Course:
PHP from scratch

by Sergey Podgornyy

Arrays and Loops

WEB ACADEMY

PROGRAMMING COURSES

About me

Sergey Podgornyy

Full-Stack Web Developer

WebbyLab

and

ignite software outsourcing

zend CERTIFIED PHP ENGINEER

Linux Professional Institute LPIC-1

Overview

- Arrays
- Indexed arrays, Associative arrays and Multidimensional arrays
- Outputting arrays
- Sorting arrays
- Main array functions
- Superglobal arrays
- Loops: for, foreach, while, do…while
- Constructions break, continue, die, exit

Arrays

- An array stores multiple values in one single variable:

```php
<?php

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

```

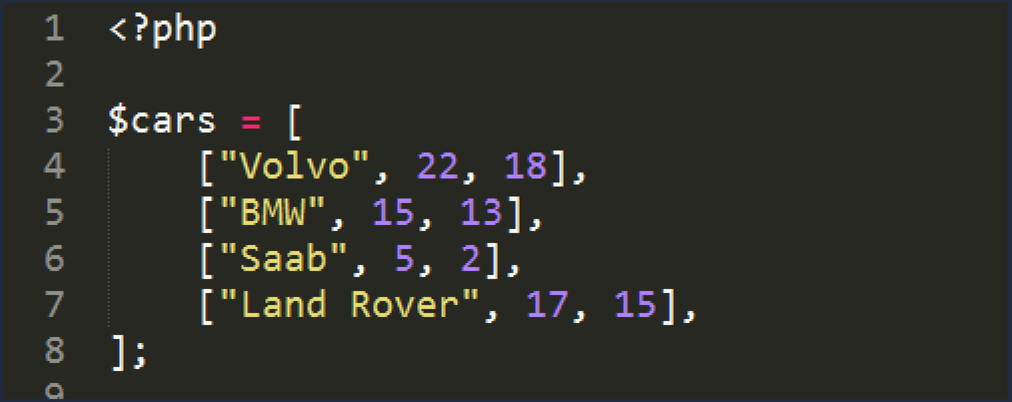
- An array is a special variable, which can hold more than one value at a time
- In PHP, there are three types of arrays:
  - Indexed arrays - Arrays with a numeric index
  - Associative arrays - Arrays with named keys
  - Multidimensional arrays - Arrays containing one or more arrays

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



```php
<?php

$cars = [
	["Volvo", 22, 18],
	["BMW", 15, 13],
	["Saab", 5, 2],
	["Land Rover", 17, 15],
];
```

Outputting arrays

- `print_r` — prints human-readable information about a variable
- `var_dump` — dumps information about a variable. This function displays structured information about one or more expressions that includes its type and value

Sorting Arrays

Sorting function attributes

Function name	Sorts by	Maintains key association	Order of sort	Related functions
array_multisort()	value	associative yes, numeric no	first array or sort options	array_walk()
asort()	value	yes	low to high	arsort()
arsort()	value	yes	high to low	asort()
krsort()	key	yes	high to low	ksort()
ksort()	key	yes	low to high	asort()
natcasesort()	value	yes	natural, case insensitive	natsort()
natsort()	value	yes	natural	natcasesort()
rsort()	value	no	high to low	sort()
shuffle()	value	no	random	array_rand()
sort()	value	no	low to high	rsort()
uasort()	value	yes	user defined	uksort()
uksort()	key	yes	user defined	uasort()
usort()	value	no	user defined	uasort()

Sorting Arrays

- **`sort()`** - sort arrays in ascending order
- **`rsort()`** - sort arrays in descending order
- **`asort()`** - sort associative arrays in ascending order, according to the value
- **`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

Array functions

Stack

- `array_push()` - Inserts one or more elements to the end of an array
- `array_pop()` - Deletes the last element of an array

Queue

- `array_unshift()` - Adds one or more elements to the beginning of an array
- `array_shift()` - Removes the first element from an array, and returns the value of the removed element

Array functions

- `array_merge()` - Merges one or more arrays into one array
- `array_key_exists()` - Checks if the specified key exists in the array
- `count()` - Returns the number of elements in an array
- `in_array()` - Checks if a specified value exists in an array
- `list()` - Assigns variables as if they were an array

Global Variables - Superglobals

Several predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special

The PHP superglobal variables are:

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

Global Variables - Superglobals

- **$GLOBALS** is a PHP super global variable which is used to access global variables from anywhere in the PHP script
- **$_SERVER** is a PHP super global variable which holds information about headers, paths, and script locations
- **$_REQUEST** is used to collect data after submitting an HTML form
- **$_POST** is widely used to collect form data after submitting an HTML form with method="post"
- **$_GET** can also be used to collect form data after submitting an HTML form with method="get"

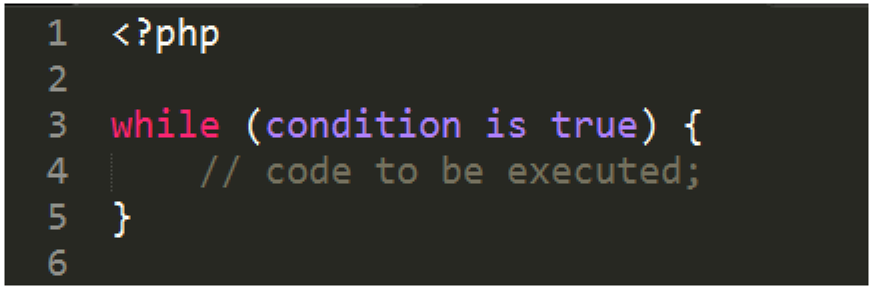
Loops

In PHP, we have the following looping statements:

- while - loops through a block of code as long as the specified condition is true
- do...while - loops through a block of code once, and then repeats the loop as long as the specified condition is true
- for - loops through a block of code a specified number of times
- foreach - loops through a block of code for each element in an array

while loop

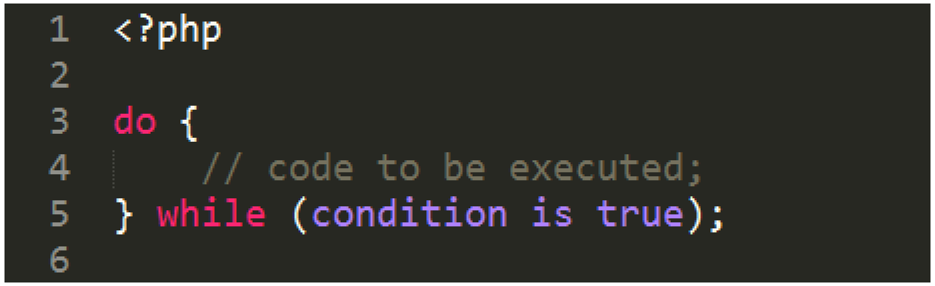
The while loop executes a block of code as long as the specified condition is true



```
<?php

while (condition is true) {
    // code to be executed;
}

```

do...while loop

The do...while loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true

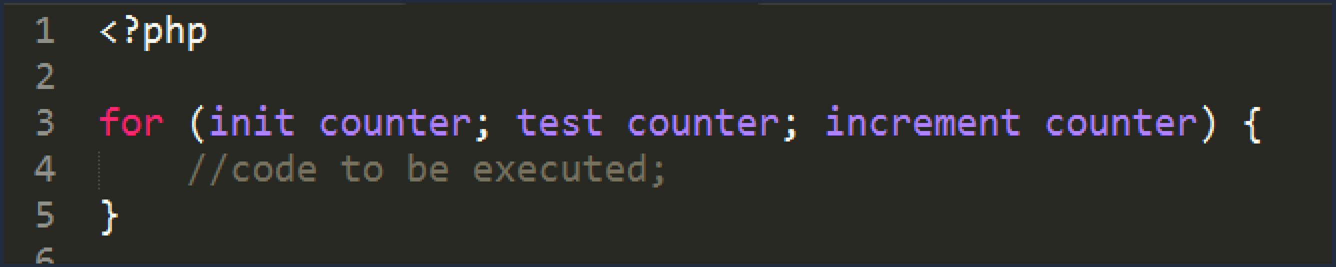


```php
<?php

do {
	// code to be executed;
} while (condition is true);

```

www.web-academy.com.ua

for loop

The for loop is used when you know in advance how many times the script should run

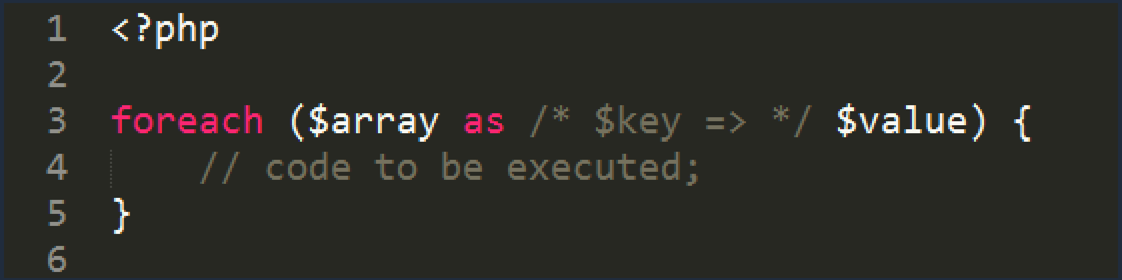


```php
<?php

for (init counter; test counter; increment counter) {
    //code to be executed;
}
```

- init counter: Initialize the loop counter value
- test counter: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends
- increment counter: Increases the loop counter value

`foreach` loop

The foreach loop works only on arrays, and is used to loop through each key/value pair in an array



```php
<?php

foreach ($array as /* $key => */ $value) {
    // code to be executed;
}

```

- For every loop iteration, the value of the current array element is assigned to $value and the array pointer is moved by one, until it reaches the last array element

Alternate "colon syntax"

PHP also supports the alternate "colon syntax" for **`for`** and **`foreach`** loops

```
<?php for ($i = 0; $i < 10; $i++) : ?>

    <p>Iteration #<?= $i+1 ?></p>

<?php endfor; ?>
```

```
<?php foreach ($users as $user) : ?>

    <div>
        <h3><?= $user['full_name'] ?></h3>
        <img src="<?= $user['avatar'] ?>">
        <p class="description">
            <?= $user['about'] ?>
        </p>
    </div>

<?php endforeach; ?>
```

break and **continue**

break ends execution of the current **for**, **foreach**, **while**, **do…while** or **switch** structure

```
$arr = ['one', 'two', 'stop', 'three'];

while (list(, $val) = each($arr)) {
    if ($val == 'stop') {
        break; // You could also write 'break 1;' here
    }
    echo "$val<br>\n";
}
```

continue is used within looping structures to skip the rest of the current loop iteration and continue execution of the next iteration

```
while (list($key, $value) = each($arr)) {
    if ($key % 2) { // skip odd members
        continue;
    }
    $even[] = $value;
}
```

Useful resources

- PHP array functions
- Learn PHP array functions
- Control Structures

Thanks for your attention

Q & A

www.web-academy.com.ua