Тема 3.7
«Массивы в PHP и функции по работе с ними»

Вступление

Второе после строк, на работу с чем максимально ориентирован PHP, -- это массивы.

Несколько фактов о массивах в PHP:

1) Они бывают ТОЛЬКО динамическими (т.е. в любой момент можно добавить элемент в массив или удалить его из массива).

2) Они могут быть многомерными, причём мерность не ограничена.

3) Они могут одновременно содержать данные любых поддерживаемых в PHP типов.

Ещё немного фактов о массивах

Массив в PHP – это упорядоченное отображение, которое устанавливает соответствие между значением и ключом.

Этот тип данных оптимизирован в нескольких направлениях, поэтому вы можете использовать его как:

- собственно массив;
- список (вектор);
- хэш-таблицу;
- словарь;
- коллекцию;
- стек;
- очередь.

Поскольку значением элемента массива может быть другой массив, вы можете легко эмулировать деревья.

Инициализация массивов: array()

Массив может быть создан языковой конструкцией array().

В качестве параметров она принимает некоторое количество разделённых запятыми пар

key => value (ключ => значение).

Т.е. синтаксис инициализации массива с помощью конструкции array() таков:

array([key =>] value , ...)

При этом key может быть integer или string, а value может быть значением любого поддерживаемого PHP типа данных.

Пример:

```
$arr = array(“A" => “lalala", 12 => true, 76);
echo $arr[“A"]; // lalala
echo $arr[12]; // 1
echo $arr[13]; // 76
```

Обратите внимание: если при инициализации массива вы не указываете значение ключа того или иного элемента, PHP автоматически берёт самое большое из использованных значений ключа, увеличивает его на 1, и использует в качестве ключа нового элемента. Если до этого в массиве не было числовых ключей, новым ключом будет значение 0.

Инициализация массивов: array()

В PHP нет разницы между **индексными** и **ассоциативными** массивами.

У индексного (индексированного, «обычного») массива ключами являются числа (чаще всего начинающиеся с нуля и идущие подряд, т.е. 0, 1, 2, 3 и т.д.

У ассоциативного массива ключами могут являться строки.

Иногда выделяют и т.н. «смешанные массивы», в которых ключами являются и строки, и числа, но, по сути, -- это лишь форма ассоциативного массива.

Пример создания индексированного массива:

```
$arr = array(“A”, “B”, “C”, 50 => “D”, 100 = “Minsk”);
```

Пример создания ассоциативного массива:

```
$arr = array(“age” => 50, “name” = “Vasya”, “city” => “Minsk”);
```

Если при инициализации массива вы укажете ключ, которому уже присвоено значение, оно будет перезаписано.

Пример:

```
// этот массив
array(5 => 43, 32, 56, "b" => 12);
// эквивалентен этому массиву
array(5 => 43, 6 => 32, 7 => 56, "b" => 12);
```

Инициализация массивов: array()

Если вы будете использовать массив, в котором максимальным в настоящий момент является отрицательный ключ, то следующий созданный ключ будет нулевым (0).

Раньше в такой ситуации новым индексом становился самый большой существующий ключ + 1, так же как и у положительных индексов.

Используя в качестве ключа TRUE вы получите ключ 1 типа integer.

Используя в качестве ключа FALSE вы получите ключ 0 типа integer.

Используя в качестве ключа NULL, вы получите пустую строку.

Использование в качестве ключа пустой строки создаст ключ со значением «пустая строка».

Попытка использовать в качестве ключа массива дробь приводит к тому, что ключом становится целая часть дробного числа.

Ключами массива **НЕ могут** быть объекты и другие массивы.

Инициализация массивов: [ ]

Можно изменять существующий массив, явно устанавливая значения в нём. Это выполняется присвоением значений массиву при указании в квадратных скобках ключа.

Кроме того, вы можете опустить ключ, в этом случае добавьте к имени переменной пустую пару скобок ([]). Поведение PHP в таком случае будет таким же, как и при отсутствии ключа в определении массива через конструкцию array.

Пример:

```
$arr[10] = “A”;
$arr[] = “BBB”;
```

Если массив $arr ещё не существует, он будет создан. Таким образом, это ещё один способ определить массив.

Для изменения определённого значения просто присвойте элементу с его ключом новое значение.

Пример:

```
$arr[10]=“Z”;
```

Добавление и удаление элементов

Если вы хотите удалить пару ключ/значение, нужно использовать функцию unset().

Пример:

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

$arr[] = 56; // эквивалентно $arr[13] = 56;
$arr["x"] = 42; // добавляет новый элемент с ключом "x“

unset($arr[5]); // удаляет элемент с ключом 5 из массива
unset($arr); // удаляет ВЕСЬ массив полностью
```

Инициализация массивов: [ ]

Ещё пример по работе с массивами:

```
// создаём массив
$arr = array(1, 2, 3, 4, 5);
print_r($arr);
// удаляем каждый элемент, но сам массив оставляем нетронутым
foreach ($arr as $i => $value) unset($arr[$i]);
print_r($arr);
// создаём элемент (новым ключом будет 5, а не 0)
$arr[] = 6;
print_r($array);
// переиндексируем массив
$array = array_values($array);
$array[] = 7;
print_r($array);
```

Этот пример выведет следующее:

```
Array (     [0] => 1     [1] => 2     [2] => 3     [3] => 4     [4] => 5 )
Array ( )
Array (     [5] => 6 )
Array (     [0] => 6     [1] => 7 )
```

Замечание по индексированию массивов

Вы **всегда** должны заключать индекс ассоциативного массива в кавычки.

К примеру, пишите $a[‘b'], а не $a[b].

Но почему $a[b] – это неверно (ведь это работает!)?

Причина в том, что этот код содержит неопределённую константу b, а не строку 'b‘, и PHP в будущем может определить константу, которая к несчастью для вашего кода будет иметь то же самое имя.

Неверный пример работает потому, что PHP автоматически преобразует «голую строку» (не заключённую в кавычки, которая не соответствует ни одной из известных конструкций языка) в константу с именем и значением, равным то, что написано в «голой строке».

Например, если константа с именем b не определена, то PHP создаст константу b со значением ‘b’ и использует её.

Но, если в будущем вы измените значение константы b (например, на 12), то вы не сможете обратиться к массиву $a[b], т.к. при инициализации в массиве оказался элемент с ключом ‘b’, а сейчас вы пытаетесь обработать элемент с ключом 12.

Вам повезёт, если такого элемента в массиве нет. Если же там случайно окажется элемент с ключом 12, вы, фактически, «покалечите» массив, начав работать не с тем элементом, который вам нужен.

Преобразование в массив

Для любого из типов: integer, float, string, boolean и resource, преобразование значения в массив, даёт массив с одним элементом (с индексом 0), являющимся значением исходного типа.

Преобразование в массив объекта (object), даёт в качестве элементов массива свойства (переменные-члены) этого объекта. Ключами будут имена переменных-членов.

Преобразование в массив значения NULL, даёт пустой массив.

Обработка всех элементов массива

Поскольку в PHP индексами массива могут быть не только числа (которые, кстати, могут идти не по порядку), но и строки, возникает необходимость в инструменте получения доступа ко всем элементам массива, когда мы не знаем значений их ключей или когда нам неудобно использовать эти значения по одному.

Таким инструментом является специальный оператор цикла foreach, который имеет следующий синтаксис:

```
foreach (array as [$key =>] $value) { statements; }
```

Смысл этого цикла прост: при проходе каждого элемента массива в переменную $key помещается индекс этого элемента, а в переменную $value – его значение. Имена этих двух переменных совершенно произвольны.

Пример:

```
$arr = array(12 => “apple", 96 => “orange", “city” => “Minsk”);
foreach($arr as $k => $v) echo $k.”->”.$v.”; ");
```

Выведет: 12->apple; 96->orange; city->Minsk;

Как видно из синтаксиса, переменная $key необязательна и может быть опущена.

Пример:

```
$arr = array(12 => “apple", 96 => “orange", “city” => “Minsk”);
foreach($arr as $v) echo $v.”; ");
```

Выведет: apple; orange; Minsk;

Многомерные массивы

Для инициализации многомерных массивов используются вложенные конструкции array() или комбинации из нескольких идущих подряд [][][].

Обход многомерных массивов достигается при помощи вложенных циклов.

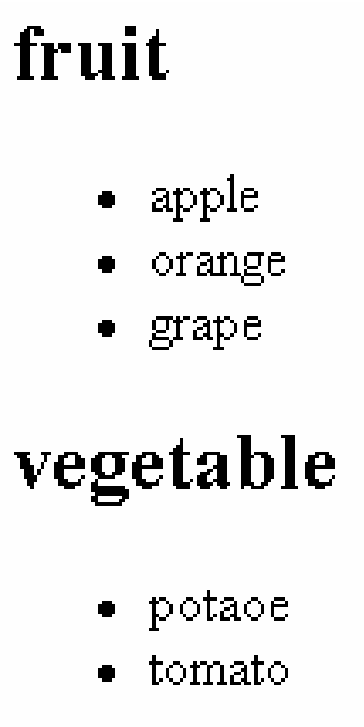
В следующем примере показаны создание и обход многомерного массива:

```
$arr = array(
‘fruit’ => array (‘apple’, ‘orange’, ‘grape’),
‘vegetable’ => array (‘potaoe’, ‘tomato’) );

foreach($arr as $k => $v)
 {
  echo("<h2>$k</h2><ul>");
  foreach($v as $v2) echo("<li>”.$v2.”</li>");
  echo("</ul>");
 }
```

Многомерные массивы

Результат:



Задание для самосовершенствования: написать рекурсивную функцию, строящую такой вывод для массивов произвольной глубины вложенности.

Многомерные массивы

При инициализации многомерных массивов с помощью синтаксиса квадратных скобок можно оставлять пустыми [] ТОЛЬКО скобки ПОСЛЕДНЕГО уровня, т.к. в противном случае увеличение ключей будет происходить по всем уровням одновременно.

Пример 1 (**правильный**):

```
$arr[5][]=“A”;
$arr[5][]=“B”;
print_r($arr);
// получится
Array ( [5] =>
                Array ( [0] => A [1] => B )
        )
```

Пример 1 (**неправильный**):

```
$arr[5][]=“A”;
$arr[][]=“B”;
print_r($arr);
// получится
Array ( [5] =>
                Array ( [0] => A )
          [6] =>
                Array ( [0] => B )
        )
```

Операторы работы с массивами

Вспомним уже рассмотренные ранее в теме «Операторы PHP» операторы работы с массивами:

Оператор	Название	Результат
$a + $b	Объединение массивов	Объединение массивов
$a == $b	Проверка на равенство массивов	TRUE, если массивы содержат одинаковые элементы
$a === $b	Проверка на тождественное равенство массивов	TRUE, если массивы содержат одинаковые элементы в том же порядке
$a != $b $a <> $b	Проверка на неравенство массивов	TRUE, если массивы содержат неодинаковые элементы
$a !== $b	Проверка на тождественное неравенство массивов	TRUE, если массивы содержат неодинаковые элементы или порядок элементов отличается

Операторы работы с массивами

Пример использования операторов работы с массивами:

```
$arr1=array(10 => ‘A’, 20 => ‘B’, 30 => ‘C’);
$arr2=array(10 => ‘aaa’, 20 => ‘bbb’, 90 => ‘ccc’);
$arr3=$arr1+$arr2;
print_r($arr3);
```

Результат:

```
Array ( [10] => A [20] => B [30] => C [90] => ccc )
```

Обратите внимание, что элементы первого массива с ключами, совпадающими с ключами элементов второго массива, не были перезаписаны.

Операторы работы с массивами

```
$arr1=array(10 => 'A', 20 => 'B', 30 => 'C');
$arr2=array(10 => 'A', 20 => 'B', 30 => 'C');
if ($arr1==$arr2) echo "OK";
// выведется OK

$arr1=array(10 => 'A', 20 => 'B', 30 => 'C');
$arr2=array(10 => 'A', 20 => 'B', 900 => 'C');
if ($arr1==$arr2) echo "Yes";
// ничего не выведется

$arr1=array(10 => 'A', 20 => 'B', 30 => 'C');
$arr2=array(10 => 'A', 20 => 'B', 900 => 'C');
if ($arr1===$arr2) echo "Yes";
// ничего не выведется
```

Операторы работы с массивами

На практике операторы работы с массивами почти не применяются в силу их «нечеловекоочевидности» и наличия значительно более простых способов работы с массивами, которые мы сейчас и рассмотрим.

Самым удобным способом работы с массивами является использование функций.

Итак…

Функции сравнения массивов

array array_diff (array array1, array array2 [, array ...]) -- возвращает массив, состоящий из значений массива array1, которые отсутствуют в любом другом массиве, перечисленном в последующих аргументах. Ключи массивов сохраняются.

Пример:

```
$array1 = array ("a" => "green", "red", "blue", "red");
$array2 = array ("b" => "green", "yellow", "red");
$result = array_diff ($array1, $array2);
// $result будет содержать array ("blue");
```

Повторения одного и того же значения в $array1 обрабатываются как одно значение.

Замечание: два элемента считаются одинаковыми если и только если (string) $elem1 === (string) $elem2 (другими словами – когда их строковое представление идентично).

Обратите внимание, что эта функция обрабатывает только одно измерение n-мерного массива. Естественно, вы можете обрабатывать и более глубокие уровни вложенности, например, используя array_diff($array1[0], $array2[0]);.

Функции сравнения массивов

array array_intersect (array array1, array array2 [, array ...]) -- возвращает массив, содержащий значения массива array1, которые содержат все перечисленные в аргументах массивы. Ключи сохраняются.

Пример:

```
$array1 = array ("a" => "green", "red", "blue");
$array2 = array ("b" => "green", "yellow", "red");
$result = array_intersect ($array1, $array2);
// $result будет содержать Array (     [a] => green     [0] => red )
```

Для этой функции актуальны все замечания к функции array_diff().

В PHP существует большое количество функций для всевозможного сравнения массивов: array_diff_ukey, array_udiff, array_diff_asoc, array_udif_assoc, array_udiff_uasoc и т.п.

Более подробную информацию смотрите в руководстве по PHP.

Подсчёт количества элементов массива

int count (mixed var [, int mode]) – возвращает количество элементов переменной var, которая обычно является array, или любым другим объектом, который может содержать хотя бы один элемент. Для объектов count() возвращает количество нестатических свойств, не принимая во внимание видимость.

Если дополнительный параметр mode установлен в COUNT_RECURSIVE (или 1), count() будет считать количество элементов массива рекурсивно (учитывать подмассивы в многомерных массивах).

Предостережение: count() может возвратить 0 для переменных, которые не установлены, но также может возвратить 0 для переменных, которые инициализированы пустым массивом.

Пример 1:

```
$a[0] = 1; $a[1] = 3; $a[2] = 5; $result = count($a); // 3
$b[0]  = 7; $b[5]  = 9; $b[10] = 11; $result = count($b); // 3
count(null);  //  0
count(false); // 1
```

Пример 2:

```
$food = array('fruit' => array('orange', 'banana', 'apple'),              'veggie' => array('carrot', 'collard', 'pea'));
echo count($food, COUNT_RECURSIVE);  // 8
echo count($food); // 2
```

Поиск элемента в массиве

bool in_array (mixed needle, array haystack [, bool strict]) -- ищет в haystack значение needle и возвращает TRUE в случае обнаружения и FALSE в противном случае.

Если третий параметр strict установлен в TRUE тогда функция in_array() также проверит соответствие типа параметра needle и типа соответствующего значения массива haystack. Если needle - строка, сравнение будет регистрозависмым.

Внимание! Если strict не установлен в TRUE, то для массивов, содержащих элемент TRUE, ВСЕГДА будет возвращаться значение TRUE вне зависимости от типа и значения needle.

Пример 1:

```
$os = array("Mac", "NT", “FreBSD", "Linux");
if (in_array(“FreeBSD", $os)) echo “OK"; // OK
if (in_array("mac", $os)) echo “Mac"; // ничего
```

Пример 2.

```
$a = array( array('p', 'h'), array('p', 'r‘), 'o‘ );
if (in_array(array('p', 'h'), $a)) echo 'p h'; // p h
if (in_array(array('f', 'i'), $a)) echo 'f i‘; // ничего
if (in_array('o', $a)) echo 'o'; // o
```

Поиск элемента в массиве

mixed array_search (mixed needle, array haystack [, bool strict]) – ищет в haystack значение needle и возвращает ключ, если такое присутствует в массиве, и FALSE в противном случае.

Если needle является строкой, производится регистро-зависимое сравнение.

Если параметр strict равен TRUE, проверяется совпадение типов.

Если needle присутствует в haystack более одного раза, будет возвращён первый найденный ключ. Для того, чтобы возвратить ключи для всех найденных значений, используйте функцию array_keys() с необязательным параметром search_value.

Пример:

```
$array = array(0 => 'blue', 1 => 'red', 2 => 0x000000, 3 => 'green', 4 => 'red');
$key = array_search('red', $array); // $key = 1;
$key = array_search('green', $array); // $key = 2; (0x000000 == 0 == 'green')
$key = array_search('green', $array, true); // $key = 3;
```

Внимание: эта функция может возвращать как логическое значение FALSE, так и не относящееся к логическому типу значение, которое приводится к FALSE, например, 0 или "". Используйте оператор === для проверки значения, возвращаемого этой функцией.

Поиск элемента в массиве

array array_keys (array input [, mixed search_value]) -- возвращает числовые и строковые ключи, содержащиеся в массиве input.

Если указан необязательный параметр search_value, функция возвращает только ключи, совпадающие с этим параметром.

Пример:

```
$array = array (0 => 100, "color" => "red");
print_r(array_keys ($array));
$array = array ("blue", "red", "green", "blue", "blue");
print_r(array_keys ($array, "blue"));
$array = array ("color" => array("blue", "red", "green"), "size" => array("small", "medium", "large"));
print_r(array_keys ($array));
```

Результат:

```
Array ( [0] => 0 [1] => color )
Array ( [0] => 0 [1] => 3 [2] => 4 )
Array ( [0] => color [1] => size )
```

Поиск элемента в массиве

array array_values (array input) – возвращает индексный массив, содержащий все значения массива input.

Пример:

```
$array = array (“A" => “B", "color" => "gold");
print_r(array_values($array));
```

Выведет:

```
Array ( [0] => B [1] => gold )
```

Сортировка массива

Все нижеперечисленные функции сортируют **ТОЛЬКО первый** уровень массива, т.е. для обработки многомерных массивов этим функциям нужно передавать каждый подмассив отдельно.

bool sort (array &array [, int sort_flags]) – сортирует массив по возрастанию значений элементов.

Эта функция назначает новые ключи для элементов array. Все ранее назначенные значения ключей будут переназначены.

Возвращает TRUE в случае успешного завершения или FALSE в случае возникновения ошибки.

Пример:

```
$fruit = array("lemon", "orange", "banana", "apple");
sort($fruit);
```

Результат:

0 => ‘apple’, 1 => ‘banana’, 2 => ‘lemon’, 3 => ‘orange’

Сортировка массива

Дополнительный второй параметр sort_flags функции sort() можно использовать для изменения поведения сортировки, используя следующие значения:

- SORT_REGULAR – сравнивать элементы нормально (не изменять типы);
- SORT_NUMERIC – сравнивать элементы как числа;
- SORT_STRING – сравнивать элементы как строки;
- SORT_LOCALE_STRING – сравнивать элементы как строки, основываясь на текущей локали (добавлено в PHP 5.0.2).

bool rsort (array &array [, int sort_flags]) – сортирует массив в обратном порядке (от большего к меньшему). В остальном аналогична sort().

Сортировка массива

bool asort (array &array [, int sort_flags]) – сортирует массив по возрастанию значений элементов, сохраняя при этом ключи.

Возвращает TRUE в случае успешного завершения или FALSE в случае возникновения ошибки.

Пример:

```
$fruit = array("d" => "lemon", "a" => "orange", "b" => "banana", "c" => "apple");
asort($fruit);
```

Результат:

c => ‘apple’, b => ‘banana’, d => ‘lemon’, a => ‘orange’

Изменить поведение сортировки можно используя дополнительный параметр sort_flags, подробнее см. sort().

bool arsort (array &array [, int sort_flags]) – сортирует массив по убыванию значений элементов. В остальном аналогична asort().

Сортировка массива

bool ksort (array &array [, int sort_flags]) – сортирует массив по возрастанию значений ключей, сохраняя значения ключей и элементов.

Возвращает TRUE в случае успешного завершения или FALSE в случае возникновения ошибки.

Пример:

```
$fruit = array("d"=>"lemon", "a"=>"orange", "b"=>"banana", "c"=>"apple");
ksort($fruit);
```

Результат:

a => ‘orange’, b => ‘banana’, c => ‘apple’, d => ‘lemon’

Изменить поведение сортировки можно, используя дополнительный параметр sort_flags, подробнее см. sort().

bool krsort (array &array [, int sort_flags]) – сортирует массив по убыванию значений ключей. В остальном аналогична ksort().

Сортировка массива

bool usort (array &array, callback cmp_function) – сортирует элементы массива, используя для сравнения значений пользовательскую функцию. Переназначает ключи.

Функция, используемая для сравнения, должна возвращать целое число, меньшее, равное или большее нуля, если первый аргумент соответственно меньше, равен или больше второго.

Замечание: если два элемента исходного массива равны, их порядок в отсортированном массиве не определён.

Возвращает TRUE в случае успешного завершения или FALSE в случае возникновения ошибки.

Пример:

```
function cmp($a, $b)
 {
   if ($a == $b) return 0;
   return ($a < $b) ? -1 : 1;
 }
$a = array(3, 2, 5, 6, 1);
usort($a, "cmp");
```

Функции ursort() не существует по понятным причинам: вы можете изменить порядок сортировки в пользовательской функции.

Сортировка массива

bool uasort (array &array, callback cmp_function) – сортирует элементы массива, используя для сравнения значений пользовательскую функцию. Сохраняет ключи.

Для сравнения используется функция, определённая пользователем.

Возвращает TRUE в случае успешного завершения или FALSE в случае возникновения ошибки.

bool uksort (array &array, callback cmp_function) – сортирует массив, используя для сравнения его ключей функцию, определённую пользователем.

Функция cmp_function должна принимать два параметра, которым будут присвоены значения двух ключей array.

Функция, используемая для сравнения, должна возвращать целое число, меньшее, равное или большее нуля, если первый параметр считается меньше, равен или больше второго.

Возвращает TRUE в случае успешного завершения или FALSE в случае возникновения ошибки.

В остальном эти функции аналогичны usort().

Сортировка массива

void natsort (array &array) – реализует алгоритм сортировки элементов массива, при котором порядок буквенно-цифровых строк будет привычным для человека. Не сохраняет ключи.

Такой алгоритм называется "natural ordering". Отличие алгоритма "natural ordering" от обычных алгоритмов сортировки, применяемых, например, функцией sort() показывает следующий пример:

```
$array1 = $array2 = array("img12.png", "img10.png", "img2.png", "img1.png");
sort($array1);
echo "Обычная сортировка ";
print_r($array1);
natsort($array2);
echo "Natural order сортировка";
print_r($array2);
```

Результат:

Обычная сортировка

Array ([0] => img1.png [1] => img10.png [2] => img12.png [3] => img2.png)

Natural order сортировка

Array ([3] => img1.png [2] => img2.png [1] => img10.png [0] => img12.png)

Функции natrsort() не существует.

Сортировка массива

void natcasesort (array &array) – нерегистрочувствительный аналог natsort. Не сохраняет ключи.

bool shuffle (array &array) -- перемешивает элементы массива в случайном порядке. Не сохраняет ключи.

array array_reverse (array array [, bool preserve_keys]) -- возвращает новый массив, порядок элементов в котором обратен исходному. Сохраняет ключи, если параметр preserve_keys равен TRUE.

Работа с курсором массива

mixed reset (array &array) – перемещает внутренний указатель массива к его первому элементу и возвращает значение первого элемента или FALSE если массив пуст.

Пример:

```
$arr = array ('step one', 'step two', 'step three', 'step four');
echo reset($arr); // step one
```

mixed end (array &array) – устанавливает внутренний указатель массива на последний элемент и возвращает его значение или FALSE, если массив пуст.

Пример:

```
$arr = array ('step one', 'step two', 'step three', 'step four');
echo end($arr); // step four
```

Работа с курсором массива

mixed next (array &array) – возвращает значение следующего элемента массива и перемещает внутренний указатель на один вперёд, или возвращает FALSE, если достигнут конец массива.

Внимание: если массив содержит пустые или равные 0 элементы, функция возвратит FALSE для этих элементов.

Для того, чтобы правильно просматривать массивы, содержащие пустые элементы, используйте функцию each() или цикл foreach.

Пример:

```
$transport = array('foot', 'bike', 'car', 'plane');
$mode = next($transport); // bike
```

mixed prev (array &array) – возвращает значение предыдущего элемента массива и перемещает внутренний указатель на один назад, или возвращает FALSE, если достигнут конец массива. В остальном аналогична next().

Работа с курсором массива

mixed current (array &array) – возвращает значение элемента массива, на который в данный момент указывает его внутренний указатель. Не перемещает указатель куда бы то ни было.

Если внутренний указатель находится за пределами списка элементов, current() возвращает FALSE.

Внимание: если массив содержит пустые элементы (0 или "", пустая строка), эта функция возвратит FALSE для этих элементов.

Пример:

```
$transport = array('foot', 'bike', 'car', 'plane');
$mode = current($transport); // foot
```

mixed key (array &array) – возвращает индекс (ключ) текущего элемента массива.

Пример:

```
$array = array( 'fruit1' => 'apple', 'fruit2' => 'orange', 'fruit3' => 'grape', 'fruit4' => 'apple', 'fruit5' => 'apple');
echo key($array); // fruit1
```

Работа с курсором массива

array each (array &array) – возвращает текущую пару ключ/значение из массива array и смещает его указатель на один вперёд.

Пара ключ/значение возвращается в виде массива из четырёх элементов, со следующими ключами: 0, 1, key и value.

Элементы 0 и key содержат ключ элемента массива, а элементы 1 и value содержат его значение.

Если внутренний указатель массива указывает на его конец, each() возвратит FALSE.

Пример 1:

```
$a = array("bob", "fred", "jussi", "jouni", "egon", "marliese");
$b = each($foo);
print_r($bar);
```

Выведет:

```
Array (    [1] => bob    [value] => bob    [0] => 0    [key] => 0 )
```

Пример 2:

```
// обход массива функцией each()
$fruit = array('a' => 'apple', 'b' => 'banana', 'c' => 'cranberry');
reset($fruit);
while (($x = each($fruit))!==FALSE) { … }
```

Работа с элементами массива

void list (mixed varname, mixed ...) – подобно array(), это не функция, а языковая конструкция.

Конструкция list() используется для того, чтобы присвоить списку переменных значения за одну операцию.

Замечание: list() работает только с массивами, индексами которых являются числа и нумерация которых начинается с 0.

Пример:

```
$info = array('coffee', 'brown', 'caffeine');
// составить список всех переменных
list($drink, $color, $power) = $info;
echo "$drink is $color and $power makes it special.";
// составить список только некоторых переменных
list($drink, , $power) = $info;
echo "$drink has $power";
```

Внимание! Если в силу каких-то обстоятельств содержимое массива изменится, вы рискуете получить непредсказуемое поведение программы. Лучше избегать использования этой функции!

Работа с элементами массива

bool array_walk (array &array, callback funcname [, mixed userdata]) – применяет пользовательскую функцию funcname к каждому элементу массива array.

Возвращает TRUE в случае успешного завершения или FALSE в случае возникновения ошибки.

Обычно у функции funcname два параметра: значение элемента массива array в качестве первого параметра, и ключ (индекс) в качестве второго. Если указан дополнительный параметр userdata, он будет передан в качестве третьего параметра в функцию обратного вызова funcname.

Если требуется, чтобы функция funcname изменила значения в массиве, определите первый параметр funcname как ссылку. Тогда все изменения будут применены к элементам массива.

Функция array_walk() обойдёт все элементы массива независимо от позиции указателя.

Вы не сможете изменить непосредственно массив при помощи функции обратного вызова, то есть добавить и удалить элементы, уничтожить значения элементов и т.д. Если массив, к которому применяется array_walk(), изменится в процессе её работы (добавятся или удалятся элементы), поведение этой функции станет неопределённым и непредсказуемым.

Работа с элементами массива

Пример:

```
$fruits = array("d" => "lemon", "a" => "orange", "b" => "banana", "c" => "apple");
function test_alter(&$item1, $key, $prefix)
 {
  $item1 = "$prefix: $item1";
 }

function test_print($item2, $key)
 {
  echo "$key. $item2<br />";
 }

echo "Before:";
array_walk($fruits, 'test_print');
array_walk($fruits, 'test_alter', 'fruit');
echo “<br />After:";
array_walk($fruits, 'test_print');
```

Выведет:

Before: d. lemon a. orange b. banana c. apple
After: d. fruit: lemon a. fruit: orange b. fruit: banana c. fruit: apple

Работа с элементами массива

bool array_walk_recursive (array &input, callback funcname [, mixed userdata]) – аналог array_walk(), умеющий работать с многомерными массивами.

Напоминание:

- чтобы проверить, существует ли в массиве элемент с некоторым значением, нужно использовать функцию in_array().
- чтобы проверить, существует ли в массиве элемент с некоторым ключом, нужно использовать функцию isset().

Разбиение в массив и сборка из массива

mixed explode (string separator, string string [, int limit]) – возвращает массив строк, полученных разбиением строки string с использованием separator в качестве разделителя.

Если передан аргумент limit, массив будет содержать максимум limit элементов, при этом последний элемент будет содержать остаток строки string.

Если separator – пустая строка (""), explode() возвращает FALSE.

Если separator не содержится в string, explode() возвращает массив, содержащий один элемент string.

По историческим причинам, функции implode() можно передавать аргументы в любом порядке, но для explode() это недопустимо: separator всегда должен содержать разделитель, а string – исходную строку.

Пример:

```
$pizza  = "piece1 piece2 piece3 piece4 piece5 piece6";
$pieces = explode(" ", $pizza);
echo $pieces[0]; // piece1
```

Эта функция безопасна для обработки данных в двоичной форме.

Разбиение в массив и сборка из массива

string implode (string glue, array pieces) – возвращает строку, полученную объединением строковых представлений элементов массива pieces, со вставкой строки glue между соседними элементами.

Пример:

```
$array = array('lastname', 'email', 'phone');
$comma_separated = implode(",", $array);
echo $comma_separated; // lastname,email,phone
```

По историческим причинам, функции implode() можно передавать аргументы в любом порядке, однако для унификации с функцией explode() следует использовать документированный порядок аргументов.

Начиная с PHP версии 4.3.0 аргумент glue функции implode() является необязательным и по умолчанию равен пустой строке (''). Для обеспечении обратной совместимости рекомендуется всегда передавать оба аргумента.

Эта функция безопасна для обработки данных в двоичной форме.

Функция join() является синонимом функции implode().

Суперглобальные массивы

В PHP есть несколько массивов, носящих громкое название «суперглобальные». Это значит, что они видны «абсолютно отовсюду» (включая внутренние области видимости функций и т.п.)

Вот эти массивы:

$GLOBALS
$_SERVER
$_GET
$_POST
$_COOKIE
$_REQUEST
$_SESSION
$_FILES
$_ENV

Рассмотрим подробнее…

Суперглобальные массивы: $GLOBALS

Содержит ссылку на каждую переменную, доступную в данный момент в глобальной области видимости скрипта (включая все остальные суперглобальные массивы в качестве подмассивов).

Ключами этого массива являются имена глобальны переменных.

Полезен для того, чтобы обращаться к значениям тех или иных переменных из «неглобальных» областей видимости (изнутри функций, методов классов и т.п.)

Содержит КУЧУ информации.

Пример…

Суперглобальные массивы: $GLOBALS

Array ([HTTP_POST_VARS] => Array ( ) **[_POST]** => Array ( ) [HTTP_GET_VARS] => Array ( ) **[_GET]** => Array ( ) [HTTP_COOKIE_VARS] => Array ( ) **[_COOKIE]** => Array ( ) [HTTP_SERVER_VARS] => Array ([HTTP_ACCEPT] => image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, */* [HTTP_ACCEPT_LANGUAGE] => en,ru;q=0.5 [HTTP_USER_AGENT] => Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; Maxthon) [HTTP_HOST] => 127.0.0.1 [HTTP_CONNECTION] => Keep-Alive [PATH] => C:\\WINDOWS\\system32;C:\\WINDOWS;C:\\WINDOWS\\System32\\Wbem;C:\\Program Files\\Common Files\\Adobe\\AGL;c:/archiver;c:/nc/bat [SystemRoot] => C:\\WINDOWS [COMSPEC] => C:\\WINDOWS\\system32\\cmd.exe [PATHEXT] => .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH [WINDIR] => C:\\WINDOWS [SERVER_SIGNATURE] => Apache/2.0.48 (Win32) PHP/4.3.4 Server at 127.0.0.1 Port 80 [SERVER_SOFTWARE] => Apache/2.0.48 (Win32) PHP/4.3.4 [SERVER_NAME] => 127.0.0.1 [SERVER_ADDR] => 127.0.0.1 [SERVER_PORT] => 80 [REMOTE_ADDR] => 127.0.0.1 [DOCUMENT_ROOT] => D:/WWW [SERVER_ADMIN] => adm@zzz.com [SCRIPT_FILENAME] => D:/WWW/9.php [REMOTE_PORT] => 2352 [GATEWAY_INTERFACE] => CGI/1.1 [SERVER_PROTOCOL] => HTTP/1.1 [REQUEST_METHOD] => GET [QUERY_STRING] => [REQUEST_URI] => /9.php [SCRIPT_NAME] => /9.php [PHP_SELF] => /9.php [PATH_TRANSLATED] => D:/WWW/9.php [argv] => Array ( ) [argc] => 0) **[_SERVER]** => Array ([HTTP_ACCEPT] => image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, */* [HTTP_ACCEPT_LANGUAGE] => en,ru;q=0.5 [HTTP_USER_AGENT] => Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; Maxthon) [HTTP_HOST] => 127.0.0.1 [HTTP_CONNECTION] => Keep-Alive [PATH] => C:\\WINDOWS\\system32;C:\\WINDOWS;C:\\WINDOWS\\System32\\Wbem;C:\\Program Files\\Common Files\\Adobe\\AGL;c:/archiver;c:/nc/bat [SystemRoot] => C:\\WINDOWS [COMSPEC] => C:\\WINDOWS\\system32\\cmd.exe [PATHEXT] => .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH [WINDIR] => C:\\WINDOWS [SERVER_SIGNATURE] => Apache/2.0.48 (Win32) PHP/4.3.4 Server at 127.0.0.1 Port 80 [SERVER_SOFTWARE] => Apache/2.0.48 (Win32) PHP/4.3.4 [SERVER_NAME] => 127.0.0.1 [SERVER_ADDR] => 127.0.0.1 [SERVER_PORT] => 80 [REMOTE_ADDR] => 127.0.0.1 [DOCUMENT_ROOT] => D:/WWW [SERVER_ADMIN] => adm@zzz.com [SCRIPT_FILENAME] => D:/WWW/9.php [REMOTE_PORT] => 2352 [GATEWAY_INTERFACE] => CGI/1.1 [SERVER_PROTOCOL] => HTTP/1.1 [REQUEST_METHOD] => GET [QUERY_STRING] => [REQUEST_URI] => /9.php [SCRIPT_NAME] => /9.php [PHP_SELF] => /9.php [PATH_TRANSLATED] => D:/WWW/9.php [argv] => Array ( ) [argc] => 0) [HTTP_ENV_VARS] => Array ([ALLUSERSPROFILE] => C:\\Documents and Settings\\All Users [APPDATA] => C:\\Documents and Settings\\Sv2\\Application Data [CLIENTNAME] => Console [CommonProgramFiles] => C:\\Program Files\\Common Files [COMPUTERNAME] => SV2ND [ComSpec] => C:\\WINDOWS\\system32\\cmd.exe [FP_NO_HOST_CHECK] => NO [HOMEDRIVE] => C: [HOMEPATH] => \\Documents and Settings\\Sv2 [LOGONSERVER] => \\\\SV2ND [NUMBER_OF_PROCESSORS] => 1 [OS] => Windows_NT [Path] => C:\\WINDOWS\\system32;C:\\WINDOWS;C:\\WINDOWS\\System32\\Wbem;C:\\Program Files\\Common Files\\Adobe\\AGL;c:/archiver;c:/nc/bat [PATHEXT] => .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH [PROCESSOR_ARCHITECTURE] => x86 [PROCESSOR_IDENTIFIER] => x86 Family 15 Model 4 Stepping 1, GenuineIntel [PROCESSOR_LEVEL] => 15 [PROCESSOR_REVISION] => 0401 [ProgramFiles] => C:\\Program Files [SESSIONNAME] => Console [SystemDrive] => C: [SystemRoot] => C:\\WINDOWS [TEMP] => c:/Temp [TMP] => C:/Temp [USERDOMAIN] => SV2ND [USERNAME] => Sv2 [USERPROFILE] => C:\\Documents and Settings\\Sv2 [windir] => C:\\WINDOWS [AP_PARENT_PID] => 1948) **[_ENV]** => Array ([ALLUSERSPROFILE] => C:\\Documents and Settings\\All Users [APPDATA] => C:\\Documents and Settings\\Sv2\\Application Data [CLIENTNAME] => Console [CommonProgramFiles] => C:\\Program Files\\Common Files [COMPUTERNAME] => SV2ND [ComSpec] => C:\\WINDOWS\\system32\\cmd.exe [FP_NO_HOST_CHECK] => NO [HOMEDRIVE] => C: [HOMEPATH] => \\Documents and Settings\\Sv2 [LOGONSERVER] => \\\\SV2ND [NUMBER_OF_PROCESSORS] => 1 [OS] => Windows_NT [Path] => C:\\WINDOWS\\system32;C:\\WINDOWS;C:\\WINDOWS\\System32\\Wbem;C:\\Program Files\\Common Files\\Adobe\\AGL;c:/archiver;c:/nc/bat [PATHEXT] => .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH [PROCESSOR_ARCHITECTURE] => x86 [PROCESSOR_IDENTIFIER] => x86 Family 15 Model 4 Stepping 1, GenuineIntel [PROCESSOR_LEVEL] => 15 [PROCESSOR_REVISION] => 0401 [ProgramFiles] => C:\\Program Files [SESSIONNAME] => Console [SystemDrive] => C: [SystemRoot] => C:\\WINDOWS [TEMP] => c:/Temp [TMP] => C:/Temp [USERDOMAIN] => SV2ND [USERNAME] => Sv2 [USERPROFILE] => C:\\Documents and Settings\\Sv2 [windir] => C:\\WINDOWS [AP_PARENT_PID] => 1948) [HTTP_POST_FILES] => Array ( ) **[_FILES]** => Array ( ) **[_REQUEST]** => Array ( ) **[GLOBALS]** => Array *RECURSION*)

Суперглобальные массивы: $_SERVER

Содержит переменные, установленные web-сервером либо напрямую связанные с окружением выполнения текущего скрипта.

Содержащаяся здесь информация бывает очень полезной для автоматической настройки скрипта на те или иные условия (версию PHP, веб-сервера, операционной системы, ip-пользователя, браузер пользователя и т.д.)

Пример…

Суперглобальные массивы: $_SERVER

Array ([HTTP_ACCEPT] => */* [HTTP_ACCEPT_LANGUAGE] => en,ru;q=0.5 [HTTP_USER_AGENT] => Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; Maxthon) [HTTP_HOST] => 127.0.0.1 [HTTP_CONNECTION] => Keep-Alive [PATH] => C:\\WINDOWS\\system32;C:\\WINDOWS;C:\\WINDOWS\\System32\\Wbem;C:\\Program Files\\Common Files\\Adobe\\AGL;c:/archiver;c:/nc/bat [SystemRoot] => C:\\WINDOWS [COMSPEC] => C:\\WINDOWS\\system32\\cmd.exe [PATHEXT] => .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH [WINDIR] => C:\\WINDOWS [SERVER_SIGNATURE] => Apache/2.0.48 (Win32) PHP/4.3.4 Server at 127.0.0.1 Port 80 [SERVER_SOFTWARE] => Apache/2.0.48 (Win32) PHP/4.3.4 [SERVER_NAME] => 127.0.0.1 [SERVER_ADDR] => 127.0.0.1 [SERVER_PORT] => 80 [REMOTE_ADDR] => 127.0.0.1 [DOCUMENT_ROOT] => D:/WWW [SERVER_ADMIN] => adm@zzz.com [SCRIPT_FILENAME] => D:/WWW/9.php [REMOTE_PORT] => 2368 [GATEWAY_INTERFACE] => CGI/1.1 [SERVER_PROTOCOL] => HTTP/1.1 [REQUEST_METHOD] => GET [QUERY_STRING] => [REQUEST_URI] => /9.php [SCRIPT_NAME] => /9.php [PHP_SELF] => /9.php [PATH_TRANSLATED] => D:/WWW/9.php [argv] => Array ( ) [argc] => 0)

Рассмотрим содержимое этого массива подробнее, т.к. нам придётся часто с ним работать.

Внимание! При запуске скрипта из командной строки (НЕ через браузер) многие элементы этого массива будут недоступны.

Суперглобальные массивы: $_SERVER

'PHP_SELF' – имя файла исполняемого в данный момент скрипта относительно document root; например, для скрипта http://example.com/test/a.php здесь будет /test/a.php. Если PHP запущен из командной строки, эта переменная недоступна.

'argv' – массив аргументов, передаваемых скрипту. Если скрипт работает из командной строки, это даёт доступ, в стиле C, к параметрам командной строки. Если скрипт вызывается через метод GET, здесь будет находиться запроса.

'argc' – содержит количество параметров командной строки, передаваемых скрипту (если запущен из командной строки).

'GATEWAY_INTERFACE' – какой вариант спецификации CGI используется сервером; например, 'CGI/1.1'.

'SERVER_NAME' – имя хоста сервера, на котором текущий скрипт выполняется. Если скрипт запущен на виртуальном хосте, это будет значение, определённое для данного виртуального хоста.

'SERVER_SOFTWARE' – строка-идентификатор («сигнатура») сервера, передаваемая в заголовке HTTP-ответа.

'SERVER_PROTOCOL' – имя и версия протокола, по которому страница запрошена; например, 'HTTP/1.0'.

Суперглобальные массивы: $_SERVER

'REQUEST_METHOD' – метод, использованный для запроса страницы; например, 'GET', 'HEAD', 'POST', 'PUT'.

'QUERY_STRING' – строка запроса (параметры, переданные методом GET), если таковые имеются.

'DOCUMENT_ROOT' – корневая директория документов, под которой выполняется текущий скрипт, как определено в файле конфигурации сервера.

'HTTP_ACCEPT' – содержимое параметра Accept из текущего запроса, если имеется.

'HTTP_ACCEPT_CHARSET' – содержимое параметра Accept-Charset из текущего запроса, если имеется. Пример: 'iso-8859-1,*,utf-8'.

'HTTP_ACCEPT_ENCODING' – содержимое параметра Accept-Encoding из текущего запроса, если имеется. Пример: 'gzip'.

'HTTP_ACCEPT_LANGUAGE' – содержимое параметра Accept-Language из текущего запроса, если имеется. Пример: 'en'. Удобно для автоматического отображения «языка по умолчанию» для мультиязычных сайтов.

Суперглобальные массивы: $_SERVER

'HTTP_CONNECTION' – содержимое параметра Connection из текущего запроса, если имеется. Пример: 'Keep-Alive'.

'HTTP_HOST' – содержимое параметра Host: из текущего запроса, если имеется. Удобно для мультидоменных сайтов.

'HTTP_REFERER' – адрес страницы, с который пользователь пришёл на эту страницу. Не все агенты будут его устанавливать, а некоторые могут модифицировать.

'HTTP_USER_AGENT' – содержимое параметра User_Agent из текущего запроса, если имеется. Это строка, обозначающая агент, выполнивший запрос к странице. Типичный пример: Mozilla/4.5 [en] (X11; U; Linux 2.2.9 i586). Помимо прочего, вы можете использовать это значение с функцией get_browser() для приспособления вывода вашей страницы к возможностям агента.

'REMOTE_ADDR' – IP-адрес, с которого пользователь просматривает текущую страницу.

'REMOTE_PORT' – порт на пользовательской машине для соединения с web-сервером.

'SCRIPT_FILENAME' – абсолютный путь к файлу исполняемого в данный момент скрипта (в файловой системе).

Суперглобальные массивы: $_SERVER

'SERVER_ADMIN' – значение, данное в директиве SERVER_ADMIN (для Apache) в файле конфигурации web-сервера. Если скрипт запущен на виртуальном хосте, это будет значение, определённое для данного виртуального хоста. Здесь содержится e-mail администратора сайта. Удобно для отправки уведомлений.

'SERVER_PORT' – порт на серверной машине, используемый веб-сервером для соединения. По умолчанию это '80‘.

'SERVER_SIGNATURE' – строка, содержащая версию сервера и имя виртуального хоста, добавленная к генерируемым сервером страницам, если эта возможность включена.

'PATH_TRANSLATED' – путь в файловой системе к текущему скрипту, после того как сервер выполнил отображение virtual-to-real (актуально для posix-систем).

'SCRIPT_NAME' – путь к текущему скрипту (в файловой системе).

'REQUEST_URI' – URI, который был задан для запроса данной страницы; например, '/index.html'.

Суперглобальные массивы: $_SERVER

'PHP_AUTH_USER' – при выполнении HTTP-аутентификации, в эту переменную устанавливается username, предоставляемое пользователем.

'PHP_AUTH_PW' – при выполнении HTTP-аутентификации, в эту переменную устанавливается password, предоставляемый пользователем.

'PHP_AUTH_TYPE' – при выполнении HTTP-аутентификации, в эту переменную устанавливается тип аутентификации.

Суперглобальные массивы: $_GET

Содержит данные, переданные скрипту методом GET.
Пример:

```
print_r($_GET);
// Array ( [a] => 10 [b] => 20 )
```

Суперглобальные массивы: $_POST

Содержит данные, переданные скрипту методом POST.
Пример:

```
print_r($_POST);
// Array ( [x] => 100 [y] => 200 )
```

Суперглобальные массивы: $_COOKIE

Содержит данные, переданные скрипту через COOKIE (о cookie погорим позже).

Пример:

```
print_r($_COOKIE);
// Array ( [name] => Vasya [lastip] => 192.168.0.1 )
```

Суперглобальные массивы: $_REQUEST

Содержит данные, переданные скрипту через все методы GET, POST, COOKIE, FILES.

Как вы думаете, что объединяет данные, содержащиеся в этом массиве?

Они пришли от пользователя, следовательно, им нельзя доверять.

Пример:

```
print_r($_REQUEST);
// Array ( [a] => Vasya [b] => Pupkin )
```

Суперглобальные массивы: $_SESSION

Содержит данные, сохранённые с использованием механизма сессий, о котором мы поговорим позже.

Пример:

```
print_r($_SESSION);
// Array ( [a] => Vasya [b] => Pupkin )
```

Суперглобальные массивы: $_FILES

Содержит данные о переданных в процессе текущего запроса файлах. Содержимое массива и варианты его организации мы рассмотрим в соответствующей теме.

Суперглобальные массивы: $_ENV

Содержит данные окружения, в котором запущен PHP. Многие предоставляются ОС, под которой PHP работает, и поэтому создать какой-то определённый список невозможно.

Пример:

Array ([ALLUSERSPROFILE] => C:\\Documents and Settings\\All Users [APPDATA] => C:\\Documents and Settings\\Sv2\\Application Data [CLIENTNAME] => Console [CommonProgramFiles] => C:\\Program Files\\Common Files [COMPUTERNAME] => SV2ND [ComSpec] => C:\\WINDOWS\\system32\\cmd.exe [FP_NO_HOST_CHECK] => NO [HOMEDRIVE] => C: [HOMEPATH] => \\Documents and Settings\\Sv2 [LOGONSERVER] => \\\\SV2ND [NUMBER_OF_PROCESSORS] => 1 [OS] => Windows_NT [Path] => C:\\WINDOWS\\system32;C:\\WINDOWS;C:\\WINDOWS\\System32\\Wbem;C:\\Program Files\\Common Files\\Adobe\\AGL;c:/archiver;c:/nc/bat [PATHEXT] => .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH [PROCESSOR_ARCHITECTURE] => x86 [PROCESSOR_IDENTIFIER] => x86 Family 15 Model 4 Stepping 1, GenuineIntel [PROCESSOR_LEVEL] => 15 [PROCESSOR_REVISION] => 0401 [ProgramFiles] => C:\\Program Files [SESSIONNAME] => Console [SystemDrive] => C: [SystemRoot] => C:\\WINDOWS [TEMP] => c:/Temp [TMP] => C:/Temp [USERDOMAIN] => SV2ND [USERNAME] => Sv2 [USERPROFILE] => C:\\Documents and Settings\\Sv2 [windir] => C:\\WINDOWS [AP_PARENT_PID] => 1948)

Массивы

На этом мы заканчиваем рассмотрение основ работы с массивами в PHP. Практику их применения мы пройдём в следующих темах.