Arrays in PHP

Arrays, Associative Arrays, Arrays Operations

Mario Peshev

Technical Trainer

http://peshev.net/

Software University

http://softuni.bg

Table of Contents

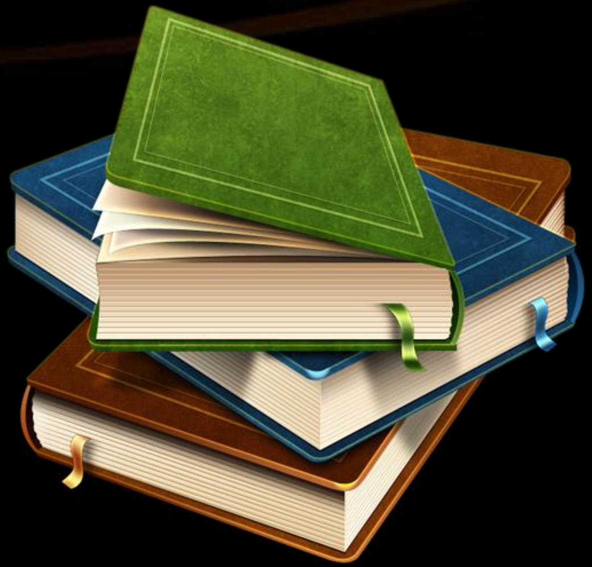
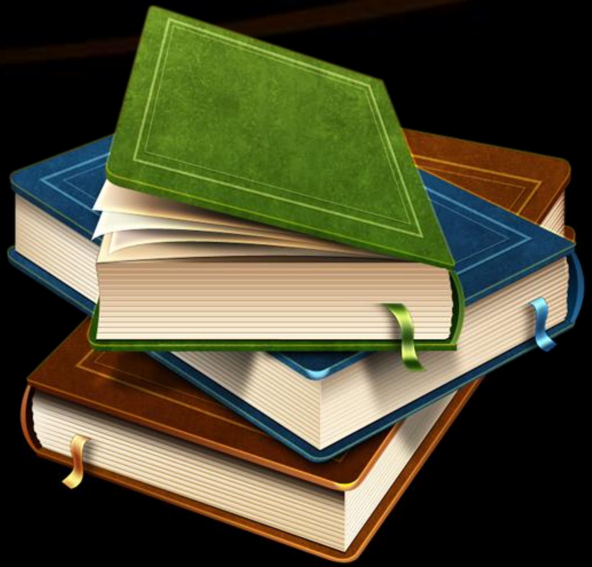


Table of Contents (2)



Declaring Simple PHP Arrays

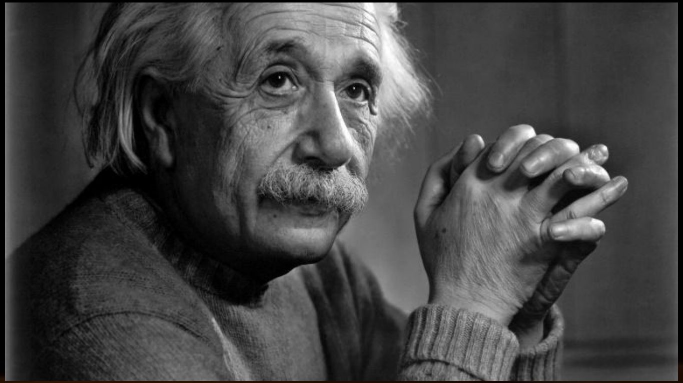
```
$array = array(
        "foo"  => "bar",
        "baz" => "quix"
    );
```

What are Arrays?

- An array in PHP is actually an ordered map. A map is a type that maps values to keys. This type is optimized in several ways, so you can use it as a real array, or a list(vector), hashtable(which is an implementation of a map), dictionary, collection, stack, queue, and probably more. Because you can have another PHP array as a value, you can also quite easily simulate trees.

Array Types

- There are different opinions of whether PHP “really” supports different array types. The book says “yes”, defining the as indexed and Associative Arrays, based on whether the key is numeric or string
- PHP.net says: “There are no different indexed and associative array types in PHP, there is only one array type, which can both contain integer and string indices.



So What's a Key?

- An array key can be thought of as the address or reference to a specific value contained within an array.
- A key can be an integer or a string
  - Floats used as keys are truncated to integers
- If you don't specify a key, the array key will automatically start with 0 (zero) and auto increment by 1 each time a new value is entered into the array.
- Key must be unique

Creating Empty Array

- An array can be created by the array() language-construct.

```
<?php
        $newArray = array();
?>
```

- As of PHP 5.4, can create array with []

```
<?php
        $newArray = [];
?>
```

Creating initialization

- An empty array can be created by the `array()` language-construct.

```
<?php
        $newArray = array();
?>
```

- As of PHP 5.4, can create empty array with `[]`

```
<?php
        $newArray = [];
?>
```

Creating an Array

- An array can be created by the array() language-construct. It takes a number of comma-separated key=>value pairs.

```
<?php

$student = array (
        'firstName' => 'Pesho',
        'lastName' => 'Georgiev',
        'town'         => 'Kaspichan');
?>
```

Value

Key

Accessing Array Elements

- Array elements can be accessed using the array[key] syntax.

```php
<?php
$array = array (
    "foo" => "bar",
     42   => 24);

    echo $array['foo'] . '</br>';
    echo $array[42];
?>
```

Declaring Simple PHP Arrays

Live Demo

```
$array = array(
    "foo" => "bar",
    "baz" => "quix"
);
```

Print Array Structure

- You can quickly view the keys and values of an array using `print_r()`

```
<?php
$array = array (
    "foo" => "bar",
     42   => 24);

    print_r($array);
?>
```

```
Array
(
    [foo] => bar
    [42] => 24
)
```

Even More Detail

- You can quickly view structure, keys and values of an array using `var_dump()`

```php
<?php
$array = array (
    "foo" => "bar",
    42   => 24);

var_dump($array);
?>
```

```
array(2) {
  ["foo"]=>
  string(3) "bar"
  [42]=>
  int(24)
}
```

Add Items

- After an array has been created, you can add values to the ‘end’ of the array

```php
<?php
$array = array(
    0 => "green",
    1 => "yellow",
    2 => 'blue');

$array[] = 'pur
print_r($array);
?>
```

```
Array
(
    [0] => green
    [1] => yellow
    [2] => blue
    [3] => purple
)
```

Delete Array Element

- After an array has been created, you can add values to the 'end' of the array:

```php
<?php
$array = array(
    0 => "green",
    1 => "yellow",
    2 => 'blue');

unset($array[1]);
print_r($array);
?>
```

```
Array
(
    [0] => green
    [2] => blue
)
```

Array size

- You can count the number of values in an array:

```php
<?php
$array = array(
    "green",    "yellow",    'blue', "purple",    "grey");

echo count($array); // output 5
?>
```

Delete or Reset Array

- You can delete all array:
  - `unset($array);`
- If a globalized variable is `unset()` inside of a function, only the local variable is destroyed. The variable in the calling environment will retain the same value as before unset() was called.
- Reset Array
  - `$array = array();`

Add and Remove

Live Demo

```
$array = array(
        "foo" => "bar",
        "baz" => "quix"
      );
```

Looping Array

Looping through an array

- Use for loop to process an array when
  - Need to keep track of the index
  - Processing is not strictly sequential from the first to the last element
- In the loop body use the element at the loop index (`$array[index]`):
- `for()` loop doesn't work on arrays with string indexes

Looping through an array Example

```
<?php
$array = array(
    "green",    "yellow",    'blue', "purpl                y");

for($i = 0; $i < count($array); $i++){
   echo $array[$i] . '</
}
?>
```

```
green
yellow
blue
purple
grey
```

Looping through an associative array

- There is a quick and easy way to accomplish the same thing: a `foreach` loop, which itself has two versions. The easiest way to use `foreach` looks like this:

```
<?php
$array = array(
    "green",    "yellow",    'blue', "purpl          ey");

foreach($array as $val) {
    print $val."\n";
}
?>
```

```
green
yellow
blue
purple
grey
```

Looping through an associative array {2}

- Here the array `$array` is looped through and its values are extracted into `$val`. In this situation, the array keys are ignored completely, which usually makes most sense when they have been auto-generated (i.e. 0, 1, 2, 3, etc).

```
<?php
$array = array(
    "green",    "yellow",    'blue', "purpl            ");

foreach ($array as $key => $val) {
    print "$key = $val
}
?>
```

1 = yellow
2 = blue
3 = purple
4 = grey

Looping Array

Live Demo

Build in Array Functions

Merge Arrays

- Merges the elements of one or more arrays together so that the values of one are appended to the end of the previous one. It returns the resulting array

```php
<?php
$array = array(
    "green",    "yellow",    'blue');

$other = array(
    "purple",    "grey");

$allTogether = array_merge($array, $
print_r($allTogether);
?>
```

```
Array
(
    [0] => green
    [1] => yellow
    [2] => blue
    [3] => purple
    [4] => grey
)
```

Join Arrays

- You can take colors and transform them into a string with `implode()` at the end of each color:

```php
<?php
$array = array(
    "green",     "yellow",     'blue', "purple",     "grey");
$newString = implode(", ", $array);

echo $newString;
?>
```

green, yellow, blue, purple, grey

Sorting Arrays

- Arrays can be sorted by key or value
- You can sort and keep keys aligned, or sort values and have new keys assigned.
- `sort()` is alphabetical if values are string or numeric if values are number.

```php
<?php
$array = array(
    "green",    "yellow",    'blue', "purple",

sort($array);
print_r($array);
?>
```

```
Array
(
    [0] => blue
    [1] => green
    [2] => grey
    [3] => purple
    [4] => yellow
)
```

Reverse Sort

- `rsort()` sorts in reverse:

```
<?php
$array = array(
    "green",     "yellow",     'blue', "purple",     "grey");

rsort($array);
print_r( $array);
?>
```

```
Array
(
    [0] => yellow
    [1] => purple
    [2] => grey
    [3] => green
    [4] => blue
)
```

Key sort

- ksort() sorts keys, keeping values correlated to keys:

```php
<?php
$array = array(
    "green",    "yellow",    'blue', "purple",    "grey");

ksort($array);
print_r( $array);
?>
```

```
Array
(
    [0] => green
    [1] => yellow
    [2] => blue
    [3] => purple
    [4] => grey
)
```

Shuffle array

- `shuffle()` randomly reorganizes the order of the array:

```
<?php
$array = array(
    "green",    "yellow",    'blue', "purple",    "grey");

shuffle($array);
print_r( $array);
?>
```

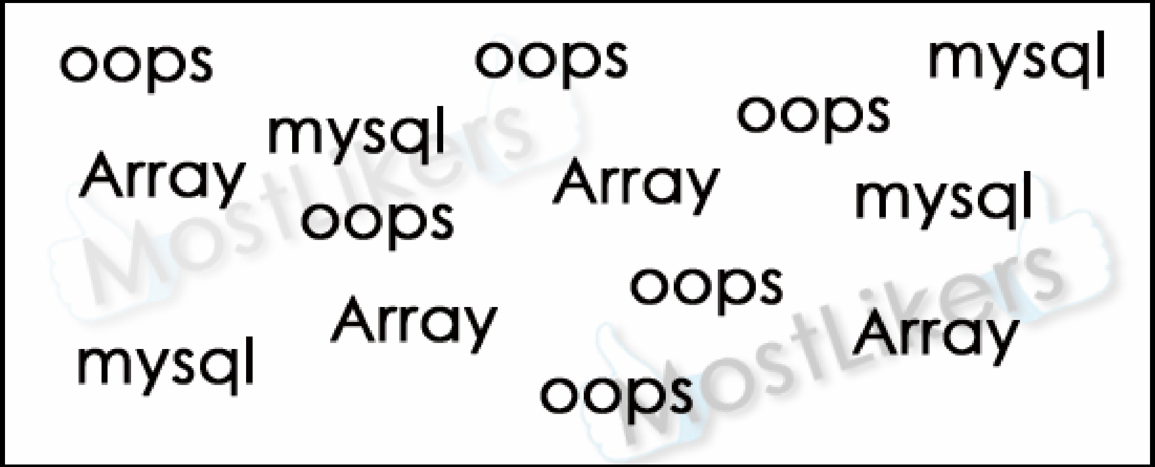
Extract Array

- Use `extract()` to automatically create variables from an array, where the variables created are the keys and their values are the associated array values:

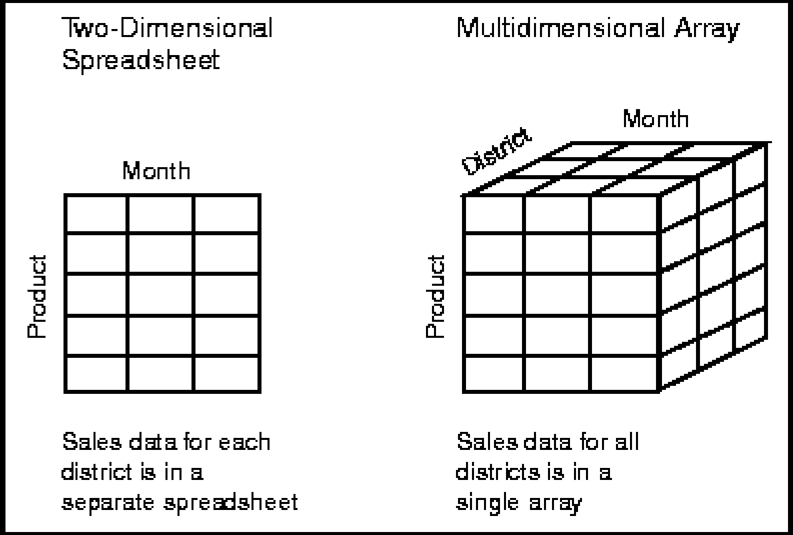
```
<?php
$array = array(
    'color1' => "green",
    'color2' =>   "yellow",
    'color3' =>    'blue',);

extract($array);

echo $color1; //Output green
echo $color2;//Output yellow
echo $color3; //Output 'blue
?>
```

Build in Array Functions

Live Demo



Multidimensional arrays



Two-Dimensional Spreadsheet
Month
Product
Sales data for each district is in a separate spreadsheet
Multidimensional Array
District
Month
Product
Sales data for all districts is in a single array

Multidimensional Arrays

- A multidimensional array is an array containing one or more arrays.
- The dimension of an array indicates the number of indices you need to select an element.
  - For a two-dimensional array you need two indices to select an element
  - For a three-dimensional array you need three indices to select an element

Multidimensional Arrays Example

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

?>
```

Multidimensional arrays

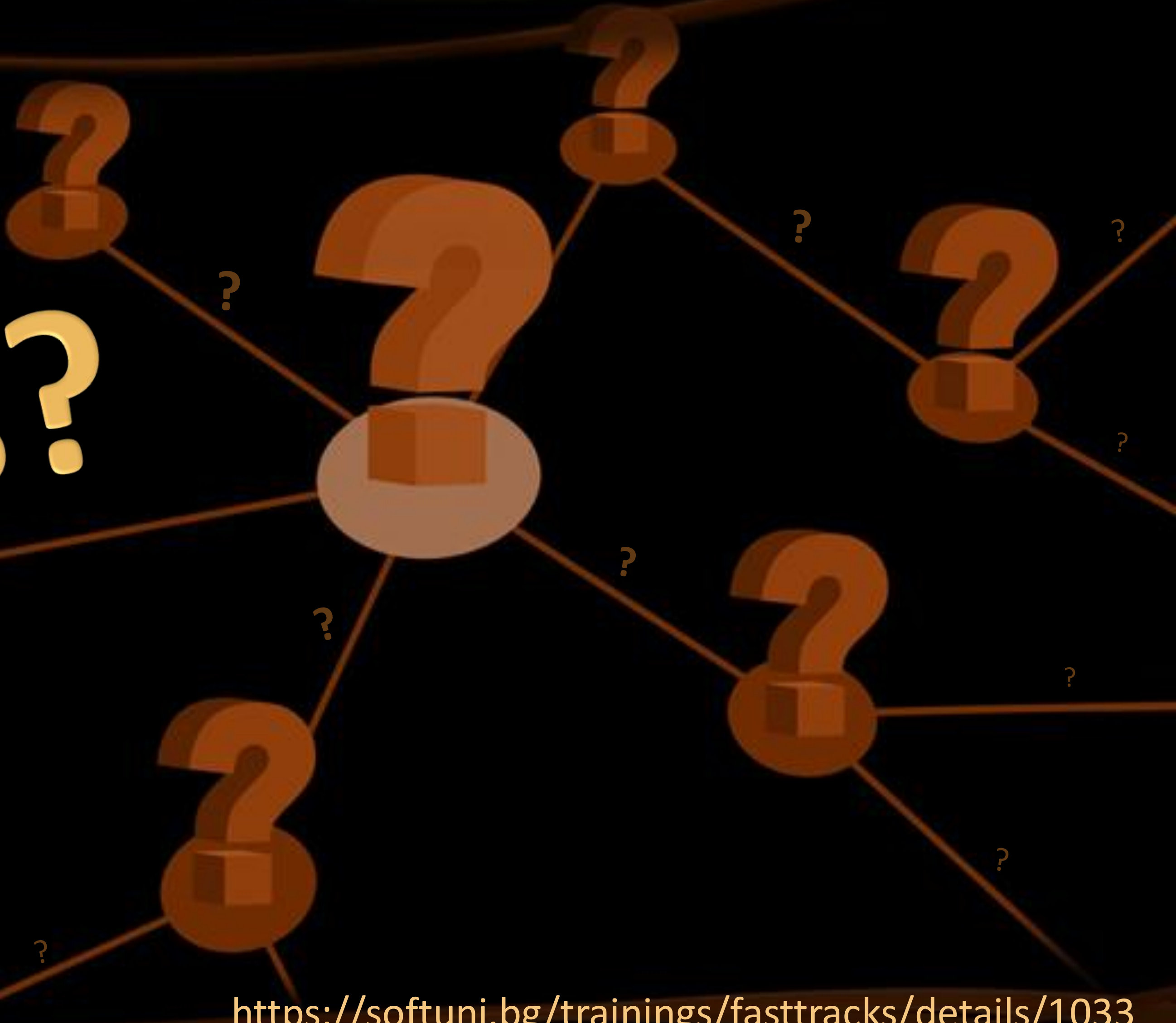
Live Demo

Summary

- Creating array
- Looping arrays
  - For, Foreach
- Build in array functions
  - sort(), ksort() shuffle(), merge, array_merge()
- Multidimensional arrays

PHP & MySQL

SOFTWARE UNIVERSITY FOUNDATION



Questions?

https://softuni.bg/trainings/fasttracks/details/1033

License

Free Trainings @ Software University

- Software University Foundation – softuni.org
- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University @ YouTube
  - youtube.com/SoftwareUniversity
- Software University Forums – forum.softuni.bg