PHP

Paolo Bettelini

Contents

1 Variable types and constants

1.1 Types

- String
- Integer
- Float
- Boolean
- Array
- Object
- NULL
- Resource

1.2 Examples

```
<?php
    $x = 41976;
    $y = 'Hello world!';
    $xx = "Hello world!";
    $z = 10.365;
    $cars = array("Volvo","BMW","Toyota");
    $n = null;
    $f = false;

    var_dump($x); // prints to the stdout variable type and content
?>
```

1.3 Constants

The syntax to create a contant value is as follows

```
define(name, value, case-insensitive);
```

```
<?php
    define("GREETING", "Welcome to W3Schools.com!", true);
    echo greeting;
?>
```

2 Comparisons

2.1 Operator

- == compares two values → 2.0 == 2 (true)
- === compares two values && data type → 2.0 === 2 (false)

2.2 String comparisons

the **strcmp** function returns 0 if the two strings are equal. It return another number if they are different. It performs a calculation with the ASCII values of all characters in the strings.
For example **strcmp("a", "b") == -1** since the ASCII values of $A - B = -1$
Or **strcmp ("aa", "ab") == -256**, in this case the additional letters have a weight of 2^8 and so on.

```php
<?php
    $var1 = "Hello";
    $var2 = "HEllo";
    if (strcmp($var1, $var2) == 0) {
        echo '$var1 is equal to $var2 in a case-sensitive string comparison';
    } elseif (strcasecmp($var1, $var2) == 0) {
        echo '$var1 is not equal to $var2 in a case insensitive string comparison';
    }
?>
```

3 Control flow

3.1 Conditions

```php
<?php
    $t = date("H");

    if ($t < "10") {
      echo "Have a good morning!";
    } elseif ($t < "20") {
      echo "Have a good day!";
    } else {
      echo "Have a good night!";
    }
?>
```

3.2 Loops

```php
<?php
    while (condition is true) {
      code to be executed;
    }

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

    for ($x = 0; $x < 10; $x++) {
      code to be executed; // 10 times
    }

    foreach ($array as $value) {
      code to be executed;
    }
?>
```

3.3 Switches

```php
<?php
    switch (n) {
      case label1:
        code to be executed if n=label1;
        break;
      case label2:
        code to be executed if n=label2;
        break;
      case label3:
        code to be executed if n=label3;
        break;
        ...
      default:
        code to be executed if n is different from all labels;
    }
?>
```

4 Require and include

4.1 Definition

You can use commands to import code from another file **require** and **include**. They both serve the same purpose, however there is a difference in behavior in case of error:

- **require** FATAL ERROR → execution stops
- **include** WARNING → execution continues

so you have to use *require* when the file is essential.
The syntax is as follows:

```
<?php
    require 'filename';
    include 'filename';
?>
```

4.2 Require Once

The require only imports itself once. If the file already exists, the file is not imported again.
Importing a file multiple times will execute its code. This ensures that you do not override functions (which cannot be redeclared).

```
<?php
    require_once('file.php');
    include_once('file.php');
?>
```

The *require* should be used when the file code needs to be rerun.

5 Namespaces

Namespace are like packages for files.

File1:

```
<?php namespace foo;
    class Cat {
        static function says() {echo 'meoow';}
    }
?>
```

or to simplify their use in classes.
File2:

```
<?php namespace main
  include 'file1.php';
  use foo\Cat as feline
  echo feline::says(), "<br />\n";
?>
```

6 Functions

6.1 Definition

```
<?php
    function functionName() {
      do things;
    }

    functionName(); // call the function
?>
```

6.2 Parameters

6.2.1 Mandatory parameters

```
<?php
    function greetName($fname) {
      echo "hello $fname.<br>";
    }

    greetName("Giecinz");
    greetName("Paola");
?>
```

6.2.2 Optional parameters

```
<?php
    function setHeight(int $minheight = 50) {
      echo "The height is : $minheight <br>";
    }
    setHeight(350);
    setHeight(); // default value = 50
?>
```

6.2.3 Type declaration

The type declaration is used to force a parameter type, return value, or property of a class.

If the type is not respected, a **TypeError** is raised.

```
<?php
  function test1(boolean $param) {}
  test1(true);

  function test2(): int {
    return 1;
  }
?>
```

6.2.4 Return

```
<?php
    function sum(int $x, int $y) {
        return $x + $y;
    }
    echo "5 + 10 = " . sum(5, 10) . "<br>";
?>
```

6.2.5 Explicit return

```
<?php
    function sum(int $x, int $y) : int {
      return $x + $y;
    }
?>
```

7 Scopes and global variables

- **local** a variable declared inside a function etc.
- **global** a variable declared with the keyword global or outside a function.
- **static** a static variable.

```
<?php
    $x = 5;
    $y = 10;

    function myTest() {
      global $x, $y;
      $y = $x + $y;
    }

    class Foo {
      static $my_var = 'Foo'; // static var
    }

    myTest();
    echo $y; // outputs 15
?>
```

8 Variables

8.1 Strings

8.1.1 Multi-line string

```
print <<< END
    <p style="background-color: yellow">
    Four score and seven years ago<br/>
    our fathers set onto this continent<br/>
    (and so on ...)<br/>
    </p>
END;
```

8.1.2 Concatenation

```
$a = "str1" . "str2";
$a .= "str3";
```

8.1.3 End of line

The constant **PHP_EOL** represents a new line.

8.1.4 Some functions

```
echo strlen(" ciao");
echo strlen(ltrim(" ciao"));
echo strtoupper("Anghilotto");
echo strtolower("AngHiLotto");
echo strcmp("a", "a") . "<br>";
echo substr("ciao", 2, 1);
echo str_replace("i", "a", "ciao");
```

8.2 Casting

```
echo (int)(9.2);
echo (int)("text"); // returns 0
```

8.3 Nested variables

You can allocate a variable whose name is the value of another variable.

```
$my_var= 'variablename';
$$my_var = 'Some text';
echo $variablename . "<br><br>";
```

8.4 Dates

```php
// current date
echo date("Y/m/d");
echo date("Y.m.d");
echo date("Y-m-d");

$date = new DateTime('now');
echo $date->format('Y-m-d H:i:s');

$diff = date_diff(date_create('2003-05-28'), date_create('2022-05-28));
```

8.5 Arrays

8.5.1 Declaration

```php
// array
$cars = array("Volvo", "BMW", "Toyota");
$cars[0] = "Panda";

$arr = array();
array_push($colors,"blue","yellow");

// associative array
$age = ["Peter"=>"35", "Ben"=>"37", "Joe"=>"43"];
```

8.5.2 Iterating

```php
$colors = array("red", "green", "blue", "yellow");
foreach ($colors as $value) {
    echo "$value <br>";
}
```

8.5.3 Some functions

```php
$arr = array('Hello','World!','Beautiful','Day!');

$str = implode(" ", $arr);
$c = count($arr);
array_reverse($arr$)

$arr = explode(" ", $str$);

sort($arr$); // sort array

// Associative arrays
asort($girl); // sort in ascending order according to value
ksort($girl); // sort in ascending order according to key
arsort($girl); // sort in descending order according to value
krsort($girl); // sort in descending order according to key
```

9 Files

9.1 Read and write

```
$content = file_get_contents($inFile);
file_put_contents($outFile, $result);
```

9.2 Csv

```
function writeToCsv(string $out, $values) {
    $fp = fopen($out, 'w');
    foreach ($values as $key => $val) {
        fputcsv($fp, [$key,$val], ";");
    }
    fclose($fp);
}
```

```
if ($handle = fopen("parole.csv", "r")) {
    while ($data = fgetcsv($handle, 1000, ";")) {
        // ...
    }

    fclose($handle);
}

$fp = fopen($out, 'w');
foreach ($values as $key => $val) {
    fputcsv($fp, [$key,$val], ";");
}
fclose($fp);
```

10 Requests

HTML form

```
<form action=file."php method"="POST>
<input type"="text name=field1>
<input type"="submit>
</form>
```

Check page request type

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    // POST
} else {
    // GET
}
```

All the request-variables are in the **$_POST** array or **$_GET** array.

```
if(isset($_POST['field1'])) {
    $a = $_POST['field1']
}
```