значение "c"
    false => 'h',
    '' => 'i',
    null => 'j', // значение "j" перезапишет значение "i"
    'k', // значение "k" присваивается ключу 2. Потому что самый большой целочис
ленный ключ до этого был 1
    2 => 'l', // значение "l" перезапишет значение "k"
);

var_dump($array);
?>
```

Этот пример включает все вариации преобразования ключей и перезаписи элементов.

Доступ к элементам массива может быть осуществлён с помощью синтаксиса array[key].

Пример: Доступ к элементам массива

```
<?php
$array = array(
    "foo" => "bar",
    42    => 24,
    "multi" => array(
        "dimensional" => array(
            "array" => "foo"
```

```
        )
    )
);

var_dump($array["foo"]);
var_dump($array[42]);
var_dump($array["multi"]["dimensional"]["array"]);
?>
```

До PHP 8.0.0 квадратные и фигурные скобки могли использоваться взаимозаменяемо для доступа к элементам массива (например, в примере выше $array[42] и $array{42} делали то же самое). Синтаксис фигурных скобок устарел в PHP 7.4.0 и больше не поддерживается в PHP 8.0.0.

Пример: Разыменование массива

```
<?php
function getArray() {
    return array(1, 2, 3);
}

$secondElement = getArray()[1];
?>
```

Попытка доступа к неопределённому ключу в массиве - это то же самое, что и попытка доступа к любой другой неопределённой переменной: будет сгенерирована ошибка уровня E_WARNING (ошибка уровня E_NOTICE до PHP 8.0.0), и результат будет null.

Массив, разыменовывающий скалярное значение, которое не является строкой (string), отдаст null. До PHP 7.4.0 не выдаётся сообщение об ошибке. Начиная с PHP 7.4.0, выдаётся ошибка E_NOTICE; с PHP 8.0.0 выдаётся ошибка E_WARNING.

Существующий массив может быть изменён путём явной установкой значений в нем.

Это выполняется присвоением значений массиву (array) с указанием в скобках ключа. Кроме того, ключ можно опустить, в результате получится пустая пара скобок ([]).

```
$arr[key] = value;
$arr[] = value;
// key может быть int или string
// value может быть любым значением любого типа
```

Если массив *$arr* ещё не существует или для него задано значение null или false, он будет создан. Таким образом, это ещё один способ определить массив array. Однако такой способ применять не рекомендуется, так как если переменная *$arr* уже содержит некоторое значение (например, значение типа string из переменной запроса), то это значение останется на месте и [] может на самом деле означать доступ к символу в строке. Лучше инициализировать переменную путём явного присваивания значения.

Для изменения определённого значения просто присвойте новое значение элементу, используя его ключ. Если вы хотите удалить пару ключ/значение, вам необходимо использовать функцию unset(), так как она и используется для удаления перечисленных переменных.

```
<?php
$arr = array(5 => 1, 12 => 2);
```

```
$arr[] = 56;    // В этом месте скрипта это
                // то же самое, что и $arr[13] = 56;

$arr["x"] = 42; // Это добавляет к массиву новый
                // элемент с ключом "x"

unset($arr[5]); // Это удаляет элемент из массива

unset($arr);    // Это удаляет массив полностью
?>
```

Как уже говорилось выше, если ключ не был указан, то будет взят максимальный из существующих целочисленных (int) индексов, и новым ключом будет это максимальное значение (в крайнем случае 0) плюс 1. Если целочисленных (int) индексов ещё нет, то ключом будет 0 (ноль).

Функции для работы с массивами

- array_change_key_case — Меняет регистр всех ключей в массиве
- array_chunk — Разбивает массив на части
- array_column — Возвращает массив из значений одного столбца входного массива
- array_combine — Создаёт новый массив, используя один массив в качестве ключей, а другой для его значений
- array_count_values — Подсчитывает количество всех значений массива
- array_diff_assoc — Вычисляет расхождение массивов с дополнительной проверкой индекса
- array_diff_key — Вычисляет расхождение массивов, сравнивая ключи
- array_diff_uassoc — Вычисляет расхождение массивов с дополнительной проверкой индекса, осуществляемой при помощи callback-функции
- array_diff_ukey — Вычисляет расхождение массивов, используя callback-функцию для сравнения ключей
- array_diff — Вычислить расхождение массивов
- array_fill_keys — Создаёт массив и заполняет его значениями с определёнными ключами
- array_fill — Заполняет массив значениями
- array_filter — Фильтрует элементы массива с помощью callback-функции
- array_flip — Меняет местами ключи с их значениями в массиве
- array_intersect_assoc — Вычисляет схождение массивов с дополнительной проверкой индекса
- array_intersect_key — Вычислить пересечение массивов, сравнивая ключи
- array_intersect_uassoc — Вычисляет схождение массивов с дополнительной проверкой индекса, осуществляемой при помощи callback-функции
- array_intersect_ukey — Вычисляет схождение массивов, используя callback-функцию для сравнения ключей
- array_intersect — Вычисляет схождение массивов
- array_is_list — Проверяет, является ли данный array списком
- array_key_exists — Проверяет, присутствует ли в массиве указанный ключ или индекс
- array_key_first — Получает первый ключ массива

- array_key_last — Получает последний ключ массива
- array_keys — Возвращает все или некоторое подмножество ключей массива
- array_map — Применяет callback-функцию ко всем элементам указанных массивов
- array_merge_recursive — Рекурсивное слияние одного или более массивов
- array_merge — Сливает один или большее количество массивов
- array_multisort — Сортирует несколько массивов или многомерные массивы
- array_pad — Дополнить массив определённым значением до указанной длины
- array_pop — Извлекает последний элемент массива
- array_product — Вычислить произведение значений массива
- array_push — Добавляет один или несколько элементов в конец массива
- array_rand — Выбирает один или несколько случайных ключей из массива
- array_reduce — Итеративно уменьшает массив к единственному значению, используя callback-функцию
- array_replace_recursive — Рекурсивно заменяет элементы первого массива элементами переданных массивов
- array_replace — Заменяет элементы массива элементами других переданных массивов
- array_reverse — Возвращает массив с элементами в обратном порядке
- array_search — Осуществляет поиск данного значения в массиве и возвращает ключ первого найденного элемента в случае успешного выполнения
- array_shift — Извлекает первый элемент массива
- array_slice — Выбирает срез массива
- array_splice — Удаляет часть массива и заменяет её чем-нибудь ещё
- array_sum — Вычисляет сумму значений массива
- array_udiff_assoc — Вычисляет расхождение в массивах с дополнительной проверкой индексов, используя для сравнения значений callback-функцию
- array_udiff_uassoc — Вычисляет расхождение в массивах с дополнительной проверкой индексов, используя для сравнения значений и индексов callback-функцию
- array_udiff — Вычисляет расхождение массивов, используя для сравнения callback-функцию
- array_uintersect_assoc — Вычисляет пересечение массивов с дополнительной проверкой индексов, используя для сравнения значений callback-функцию
- array_uintersect_uassoc — Вычисляет пересечение массивов с дополнительной проверкой индекса, используя для сравнения индексов и значений индивидуальные callback-функции
- array_uintersect — Вычисляет пересечение массивов, используя для сравнения значений callback-функцию
- array_unique — Убирает повторяющиеся значения из массива
- array_unshift — Добавляет один или несколько элементов в начало массива
- array_values — Выбирает все значения массива
- array_walk_recursive — Рекурсивно применяет пользовательскую функцию к каждому элементу массива
- array_walk — Применяет заданную пользователем функцию к каждому элементу массива
- array — Создаёт массив
- arsort — Сортирует массив в порядке убывания и поддерживает ассоциацию индексов

- asort — Сортирует массив в порядке возрастания и поддерживает ассоциацию индексов
- compact — Создаёт массив, содержащий названия переменных и их значения
- count — Подсчитывает количество элементов массива или Countable объекте
- current — Возвращает текущий элемент массива
- each — Возвращает текущую пару ключ/значение из массива и смещает его указатель
- end — Устанавливает внутренний указатель массива на его последний элемент
- extract — Импортирует переменные из массива в текущую таблицу символов
- in_array — Проверяет, присутствует ли в массиве значение
- key_exists — Псевдоним array_key_exists
- key — Выбирает ключ из массива
- krsort — Сортирует массив по ключу в порядке убывания
- ksort — Сортирует массив по ключу в порядке возрастания
- list — Присваивает переменным из списка значения подобно массиву
- natcasesort — Сортирует массив, используя алгоритм "natural order" без учёта регистра символов
- natsort — Сортирует массив, используя алгоритм "natural order"
- next — Перемещает указатель массива вперёд на один элемент
- pos — Псевдоним current
- prev — Передвигает внутренний указатель массива на одну позицию назад
- range — Создаёт массив, содержащий диапазон элементов
- reset — Устанавливает внутренний указатель массива на его первый элемент
- rsort — Сортирует массив в порядке убывания
- shuffle — Перемешивает массив
- sizeof — Псевдоним count
- sort — Сортирует массив по возрастанию
- uasort — Сортирует массив, используя пользовательскую функцию для сравнения элементов с сохранением ключей
- uksort — Сортирует массив по ключам, используя пользовательскую функцию для сравнения ключей
- usort — Сортирует массив по значениям используя пользовательскую функцию для сравнения элементов

Контрольные вопросы:

1. Понятие массива в PHP и общий вид записи;
2. Преобразования, доступные с ключом key;
3. Пример использования дампа информации о переменной;
4. Пример использования доступа к элементу массива;
5. Доступные функции для работы с массивами.