Основы PHP

Функции для работы с массивами

Функции для работы с массивами

list()

Предположим, у нас есть массив, состоящий из трех элементов:

```
$names[0] = "Александр";
$names[1] = "Николай";
$names[2] = "Яков";
```

Допустим, в какой-то момент нам нужно передать значения всех трех элементов массива, соответственно трем переменным: $alex, $nick, $yakov. Это можно сделать так:

```
$alex = $names[0];
$nick = $names[1];
$yakov = $names[2];
```

Функции для работы с массивами

list()

Если массив большой, то такой способ присвоения элементов массива переменным не очень удобен.
Есть более рациональный подход - использование функции list():

```
list ($alex, $nick, $yakov) = $names;
```

Если нам нужны только "Николай" и "Яков", то мы можем сделать так:

```
list (, $nick, $yakov) = $names;
```

Функции для работы с массивами

array()

Функция Array() используется специально для создания массивов. При этом она позволяет создавать пустые массивы. Вот методы использования функции Array():

```
<?php
    $arr = array(); // Создает пустой массив
    $arr2 = array("Иванов","Петров","Сидоров"); // Создает список с тремя элементами.  Индексы начинаются с нуля
    $arr3 = array("Иванов"=>"Иван", "Петров"=>"Петр", "Сидоров"=>"Сидор"); // Создает ассоциативный массив с тремя элементами
    // Создает многомерный ассоциативный массив:
    $arr4 = array("name"=>"Иванов", "age"=>"24", "email"=>"ivanov@mail.ru");
    $arr4 = array("name"=>"Петров", "age"=>"34", "email"=>"petrov@mail.ru");
    $arr4 = array("name"=>"Сидоров", "age"=>"47", "email"=>"sidorov@mail.ru");
?>
```

Операции над массивами

Сортировка массива по значениям с помощью функций asort() и arsort()

Функция asort() сортирует массив, указанный в ее параметре, так, чтобы его значения шли в алфавитном (если это строки) или в возрастающем (для чисел) порядке.
Функция arsort() выполняет то же самое, за одним исключением: она упорядочивает массив не по возрастанию, а по убыванию.

Сортировка по ключам с помощью функций ksort() и krsort()

Функция ksort() практически идентична функции asort(), с тем различием, что сортировка осуществляется не по значениями, а по ключам (в порядке возрастания).
Функция для сортировки по ключам в обратном порядке называется krsort() и применяется точно в таком же контексте, что и ksort().

Операции над массивами

Сортировка по ключам при помощи функции uksort()

Довольно часто нам приходится сортировать что-то по более сложному критерию, чем просто по алфавиту. В этом случае нам стоит воспользоваться функцией uksort(), написав предварительно функцию сравнения с двумя параметрами, как того требует uksort().
Функция uasort() очень похожа на uksort(), с той разницей, что сменной (пользовательской) функции сортировки "подсовываются" не ключи, а очередные значения из массива.

Функция array_reverse()

Функция array_reverse() возвращает массив, элементы которого следуют в обратном порядке относительно массива, переданного в параметре.

Операции над массивами

Сортировка списка при помощи функций sort() и rsort()

Эти две функции предназначены в первую очередь для сортировки списков. Функция sort() сортирует список (разумеется, по значениям) в порядке возрастания, а rsort() — в порядке убывания.

Функция shuffle()

Функция shuffle() "перемешивает" список, переданный ей первым параметром, так, чтобы его значения распределялись случайным образом. Обратите внимание, что, во-первых, изменяется сам массив, а во вторых, ассоциативные массивы воспринимаются как списки.
Выполнив shuffle() несколько раз, вы можете обнаружить, что от запуска к запуску очередность следования чисел не изменяется. Это свойство обусловлено тем, что функция shuffle() использует стандартный генератор случайных чисел, который перед работой необходимо инициализировать при помощи вызова srand().

Операции с ключами и значениями массива

array_flip(array $arr)

Функция array_flip() "пробегает" по массиву и меняет местами его ключи и значения. Исходный массив $arr не изменяется, а результирующий массив просто возвращается.
Конечно, если в массиве присутствовали несколько элементов с одинаковыми значениями, учитываться будет только последний из них:

```
$A = array("a"=>"aaa", "b"=>"aaa", "c"=>"ccc");
$A = array_flip($A); // теперь $A===array("aaa"=>"b", "ccc"=>"c");
```

array_keys(array $arr [,mixed $SearchVal])

Функция array_keys() возвращает список, содержащий все ключи массива $arr. Если задан необязательный параметр $SearchVal, то она вернет только те ключи, которым соответствуют значения $SearchVal.
Фактически, эта функция с заданным вторым параметром является обратной по отношению к оператору [] — извлечению значения по его ключу.

Операции с ключами и значениями массива

array_values(array $arr)

Функция array_values() возвращает список всех значений в ассоциативном массиве $arr. Очевидно, такое действие бесполезно для списков, но иногда оправдано для хэшей.

in_array(mixed $val, array $arr)

Функция in_array() возвращает true, если элемент со значением $val присутствует в массиве $arr.
Впрочем, если вам часто приходится проделывать эту операцию, подумайте: не лучше ли будет воспользоваться ассоциативным массивом и хранить данные в его ключах, а не в значениях? На этом вы можете сильно выиграть в быстродействии.

Операции с ключами и значениями массива

array_count_values(list $List)

Функция array_count_values() подсчитывает, сколько раз каждое значение встречается в списке $List, и возвращает ассоциативный массив с ключами — элементами списка и значениями — количеством повторов этих элементов. Иными словами, функция array_count_values() подсчитывает частоту появления значений в списке $List.

```
$List = array(1, "hello", 1, "world", "hello");
array_count_values($array); // возвращает array(1=>2, "hello"=>2, "world"=>1)
```

Слияние массивов

Слияние (конкатенация) массивов - это операция создания массива, состоящего из элементов нескольких других массивов.
Слияние массивов - это очень опасная операция, поскольку результат слияния подчиняется своей логике, забыв о которой можно потерять данные.
Слияние массивов реализуется при помощи оператора "+" или с помощью функции array_merge().
Слияние списков может осуществляться только с помощью функции array_merge().

Слияние массивов

Предположим, мы имеем два массива:

```
$A = array("1"=>"первый", "2"=>"второй");
$B = array(«3"=>"третий", "4"=>"четвертый");
```

Теперь сольем данные два массива в один массив $C:

```
$C = $A + $B;
```

Оператор "+" для массивов не коммутативен. Это означает, что $A + $B не равно $B + $A.

В результате рассмотренного примера мы получим массив $C следующего вида:

"1"=>"первый", "2"=>"второй", "3"=>"третий", "4"=>"четвертый"

А в результате $B + $A мы получим такой массив:

"3"=>"третий", "4"=>"четвертый", "1"=>"первый", "2"=>"второй"

Слияние массивов

При слиянии списков такой метод не работает. Поясним данный факт на примере:
Предположим, у нас есть два массива:

```
$A = array(10,11,12);
$B = array(13,14,15);
```

В результате слияния списков $A и $B ($A + $B) мы получим: 10,11,12.
А это совсем не тот результат, который мы хотели получить...
Связано это с тем, что при слиянии списков с одинаковыми индексами в результирующем массиве остается элемент первого массива, причем на том же месте.
В таком случае нам необходимо использовать функцию array_merge().

Слияние массивов

array_merge()

Функция array_merge() призвана устранить все недостатки, присущие оператору "+" для слияния массивов.
А именно, она сливает массивы, перечисленные в ее аргументах, в один большой массив и возвращает результат.
Если в массивах встречаются одинаковые ключи, в результат помещается пара "ключ=>значение" из того массива, который расположен правее в списке аргументов.
Однако это не затрагивает числовые ключи: элементы с такими ключами помещаются в конец результирующего массива в любом случае.

Слияние массивов

array_merge()

Таким образом, с помощью array_merge() мы можем избавиться от всех недостатков оператора "+" для массивов. Вот пример, сливающий два списка в один:

```
$L1 = array(100,200,300);
$L2 = array(400,500,600);
$L = array_merge($L1,$L2); // теперь $L===array(100,200,300,400,500,600);
```

Всегда используйте эту функцию, если вам нужно работать именно со списками, а не с обычными ассоциативными массивами.

Получение части массива

array_slice(array $Arr, int $offset [, int $len])

Для получения части массива можно использовать функцию array_slice(). Эта функция возвращает часть ассоциативного массива, начиная с пары "ключ=>значения" со смещением (номером) $offset от начала и длиной $len (если последний параметр не задан - до конца массива). Параметры $offset и $len задаются по точно таким же правилам, как и аналогичные параметры в функции substr(). А именно, они могут быть отрицательными (в этом случае отсчет осуществляется от конца массива), и т. д.

Вот несколько примеров:

```
$input = array ("a", "b", "c", "d", "e");
$output = array_slice ($input, 2); // "c", "d", "e"
$output = array_slice ($input, 2, -1); // "c", "d"
$output = array_slice ($input, -2, 1); // "d"
$output = array_slice ($input, 0, 3); // "a", "b", "c"
```

Вставка и удаление элементов массивов

Оператор [] (пустые квадратные скобки) добавляет элемент в конец массива, присваивая ему числовой ключ, а оператор Unset() вместе с извлечением по ключу удаляет нужный элемент.
Язык PHP поддерживает и многие другие функции, которые иногда бывает удобно использовать.

```
array_push(alist &$Arr, mixed $var1 [, mixed $var2, …])
```

Эта функция добавляет к списку $Arr элементы $var1, $var2 и т. д. Она присваивает им числовые индексы — точно так же, как это происходит для стандартных []. Если вам нужно добавить всего один элемент, наверное, проще и будет воспользоваться этим оператором:

```
array_push($Arr,1000); // вызываем функцию…
$Arr[] = 100; // то же самое, но короче
```

Обратите внимание, что функция array_push() воспринимает массив как стек и добавляет элементы всегда в его конец. Она возвращает новое число элементов в

Вставка и удаление элементов массивов

array_pop(list &$Arr)

Функция array_pop(), является противоположностью array_push(), снимает элемент с "вершины" стека (то есть берет последний элемент списка) и возвращает его, удалив после этого его из $Arr.
С помощью этой функции мы можем строить конструкции, напоминающие стек.
Если список $Arr был пуст, функция возвращает пустую строку.

Вставка и удаление элементов массивов

array_splice(array &$Arr, int $offset [, int $len] [, int $Repl])

Функция array_splice, также как и array_slice(), возвращает подмассив $Arr, начиная с индекса $offset максимальной длины $len, но, вместе с тем, она делает и другое полезное действие.

А именно, она заменяет только что указанные элементы на то, что находится в массиве $Repl (или просто удаляет, если $Repl не указан).

```
<?php
    $input = array("red", "green", "blue", "yellow");
    array_splice($input,2); // Теперь $input===array("red", "green")
    array_splice($input,1,-1); // Теперь $input===array("red", "yellow")
    array_splice($input, -1, 1, array("black", "maroon")); // Теперь $input===array("red", "green", "blue", "black", "maroon")
    array_splice($input, 1, count($input), "orange");// Теперь $input===array("red", "orange")
```

Вставка и удаление элементов массивов

compact(mixed $vn1 [, mixed $vn2, …])

Функция compact() упаковывает в массив переменные из текущего контекста (глобального или контекста функции), заданные своими именами в $vn1, $vn2 и т. д. При этом в массиве образуются пары с ключами, равными содержимому $vnN, и значениями соответствующих переменных. Вот пример использования этой функции:

```
$a = "Test string";
$b = "Some text";
$A = compact("a","b"); // теперь $A===array("a"=>"Test string", "b"=>"Some text")
```

Функция extract() производит действия, прямо противоположные compact().
А именно, она получает в параметрах массив $Arr и превращает каждую его пару "ключ=>значение" в переменную текущего контекста.

Создание списка – диапазона чисел

range(int $low, int $high)

Эта функция очень простая. Она создает список, заполненный целыми числами от $low до $high включительно.

Удаление массива и его элементов

unset()

Если вы хотите удалить массив целиком, воспользуйтесь функцией unset(). Если вы хотите удалить пару ключ/значение, вы также можете использовать функцию unset().

```
<?php
    $arr = array(5=>1, 12=>2);
    $arr[] = 56; // В этом месте скрипта это эквивалентно $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
?>
```