Функции за обработка на масиви

В тази тема:

1. Въведение в масивите
2. Създаване на масиви
3. Показване на съдържанието на масив
4. Функции за сортиране на масиви
5. Изтриване на елементи на масивите

1. Въведение в масивите

- Масивът е подредена съвкупност от данни. Напр.:
  - red
  - green
  - blue
  - black
  - white
- Всеки елемент се обозначава с индекс, който служи за достъп до него.

Видове масиви в PHP:

- **Индексни масиви** - масиви с числови индекси
- **Асоциативни масиви** – масиви с именувани ключове
- **Многомерни масиви** – масиви, съдържащи един или повече масиви

2. Създаване на масиви

Масив може да се създаде с функцията **array()**.

array array ([mixed $...])

- Може да се използва синтаксис "индекс => стойности", разделени със запетаи.
- Индексите могат да бъдат от низов или от целочислен тип.
- При пропускане на индекса се генерира целочислен индекс, започващ от 0.

2.1. Създаване на индексни масиви

Създават се по 2 начина:

- ***С автоматично присвояване на индексите (винаги започват от 0)***

```
$cars = array("Volvo", "BMW", "Toyota");
```

- ***С ръчно присвояване на индексите***

```
$cars2[1] = "Volvo";
$cars2[2] = "BMW";
$cars2[3] = "Toyota";
```


Цикъл за индексиран масив

Дължината на масив (броя на елементите) се получава с функцията **count()** или **sizeof()**.

Пример:

```
$arrlength = count($cars);

$cars = array("Volvo", "BMW", "Toyota");
$arrlength = count($cars);

for($x = 0; $x < $arrlength; $x++) {
   echo $cars[$x];
   echo "<br>";
}
```


2.2. Създаване на асоциативни масиви

Асоциативните масиви позволяват използването на смислови стойности на ключовете за всяка стойност.

Създават се по 2 начина:

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
```

или

```
$age['Peter'] = "35";
$age['Ben'] = "37";
$age['Joe'] = "43";
```

Цикъл за асоциативен масив

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

foreach($age as $x => $x_value) {
   echo "Key=" . $x . ", Value=" . $x_value;
   echo "<br>";
}
```


2.3. Създаване на многомерни масиви

Пример: Създаване на двумерен масив

Name	Stock	Sold
Volvo	22	18
BMW	15	13
Saab	5	2
Land Rover	17	15

```
$cars = array
 (
 array("Volvo",22,18),
 array("BMW",15,13),
 array("Saab",5,2),
 array("Land Rover",17,15)
 );
```


Достъпът до елементите на масива $cars се извършва с посочване на два индекса – **ред** и **колона**.

```
echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2].".<br>";
echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2].".<br>";
echo $cars[2][0].": In stock: ".$cars[2][1].", sold: ".$cars[2][2].".<br>";
echo $cars[3][0].": In stock: ".$cars[3][1].", sold: ".$cars[3][2].".<br>";
```


Цикъл за многомерен масив

```
for ($row = 0; $row < 4; $row++) {
  echo "<p><b>Ред No $row</b></p>";
  echo "<ul>";
  for ($col = 0; $col < 3; $col++) {
    echo "<li>".$cars[$row][$col]."</li>";
  }
  echo "</ul>";
}
```

3. Показване на съдържанието на масив

За целите на дебъгинга има 2 функции, които могат да показват съдържанието на масиви:

print_r - показва информация за променлива, която лесно се чете от човека

print_r($age);

var_dump – показва информация за променлива

var_dump($age);

4. Функции за сортиране на масиви

sort()

Сортира масиви в увеличаващ се ред

bool sort (array &$array [, int $sort_flags = SORT_REGULAR])

- array – входящия масив
- sort_flags – флаг, който може да промени начина на подреждане.
  - SORT_REGULAR – сравняване на елементите нормално (без смяна на типа)
  - SORT_NUMERIC – сравнява елементите като числа
  - SORT_STRING – сравнява елементите като стрингове
  - SORT_LOCALE_STRING – сравнява елементите като стрингове, базирани на текущия compare items as strings, based on the current locale. It uses the locale, which can be changed using setlocale()
  - SORT_NATURAL - compare items as strings using "natural ordering" like natsort()
  - SORT_FLAG_CASE - can be combined (bitwise OR) with SORT_STRING or SORT_NATURAL to sort strings case-insensitively

Пример: Сортиране на масив в увеличаващ се азбучен ред.

$cars = array("Volvo", "BMW", "Toyota");
sort($cars);

Пример: Сортиране на елементите на масив в увеличаващ се числов ред.

$numbers = array(4, 6, 2, 22, 11);
sort($numbers);

rsort()

Сортира масив в обратен ред.

bool rsort (array &$array [, int $sort_flags = SORT_REGULAR])

Пример: Сортиране на елементите на масив в намалящ числов ред.

$numbers = array(4, 6, 2, 22, 11);
rsort($numbers);

Сортиране на асоциативни масиви

- **asort()** – сортира асоциативен масив в увеличаващ се ред в съответсветствие стойността на елементите
- **ksort()** – сортира асоциативен масив в увеличаващ се ред в съответствие на ключовете (индексите)
- **arsort()** – сортира асоциативен масив в намаляващ ред в съответсветствие стойността на елементите
- **krsort()** - сортира асоциативен масив в намаляващ ред в съответствие на ключовете (индексите)

asort()

Сортира масив, запазвайки връзката между ключовете и стойностите.

bool asort (array &$array [, int $sort_flags = SORT_REGULAR])

- Подрежда стойностите на масива
- Използва се главно за асоциативни масиви

Пример: Сортиране на елементите на асоциативен масив.

```
$fruits = array("d" => "lemon", "a" => "orange", "b" =>
"banana", "c" => "apple");
asort($fruits);
foreach ($fruits as $key => $val) {
    echo "$key = $val\n";
}
```

Резултатът е:

c = apple
b = banana
d = lemon
a = orange

5. Изтриване на елементи на масивите

Изтриването на елемент на масив се извършва с функцията **unset()**

Пример:

```
unset($cars["Volvo"]);
```

Проверка дали масив съдържа желан елемент – **isset()**.

Пример:

```
if (isset ($cars["Volvo"])) echo "OK";
```

Използвани източници:

- **PHP Array tutorial. PHP F1**
  http://www.phpf1.com/tutorial/php-array.html
- **PHP 5 Arrays. w3schools.com**
  http://www.w3schools.com/php/php_arrays.asp
- **Масиви. Ръководство по PHP**
  http://php.net/manual/bg/book.array.php