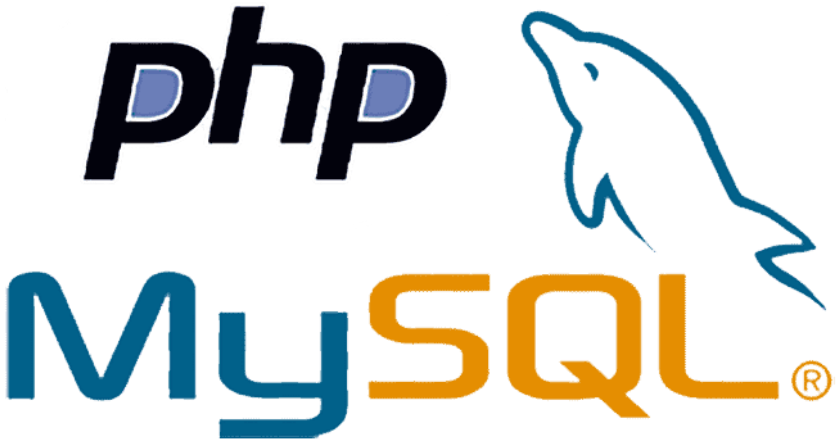
2022 - 2023



Arrays in PHP

Eng. Hazem A. Alrakhawi

What is Array?

Eng. Hazem A. Alrakhawi

What is an Array?

- An array is a special variable, which **can hold more than one value** at a time.
- If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
$cars1 = "Volvo";
$cars2 = "BMW";
$cars3 = "Toyota";
```

What is an Array?

- However, what **if you want to loop through the cars** and find a specific one? And **what if you had not 3 cars, but 300?**
- **Solution**: An array can hold many values under a single name, and you can access the values by referring to an index number.
- **In PHP**, the **array() function** is used to create an array:

```php
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

What is an Array?

- The **count() function** is used to return the length (the number of elements) of an array:

```php
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo count($cars);
?>
```

Array Types in PHP

Eng. Hazem A. Alrakhawi

Array types in PHP

In PHP, there are three types of arrays:

- **Indexed arrays** - Arrays with a numeric index.
- **Associative arrays** - Arrays with named keys.
- **Multidimensional arrays** - Arrays containing one or more arrays.

PHP Indexed Arrays

Eng. Hazem A. Alrakhawi

PHP Indexed Arrays

There are two ways to create indexed arrays:

- The index can be assigned automatically (**index always starts at 0**), like this:

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

- Or the index can be assigned manually:

```
$cars[0] = "Volvo";
$cars[1] = "BMW";
$cars[2] = "Toyota";
```

PHP Indexed Arrays

The following example **creates an indexed array named $cars, assigns three elements to it**, and then prints a text containing the array values:

```php
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

Loop Through an Indexed Array

To loop through and **print all the values of an indexed array**, you could use a for loop, like this:

```php
<?php
$cars = array("Volvo", "BMW", "Toyota");
$arrlength = count($cars);

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

PHP Associative Arrays

Eng. Hazem A. Alrakhawi

PHP Associative Arrays

Associative arrays are arrays that **use named keys** that you assign to them.

There are two ways to create an associative array:

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

- Or another way:

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

PHP Associative Arrays

The named keys can then be used in a script:

```php
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>
```

Loop Through an Associative Array

To loop through and print all the values of an associative array, **you could use a foreach loop**, like this:

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

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

PHP Multidimensional Arrays

Eng. Hazem A. Alrakhawi

PHP Multidimensional Arrays

- A multidimensional array **is an array containing one or more arrays.**
- PHP supports multidimensional arrays that are two, three, four, five, or more levels deep.
- However, arrays more than three levels deep are hard to manage for most people.

PHP - Two-dimensional Arrays

- A two-dimensional array is an array of arrays (a three-dimensional array is an array of arrays of arrays).
- **First, take a look at the following table:**

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

PHP - Two-dimensional Arrays

- We can store the data from the table above in a two-dimensional array, like this:

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

PHP - Two-dimensional Arrays

- Now the two-dimensional $cars array contains four arrays, and it has two indices: **row and column**.
- To get access to the elements of the $cars array **we must point to the two indices (row and column):**

```php
<?php
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>";
?>
```

PHP - Two-dimensional Arrays

- We can also put a for **loop inside another for loop to get the elements of the $cars array** (we still have to point to the two indices):

```php
<?php
for ($row = 0; $row < 4; $row++) {
  echo "<p><b>Row number $row</b></p>";
  echo "<ul>";
  for ($col = 0; $col < 3; $col++) {
    echo "<li>".$cars[$row][$col]."</li>";
  }
  echo "</ul>";
}
?>
```

PHP Sorting Arrays

Eng. Hazem A. Alrakhawi

PHP Sorting Arrays

- The elements in an array **can be sorted** in **alphabetical** or **numerical** order, **descending** or **ascending**.

PHP - Sort Functions For Arrays

- **We will go through the following PHP array sort functions:**
- **sort()** - sort arrays in ascending order.
- **rsort()** - sort arrays in descending order.
- **asort()** - sort associative arrays in ascending order, according to the value.

PHP - Sort Functions For Arrays

- **ksort()** - sort associative arrays in ascending order, according to the key.
- **arsort()** - sort associative arrays in descending order, according to the value.
- **krsort()** - sort associative arrays in descending order, according to the key.

Sort Array in Ascending Order - sort()

- **The following example sorts the elements of the $cars array in ascending alphabetical order:**

```php
<?php
$cars = array("Volvo", "BMW", "Toyota");
sort($cars);

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

BMW

Toyota

Volvo

Sort Array in Ascending Order - sort()

- **The following example sorts the elements of the $numbers array in ascending numerical order:**

```php
<?php
$numbers = array(4, 6, 2, 22, 11);
sort($numbers);

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

```
2
4
6
11
22
```

Sort Array in Descending Order - rsort()

- **The following example sorts the elements of the $cars array in descending alphabetical order:**

```php
<?php
$cars = array("Volvo", "BMW", "Toyota");
rsort($cars);

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

Volvo
Toyota
BMW

Sort Array in Descending Order - rsort()

- **The following example sorts the elements of the $numbers array in descending numerical order:**

```php
<?php
$numbers = array(4, 6, 2, 22, 11);
rsort($numbers);

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

```
22
11
6
4
2
```

Sort Array (Ascending Order), According to Value - asort()

- **The following example sorts an associative array in ascending order, according to the value:**

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

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

Key=Peter, Value=35

Key=Ben, Value=37

Key=Joe, Value=43

Sort Array (Ascending Order), According to Key - ksort()

- **The following example sorts an associative array in ascending order, according to the key:**

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

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

Key=Ben, Value=37
Key=Joe, Value=43
Key=Peter, Value=35

Sort Array (Descending Order), According to Value - arsort()

- **The following example sorts an associative array in descending order, according to the value:**

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

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

Key=Joe, Value=43
Key=Ben, Value=37
Key=Peter, Value=35

Sort Array (Descending Order), According to Key - krsort()

- **The following example sorts an associative array in descending order, according to the key:**

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

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

Key=Peter, Value=35

Key=Joe, Value=43

Key=Ben, Value=37

PHP Array Functions

Eng. Hazem A. Alrakhawi

PHP Array Functions – Add

- **The array_push()** function push one or more elements onto the end of array.
- **Example: array_push($array, 8, 9);**
- **The array_unshift()** function inserts new elements to an array at the beginning of the array.
- **Example: array_unshift($numbers, 11, 12);**

PHP Array Functions – Search

- **in_array(value, array, type)**
- **Value :** **Required**. Specifies the what to search for.
- **Array: Required.** Specifies the array to search.
- **Type: Optional.** The default is false, If this parameter is set to TRUE, the in_array() function **searches for the search-string and specific type** in the array.

PHP Array Functions – Search

- **Example: in_array(value, array, type)**

```php
<?php
$arr = array(1,2,3,"10");

//10 and "10" the same when false
var_dump(in_array(10,$arr)); //return true

echo "<br>";

//10 and "10" not the same when true
var_dump(in_array(10,$arr,true)); //return false
?>
```

PHP Array Functions – Search

- **array_search(value, array, strict)**
- **Value :** **Required**. Specifies the what to search for.
- **Array: Required.** Specifies the array to search.
- **Strict: Optional.** The default is false, If this parameter is set to TRUE, the in_array() function **searches for the search-string and specific type** in the array.

PHP Array Functions – Search

- **Example: array_search(value, array, strict)**

```php
<?php
$arr = array(1,2,3,"10");

var_dump(array_search("10",$arr)); //int(3)

echo "<br>";

var_dump(array_search(10,$arr,true)); //bool(false)

?>
```

PHP Array Functions – Search

- **The array_key_exists()** function checks an array for a specified key, and returns true if the key exists and false if the key does not exist.
- **array_key_exists(key, array)**
- **Key: Required.** Specifies the key.
- **array : Required**. Specifies an array.

PHP Array Functions – Search

- **Example: array_key_exists(key, array)**

```php
<?php
$arr = array(1,2,"10",4);

var_dump(array_key_exists(3,$arr)); //return true

$langs = array(
    "H" => "HTML",
    "C" => "CSS",
    "J" => "JS"
);

var_dump(array_key_exists("C",$langs)); //return true

?>
```

PHP Array Functions – Delete

- **The array_pop**() function deletes the last element of an array and returns the value of the removed element.
- **array_pop(array)**
- **Array: Required.** Specifies an array.

PHP Array Functions – Delete

- **The array_shift()** function removes the first element from an array, and returns the value of the removed element.
- **array_shift(array)**
- **array** : **Required**. Specifies an array
- **Note:** If the keys are numeric, all elements will get new keys, starting from 0 and increases by 1.

PHP Array Functions – Delete

- **Example: array_pop(array) and array_shift(array)**

```php
<?php
$arr = array(1,2,"10",4);

$lastvaluedeleted = array_pop($arr); //delete the last value in array
echo $lastvaluedeleted;

$langs = array(
    "H" => "HTML",
    "C" => "CSS",
    "J" => "JS"
);
$firstvaluedeleted = array_shift($langs); //delete the first value in array
echo $firstvaluedeleted;

?>
```

PHP Array Functions – unset()

- **Example: unset(value)**
- **For delete specific value by array index or by array key.**

```php
<?php
$arr = array(1,2,"10",4);

unset($arr[0]); //delete value by index
print_r($arr);

echo "<br>";

$langs = array(
    "H" => "HTML",
    "C" => "CSS",
    "J" => "JS"
);
unset($langs["C"]);  //delete value by key
print_r($langs);

?>
```

PHP explode() Function

- The **explode()** function splits a string into an array.
- **explode(separator,string,limit)**
- **Separator Required**. Specifies where to break the string
- **String Required**. The string to split
- **Limit Optional**. Specifies the number of array elements to return.

PHP explode() Function

- The "**separator**" parameter **cannot be an empty string** and the function **return bool(false).**

```php
<?php
$str = "Hello world. It's a beautiful day.";
$array = explode(" ",$str);
print_r($array);
echo "<br>";
echo count($array);
?>
```

PHP explode() Function limit

Possible values of limit:

- **Greater than 0** - Returns an array with a maximum of limit element(s)
- **Less than 0** - Returns an array except for the last -limit elements()
- **0** - Returns an array with one element

PHP explode() Function limit

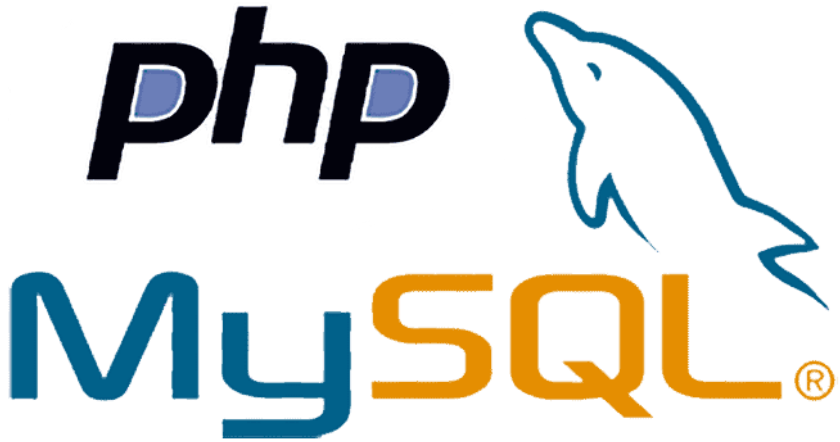
```php
<?php
$str = 'one,two,three,four';

// zero limit
print_r(explode(',',$str,0));
print "<br>";

// positive limit
print_r(explode(',',$str,2));
print "<br>";

// negative limit
print_r(explode(',',$str,-1));
?>
```

```
Array ( [0] => one,two,three,four )
Array ( [0] => one [1] => two,three,four )
Array ( [0] => one [1] => two [2] => three )
```

Another PHP Array Functions

- **Array_fill(startIndex,count,value)** Fill an array with values.
- **Array_reverse($array,true)** returns array in reverse order.
- **Array_unique($array)** removes duplicate values from an array.
- **Array_sum($array)** returns the sum of values in an array.
- **Array_rand($array,number)** returns number of random key

2022 - 2023



The End.

Eng. Hazem A. Alrakhawi