Масиви. Оператор foreach

Масивите в PHP са асоциативни: те са съвкупност от подредени асоциации - двойки ключ-стойност, за разлика от масивите в другите езици, където са индексирани.
Масив може да бъде създаден посредством езиковата конструкция array(), която приема определено количество двойки ключ => стойност, разделени със запетаи:
`array([ключ=>]) стойност [[,ключ=>] стойност …);`
Ключът може да бъде или цяло число, или низ. Винаги трябва да загражда̀те низовите индекси на масиви с апострофи или кавички. Стойността може да бъде всеки PHP тип.

Пример 1. Дефиниране на масив $a и извеждане с print_r(). Обърнете внимание, че за третия елемент е зададена само стойност, без ключ. Вижте какъв е ключа (индекса) на този елемент (с 1 по-голям от последния целочислен индекс).

```
<?php
$a=array(3=>"Peter", 'b'=>"John","Hary");
print_r($a);?>
```

Резултатът е:

```
Array ([3] => Peter [b] => John [4] => Hary)
```

Вижда се, че ако не укажете изрично ключ за дадена стойност, то ще се вземе най-голямата стойност от целочислените индекси и новият ключ ще бъде тази стойност + 1.
Функция **print_r()** разпечатва всички елементи на масива.
С директива <pre>, тоест при следната вариация на пример1, изходът ще е в следния по-удобен вид:

```
<pre>
<?php
$a=array(3=>"Peter", 'b'=>"John","Hary");
print_r($a);
?>
</pre>
```

Изходът е:

```
Array
(    [3] => Peter
     [b] => John
     [4] => Hary
)
```

Пример 2. Дефиниране на масив $arr и извеждане поелементно с echo. Обърнете внимание на резултата, който виждате.

```
<?php
$arr  =  array("Ina"  =>  "Simeonova",  12  =>  true,  "13"  =>  "Maria",
"012"=>"Dean");
echo $arr["Ina"];        // Simeonova
echo "<br>".$arr[12];    // 1
echo "<br>".$arr["12"];    // 1
echo "<br>".$arr["012"];    // Dean
echo "<br>".$arr["13"];  //Maria
echo "<br>".$arr[13];    // Maria
?>
```

```
$arr = array("Ina" => "Simeonova", 12 => true, "13" => "Maria",
"012"=>"Dean","12" => "Milena");
```

Да видим какъв е изхода, ако допълним масива с още един елемент, например "12" =>"Milena":

```
<?php
$arr = array("Ina" => "Simeonova", 12 => true, "13" => "Maria",
"012"=>"Dean","12" =>"Milena");
print_r($arr);
//Array ( [Ina] => Simeonova [12] => Milena [13] => Maria [012] => Dean )
?>
```

Или ако добавим двойката 13.3=>"Toni":

```
$arr = array("Ina" => "Simeonova", 12 => true, "13" => "Maria",
"012"=>"Dean",13.3=>"Toni");
```

Ще работи ли коректно скрипта във втория случай? Да, всичко е коректно. Масивът ще съдържа следните двойки:

```
[Ina] => Simeonova
[12] => 1
[13] => Toni
[012] => Dean
```

Извод: Ако ключът е представен като обикновено цяло число, той ще бъде интерпретиран като такова (т.е. "12" ще се интерпретира като 12, докато "012" - като "012"). Ако укажете ключ, който вече има присвоена стойност, то тази стойност ще бъде презаписана. Плаващите числа в ключовете се съкращават до цели и ако имаме: 13.3=>"Toni", то това ще се интерпретира като [13] => Toni.
Както казахме, стойността може да бъде всякакъв PHP тип, включително масив (примера по-долу).

Пример 3. Дефиниране на масив $arr, със ключове "Article" и "Price" и стойност за всеки един от ключовете - масив:

```
<?php
$arr = array("Article" => array(1 => "Kiwi", 5 => "Apple", 3 => "Orange"),
"Price"=>array(1=>2.35, 5=>1.35, 3=>1.70));
echo $arr["Article"][1]." - ". $arr["Price"][1];
echo "<br>".$arr["Article"][5]." - ". $arr["Price"][5];
echo "<br>".$arr["Article"][3]." - ". $arr["Price"][3];
?>
```

```
изходът е:
```

Kiwi - 2.35
Apple - 1.35
Orange - 1.7

Отново ще припомним, че ако не укажете изрично ключ за дадена стойност, то ще се вземе най-голямата стойност от целочислените индекси и новият ключ ще бъде тази стойност + 1.
Ако укажете ключ, който вече има присвоена стойност, то тази стойност ще бъде презаписана.
Ако добавяте елемент към масив, в който текущият максимален ключ е отрицателен, следващият създаден ключ ще бъде нула (0).
Използването на TRUE като ключ ще се изчисли като целочислено 1. Употребата на FALSE като ключ ще се изчисли като целочислено 0.

Пример 4.

```
<?php
$arr=array(-5 => 10, 20, 30, 35, 45, 3 => 12, true=>100);
print_r($arr);
//Array ( [-5] => 10 [0] => 20 [1] => 100 [2] => 35 [3] => 12 )
?>
```

Създаване/променяне с квадратни скоби

Можете също да създавате/променяте съществуващ масив чрез изрично установяване на стойностите в него. Това се осъществява чрез присвояване на стойностите в масива, като ключовете се указват в квадратни скоби. Можете да разпечатвате стойности на елементите и чрез оператор за цикъл.

Пример 5:

```
<?php
    $song[0] = 'Tonight will be forever';
    $song[1] = 'I will always love you';
    $song[2] = 'Don\'t turn off the music';

     for ($i = 0; $i < 3; $i++)
        echo $song[$i] . '<br />';
 ?>
```

Резултат:

```
Tonight will be forever
I will always love you
Don't turn off the music
```

Пример 6: Функция sizeof($song) - връща броя на елементите на масива (в долния пример - 4).

```
<?php
    $song[0] = 'Tonight will be forever';
    $song[1] = 'I will always love you';
    $song[2] = 'Don\'t turn off the music';
    $song[] = 'My love';
     for ($i = 0; $i < sizeof($song); $i++)
        echo $song[$i] . '<br />';
?>
```

Изход:

```
Tonight will be forever
I will always love you
Don't turn off the music
My love
```

Пример 7: Вие можете да използвате foreach оператор за обхождане на масива и sort() функция – за сортиране на стойностите на масива.

Оператор foreach е оператор за цикъл в PHP, създаден специално за обхождане на масиви. Има 2 форми:

```
foreach (име на масив as $val)
{ блок инструкции }
```

и

```
foreach (име на масив as $key => $val)
{ блок инструкции }
```

При изпълнение на цикъла, указателят на масива се установява на първия елемент. При всяко изпълнение на „блока от инструкции”, променливата $val получава стойността на поредния елемент на масива, а променливата $key - на ключа.

```
<?php
$toppings=array('pepperoni','mushrooms','sausage','extra cheese');
echo "<br>Before sort";
foreach($toppings as $key=>$info)
{echo "<br>".$key." ".$info.' ';}
sort($toppings);
echo "<br><br>After sort";
foreach($toppings as $info)
    {echo "<br>".$info.' ';}
?>
```

Резултат:

```
Before sort
0 pepperoni
1 mushrooms
2 sausage
3 extra cheese

After sort
extra cheese
mushrooms
pepperoni
sausage
```

Пример 8: Вие можете да използвате asort() функция – за сортиране на стойностите на масива, при запазване на ключовете:

```
<?php
$prices=array('Chicken' => '$5', 'Shrimp'=>'$7', 'Fish'=>'$3');
asort($prices);
foreach ( $prices as $k=>$v)
echo "<br>".$k." has price ".$v;
?>
```

Резултат:

```
Fish has price $3
Chicken has price $5
Shrimp has price $7
```

Ако замените asort() със sort() ще загубите истинските ключове и резултата ще бъде:

```
0 has price $3
1 has price $5
2 has price $7
```

Пример 9: В този пример също не е удачно да използвате sort(), вместо това използвайте asort().

```
<?php
$state['MD']='Maryland';
$state['VA']='Virginia';
$state['NY']='New York';
foreach($state as $key=>$info) {
     echo "$key $info<br />";  //prints the initial array.
}
asort($state);
echo "<br><br>After Sorting:<br><br>";
foreach($state as $key=>$info) {
     echo "$key $info<br />";  //prints the sorted array.
}
?>
```

Резултат:

```
MD Maryland
VA Virginia
NY New York

After Sorting:
MD Maryland
NY New York
VA Virginia
```

Масивът може да се инициализира също и по следния начин:
$state=array('MD'=>'Maryland', 'VA'=>'Virginia', 'NY'=>'New York');

Има доста на брой полезни функции за работа с масиви, които няма да са обект на разглеждане в това упражнение.