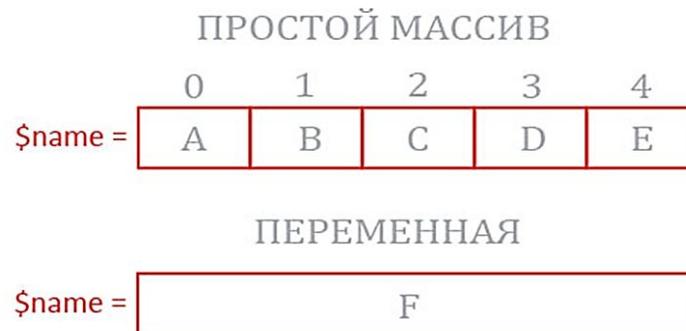


Массивы

Массивы

Массив – это упорядоченный набор данных, идентифицируемых с помощью одного (одномерные) или нескольких (многомерные) индексов.



Основные типа массивов:

- Гомогенные (однотипные) и гетерогенные;
- Индексные (линейные) и ассоциативные;
- Статические и динамические.

Гомогенные и гетерогенные массивы

Гомогенный массив – массив, содержащий элементы данных одного типа.

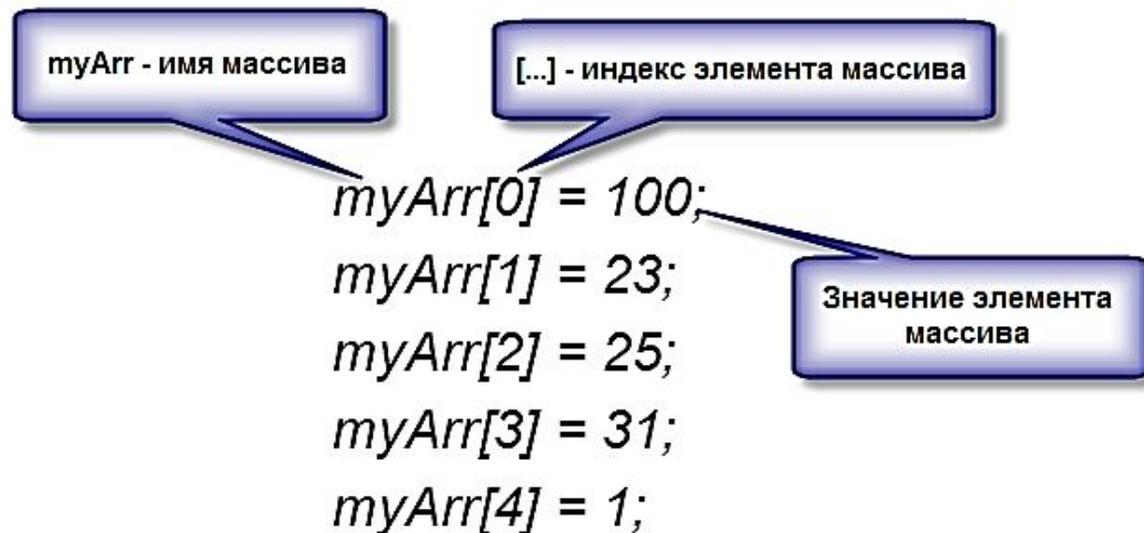
Гетерогенный массив – массив, в разные элементы которого могут быть записаны значения, относящиеся к различным типам данных.

Гетерогенные массивы удобны как универсальная структура для хранения наборов данных произвольных типов. С другой стороны, реализация гетерогенности требует усложнения механизма поддержки массивов в трансляторе языка. Гетерогенный массив как встроенный тип данных присутствует в языке PHP.

Индексный массив

Индексный (простой) массив — именованный набор переменных, расположенных в памяти непосредственно друг за другом, доступ к которым осуществляется по индексу.

Индекс массива — обычно целое число, указывающее на конкретный элемент массива.



Ассоциативный массив

Представляет собой массив, хранящий значения элементов, связывая элементы со значением **ключей**, а не хранят элементы в строгом порядке линейных индексов. Если в массиве сохранен некоторый элемент в ассоциации с ключом, то для последующей выборки элементов из массива достаточно указать значение ключа.

`myArr` - название массива

[...] - ключ элемента массива

Значение элемента массива

```
myArr['Педадькина'] = 'Фекла';  
myArr['Сидоров'] = 'Сидор';  
myArr['Пупкин'] = 'Василий';
```

Индексный (простой) и ассоциативный массивы

ПРОСТОЙ МАССИВ

	0	1	2	3	4
\$name =	A	B	C	D	E

АССОЦИАТИВНЫЙ МАССИВ

	white	black	red	green	blue
\$name =	белый	черный	красный	зеленый	синий

Статические и динамические массивы

Динамическим называется массив, размер которого может меняться во время исполнения программы. Динамические массивы дают возможность более гибкой работы с данными, так как позволяют не прогнозировать хранимые объёмы данных, а регулировать размер массива в соответствии с реально необходимыми объёмами.

Обычные, не динамические массивы называют ещё **статическими**.

Массивы в РНР

Массивы РНР позволяют хранить данные сразу нескольких типов и автоматически организовывать эти данные от имени пользователя с помощью широкого набора различных средств, т.е. являются **гетерогенными**.

Интерпретатор РНР не делает различий между простыми (индексируемыми) и ассоциативными массивами; не распределяет фиксированное количество участков для элементов массива, а создает участки по мере добавления новых элементов к массиву (**динамические массивы**).

Массивы в PHP

Массивы PHP не требуют, чтобы все элементы относились к одному типу, поэтому элементам массивов можно присваивать произвольные значения PHP.

Ключи массивов могут быть только строкового (string) или целочисленного (integer) типа, значения могут относиться к любому типу:

```
$my_array[1] = "This is the second element of the array";  
$my_array['orange'] = 2;  
$my_array[3] = false;
```

В результате в массиве появляются три элемента со значениями (*"This is the second element of the array"*, 2, *false*), каждый из которых хранится в ассоциации с определенным ключом (1, 'orange', 3).

Ключи массивов

Ключ может быть либо типа `integer`, либо типа `string`.

Ключи других типов будут приведены следующим образом:

- Строки, содержащие целое число будут преобразованы к типу `integer`. Например, ключ со значением "8" будет в действительности сохранен со значением 8. С другой стороны, значение "08" не будет преобразовано, так как оно не является корректным десятичным целым;
- Числа с плавающей точкой (тип `float`) будут преобразованы к типу `integer`;
- Тип `bool` преобразуется к типу `integer`;
- Тип `null` будет преобразован к пустой строке ("").
- Массивы (тип `array`) и объекты (тип `object`) не могут использоваться в качестве ключей. При подобном использовании будет генерироваться предупреждение: *Недопустимый тип смещения (Illegal offset type)*.

Создание массивов

Существуют три основных способа:

1. Непосредственное присваивание значения одному из элементов массива (и тем самым неявное создание массива);
2. Конструкция `array()`. Начиная с версии PHP 5.4 возможно использование короткого синтаксиса `[]`;
3. Вызов функции, которая возвращает массив в качестве значения.

Непосредственное присваивание

Простейший способ создания массива, состоящий в выполнении с некоторой переменной таких действий, как будто эта переменная уже представляет собой массив:

```
$my_array[1] = "The first thing in my array";
```

Пример массива со строковым ключом:

```
<?php
$names["Иванов"] = "Иван";
$names["Сидоров"] = "Сергей";
$names["Петрова"] = "Мария";

// Доступ по ключу:
Echo $names["Иванов"];
?>
```

Конструкция array()

Создает новый массив элементов; в качестве параметров принимает любое количество разделенных запятыми пар `key => value` (ключ => значение). Общий синтаксис:

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

Конструкция array()

Пример создания нового массива с именем \$fruit_basket:

```
$fruit_basket = array('apple', 'orange', 'banana', 'pear');
```

Имя переменной \$fruit_basket становится именем массива с четырьмя строковыми элементами, имеющими индексы (0,1,2,3).

Задание индексов с помощью конструкции `array()`

Существует специальная синтаксическая конструкция для указания на то, какие должны быть индексы. Вместо перечисления элементов, разделенных запятыми, в этой конструкции можно задать разделенные запятыми пары «ключ – значение», в которых ключи и значения разделены с помощью специального символа `=>`:

```
$fruit_basket = array (  
    0 => 'apple',  
        1 => 'orange',  
        2 => 'banana',  
        3 => 'pear'  
);
```

Создание массива с PHP 5.4

В версии PHP 5.4 и выше возможно использование **короткого синтаксиса** объявления массива:

```
$arr1 = [1, 2, 3, 4];
```

```
$arr2 = ['one' => 1, 'two' => 2,  
        'three' => 3, 'four' => 4];
```

```
$fruit_basket = [0 => 'apple',  
                1 => 'orange',  
                2 => 'banana',  
                3 => 'pear' ];
```

Некоторые функции, возвращающие массивы

Чаще всего такие функции используются при обращении к базам данных:

<i>Функция</i>	<i>Назначение функции</i>
<i>mysql_fetch_row</i>	Обрабатывает ряд результата запроса и возвращает неассоциативный массив.
<i>mysql_fetch_assoc</i>	Обрабатывает ряд результата запроса и возвращает ассоциативный массив.
<i>mysql_fetch_array</i>	Обрабатывает ряд результата запроса, возвращая ассоциативный массив, численный массив или оба.

Пример №1:

```
<?php require_once ("connections/MySiteDB.php");
$select_db = mysqli_select_db ($link, $db);
$query = "SELECT * FROM notes";
$result = mysqli_query ($link, $query);
while ($note = mysqli_fetch_assoc ($result))
{
    foreach ($note as $key => $value)
    {
        echo $key." ".$value."<br> ";
    }
}
?>
```

id 1
title Заметка №1
added 2011-02-01
content Это первая заметка моего блога.

id 2
title Заметка №2
added 2011-02-01
content Это вторая заметка моего блога. Было бы неплохо сделать ее подлиннее

id 3
title Заметка №3
added 2011-02-02
content Количество заметок моего блога растет. Неплохо было бы внести еще некий смысл =)

id 4
title Заметка №4
added 2011-02-07
content Это моя четвертая заметка %))

Пример №2

```
<?php require_once ("connections/MySiteDB.php");
$select_db = mysqli_select_db ($link, $db);
$query = "SELECT * FROM notes";
$result = mysqli_query ($link, $query);
while ($note = mysqli_fetch_array ($result))
{
    foreach ($note as $key => $value)
    {
        echo $key." ".$value."<br>";
    }
}
?>
```

```
0 1
id 1
1 Заметка №1
title Заметка №1
2 2011-02-01
added 2011-02-01
3 Это первая заметка моего блога.
content Это первая заметка моего блога.
```

```
0 2
id 2
1 Заметка №2
title Заметка №2
2 2011-02-01
added 2011-02-01
3 Это вторая заметка моего блога. Было бы неплохо сделать ее подлиннее
content Это вторая заметка моего блога. Было бы неплохо сделать ее подлиннее
```

```
0 3
id 3
1 Заметка №3
title Заметка №3
2 2011-02-02
added 2011-02-02
3 Количество заметок моего блога растет. Неплохо было бы внести еще некий смысл =)
content Количество заметок моего блога растет. Неплохо было бы внести еще некий смысл =)
```

Некоторые функции, возвращающие массивы

range — создает массив, содержащий диапазон элементов.

```
array range ( mixed $start , mixed $end [, number $step = 1 ] )
```

, где

- Start - первое значение последовательности.
- End - конечное значение, которым заканчивается последовательность.
- Step - если указан параметр step, то он будет использоваться как инкремент между элементами последовательности. step должен быть положительным числом. Если step не указан, он принимает значение по умолчанию 1.

```
<?php
```

```
// array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12)
    foreach (range(0, 12) as $number) {
        echo $number;
    }
?>
```

Многомерные массивы

Многомерный массив – это массив, который содержит в себе другие массивы (т.н. «массив массивов»).

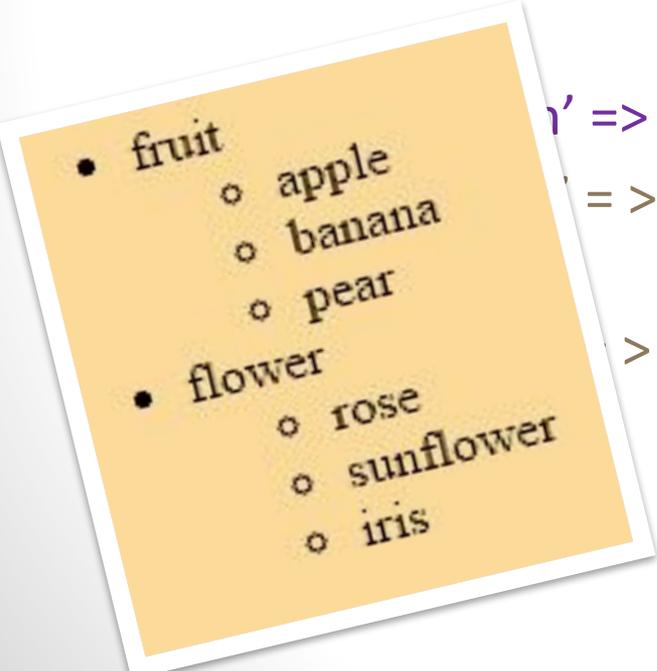
`$multi_array [1][2][3][4][5] = “...”;`

\$massiv =	Стационарный ПК			Ноутбук			Нетбук		
	ОЗУ	НDD	ГЦ	ОЗУ	НDD	ГЦ	ОЗУ	НDD	ГЦ
	4096	500	3	3072	320	2	2048	250	1,6

Многомерные массивы

Пример многомерного массива:

```
$multi_array = array ('fruit' =>  
    array('red' => 'apple',  
        'orange' => 'orange',  
        'yellow' => 'banana',  
        'green' => 'pear'),  
    'flower' =>  
    array('red' => 'rose',  
        'yellow' => 'sunflower',  
        'blue' => 'iris'))
```

- 
- fruit
 - apple
 - banana
 - pear
 - flower
 - rose
 - sunflower
 - iris

Цикл просмотра foreach

Позволяет выполнять действия над элементами массива.

Синтаксис:

```
foreach (<Имя массива> as <Индекс> =><Значение>)  
{  
    <Тело цикла>;  
}
```

Функции перемещения по элементам массива

- `current()` – значение текущей ячейки;
- `next()` – значение следующей ячейки;
- `prev()` – значение предыдущей ячейки;
- `end()` – значение последней ячейки;
- `reset()` – значение первой ячейки.

Операторы для работы с массивами

Пример	Название	Результат
$\$a + \b	Объединение	Объединение массива $\$a$ и массива $\$b$.
$\$a == \b	Равно	TRUE в случае, если $\$a$ и $\$b$ содержат одни и те же пары ключ/значение.
$\$a === \b	Тождественно равно	TRUE в случае, если $\$a$ и $\$b$ содержат одни и те же пары ключ/значение в том же самом порядке и того же типа.
$\$a != \b	Не равно	TRUE, если массив $\$a$ не равен массиву $\$b$.
$\$a <> \b	Не равно	TRUE, если массив $\$a$ не равен массиву $\$b$.
$\$a !== \b	Тождественно не равно	TRUE, если массив $\$a$ не равен тождественно массиву $\$b$.

Пример использования операторов для работы с массивами (объединение массивов)

```
<?php
$a = array("a" => "first", "b" => "second");
$b = array("q" => "1", "w" => "2", "z" => "3");

$c = $a + $b;
echo "Объединение массивов \$a и \$b: \n";
echo "<pre>", var_dump($c), "</pre>";

$c = $b + $a;
echo "Объединение массивов \$b и \$a: \n";
echo "<pre>", var_dump($c), "</pre>";
?>
```

Объединение массивов \$a и \$b:

```
array(5) {
  ["a"]=>
  string(5) "first"
  ["b"]=>
  string(6) "second"
  ["q"]=>
  string(1) "1"
  ["w"]=>
  string(1) "2"
  ["z"]=>
  string(1) "3"
}
```

Объединение массивов \$b и \$a:

```
array(5) {
  ["q"]=>
  string(1) "1"
  ["w"]=>
  string(1) "2"
  ["z"]=>
  string(1) "3"
  ["a"]=>
  string(5) "first"
  ["b"]=>
  string(6) "second"
}
```

Пример использования операторов для работы с массивами (сравнение массивов)

```
<?php
$a = array("apple", "banana");
$b = array(1 => "banana", "0" => "apple");

var_dump($a == $b); // true
var_dump($a === $b); // false
?>
```

Функция ***array_diff()*** находит расхождения в массивах

```
<?php
$array1 = array("a" => "green", "red", "blue", "red");
$array2 = array("b" => "green", "yellow", "red");
$result = array_diff($array1, $array2);

print_r($result); // Array ( [1] => blue)
?>
```

Основные функции сортировки массивов

- `sort` – сортировка массива по возрастанию;
- `rsort` — сортировка массива по убыванию;
- `krsort` – сортировка массива по ключам по возрастанию;
- `krsort` — сортировка массива по ключам по убыванию;
- `usort` — Сортирует массив по значениям используя
- пользовательскую функцию для сравнения элементов
- `uksort` — Сортирует массив по ключам, используя
- пользовательскую функцию для сравнения ключей
- `uasort` — Сортирует массив, используя
- пользовательскую
- функцию для сравнения элементов с сохранением
- ключей

и т.п.

Базовые функции для работы с массивами

- `count (sizeof) ()` - подсчитывает количество элементов массива;
- `print_r()` - выводит информацию о массиве;
- `var_dump()` - выводит структурированную информацию о массиве;
- `list()` - присваивает переменным из списка значения подобно массиву;
- `compact()` – упаковка элементов в массив;
- `extract()` – извлечение элементов из массива;
- `unset()` – удаление массива;
- другие функции, включая функции, начинающиеся с *array*, например, *array_combine()*

Суперглобальные массивы (переменные)

Суперглобальные переменные (массивы) - это встроенные переменные (массивы), которые всегда доступны во всех областях видимости в любом месте скрипта:

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

\$GLOBALS

Представляет собой ассоциативный массив, содержащий ссылки на все переменные глобальной области видимости скрипта, определенные в данный момент.

Имена переменных являются ключами массива.

`$_SERVER`

Представляют собой массив, содержащий информацию, такую как заголовки, пути и местоположения скриптов. Записи в этом массиве создаются веб-сервером (следовательно, их наличие и содержание определяется настройками сервера).

Основные индексы `$_SERVER`

- `'PHP_SELF'` - имя файла скрипта, который сейчас выполняется, относительно корня документов;
- `'SERVER_ADDR'` - IP адрес сервера, на котором выполняется текущий скрипт;
- `'SERVER_NAME'` - имя хоста, на котором выполняется текущий скрипт;
- `REQUEST_METHOD` - какой метод был использован для запроса страницы;
- `'DOCUMENT_ROOT'` – корневая папка документов, в которой выполняется текущий скрипт, в точности та, которая указана в конфигурационном файле сервера;
- `'REMOTE_ADDR'` - IP-адрес, с которого пользователь просматривает текущую страницу;
- и т.п.

Глобальные массивы

- `$_GET`, `$_POST`, `$_REQUEST` - ассоциативные массивы, работающие с данными, передаваемыми методами GET, POST;
- `$_FILES` - ассоциативный массив, содержащий данные файлов, загруженных по HTTP;
- `$_COOKIE` – ассоциативный массив, содержащий значения, переданные скрипту через HTTP cookie;
- `$_SESSION` - ассоциативный массив, содержащий переменные сессии, которые доступны для текущего скрипта;
- `$_ENV` - ассоциативный массив значений, переданных скрипту через переменные окружения. Эти значения импортируются в глобальное пространство имен PHP из системных переменных окружения, в котором запущен парсер PHP. Большинство значений передаётся из командной оболочки, под которой PHP запущен, и различных системных приложений (полного и точного списка не существует).