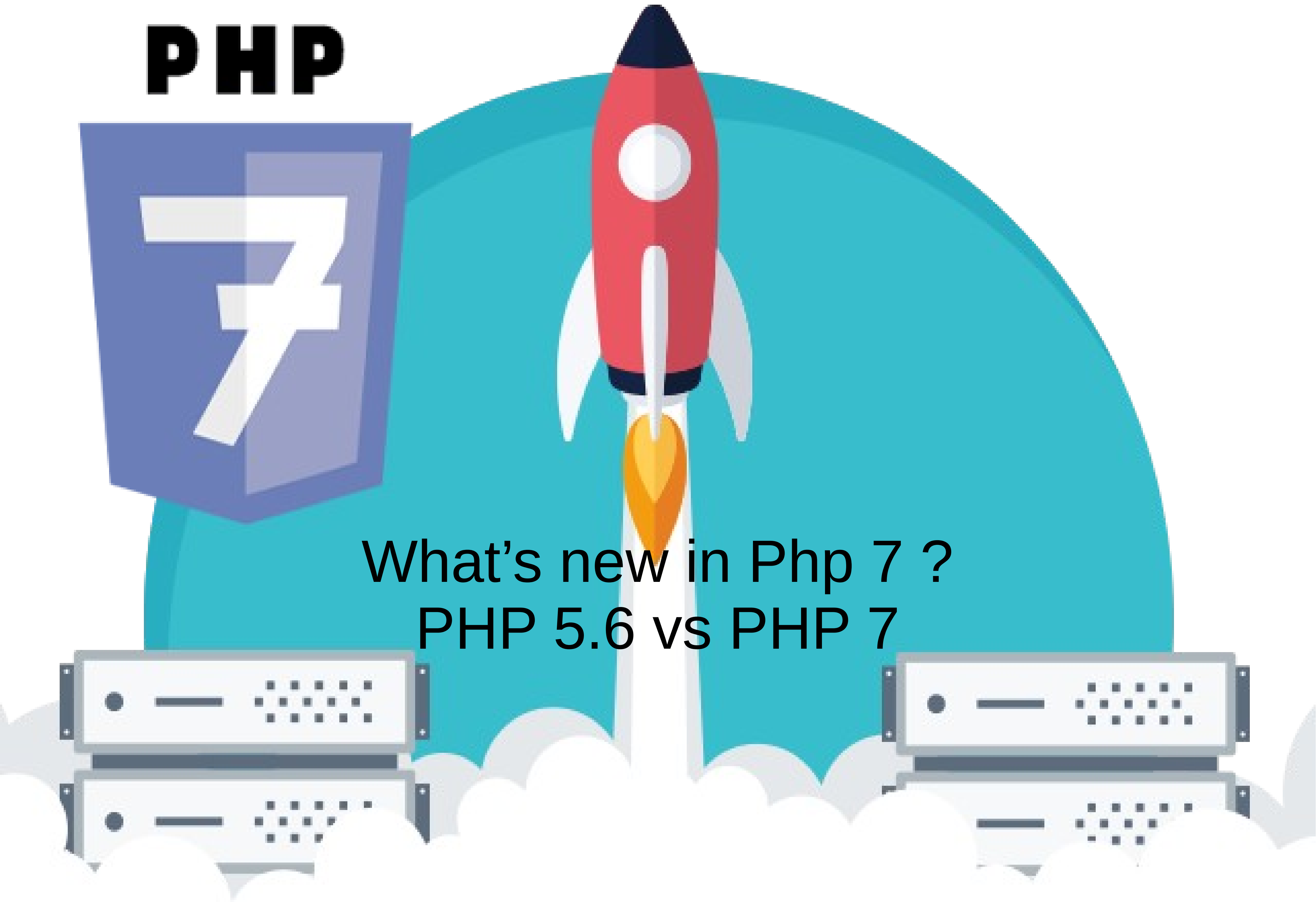
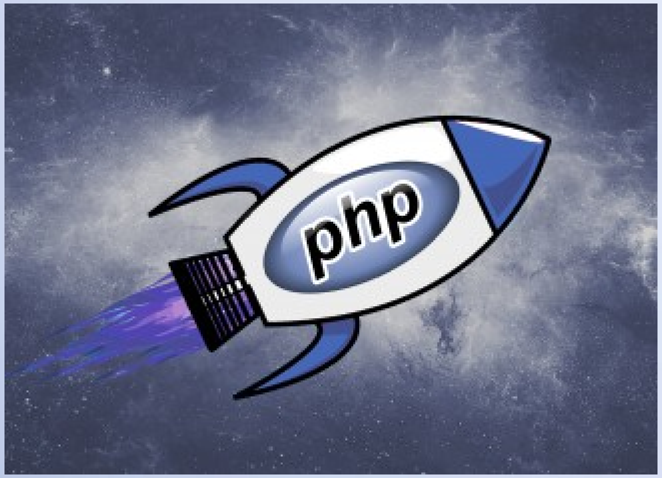


What's new in Php 7 ?
PHP 5.6 vs PHP 7

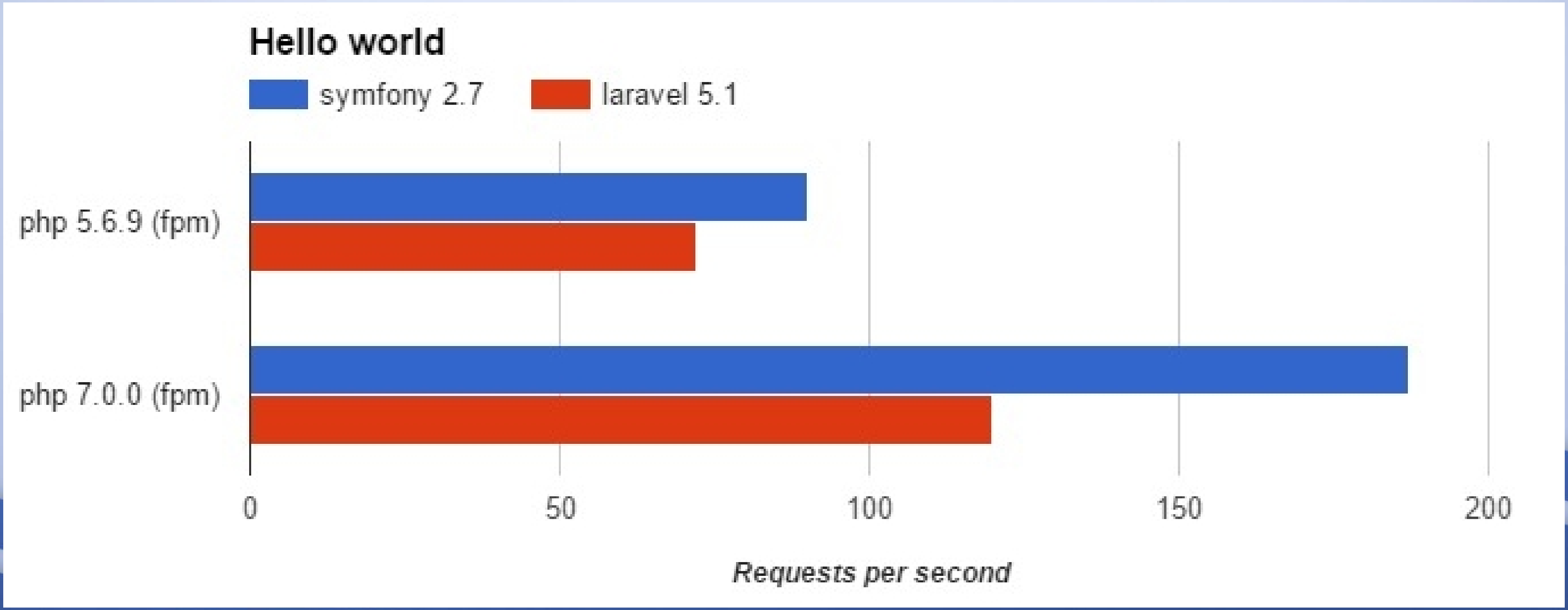
PERFORMANCE

- Php 7 is two times faster than php 5.6
- Php 7 uses significantly less memory than php 5.6



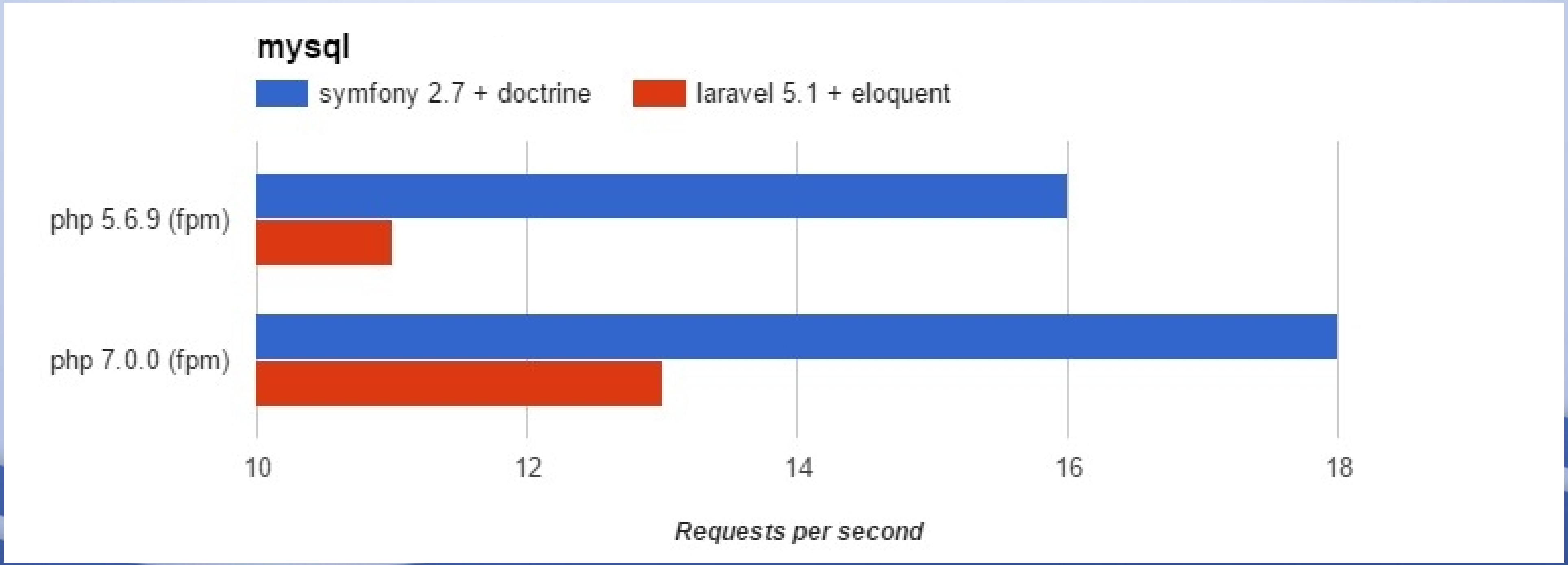
PERFORMANCE

- Hello world – one action, one controller and one view. It was without any additional optimization.

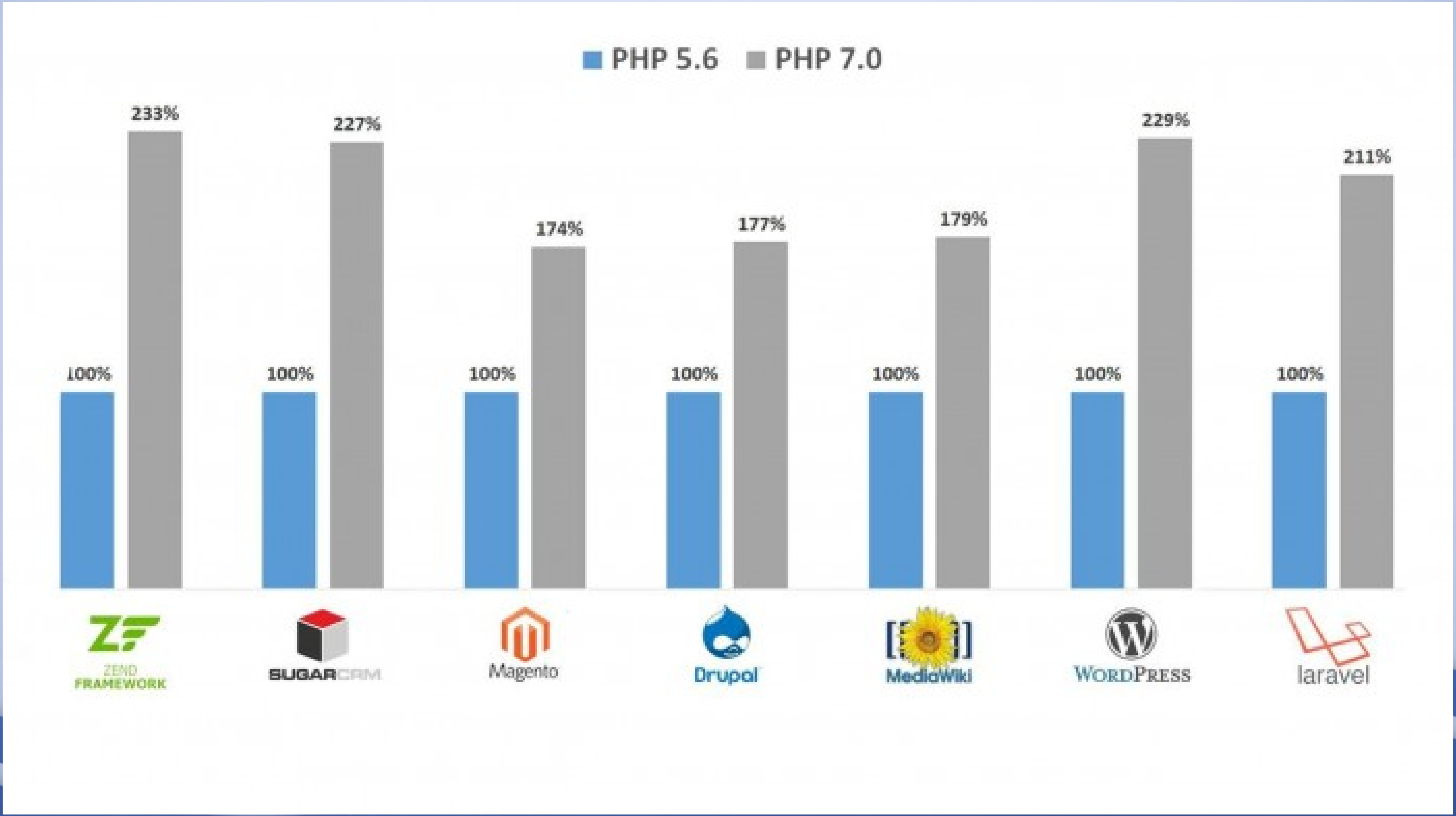


PERFORMANCE

- Mysql - one „INSERT INTO” using a standard tool for handing databases supplied with frameworks.



PERFORMANCE



New features

- Scalar type declarations;
- Return type declarations;
- Null coalescing operator;
- Spaceship operator;
- Constant arrays using define();
- Anonymous classes;
- Filtered unserialize();
- IntlChar;
- Engine Exceptions;
- Group *use* declarations;
- Generator Return Expressions;
- Generator delegation;
- Integer division with intdiv();
- Unicode codepoint escape syntax;
- Session options;

Scalar type declarations

Scalar type declarations come in two flavours:

- coercive (default);
- strict;

Scalar type declarations

The following types for parameters can now be enforced (either coercively or strictly):

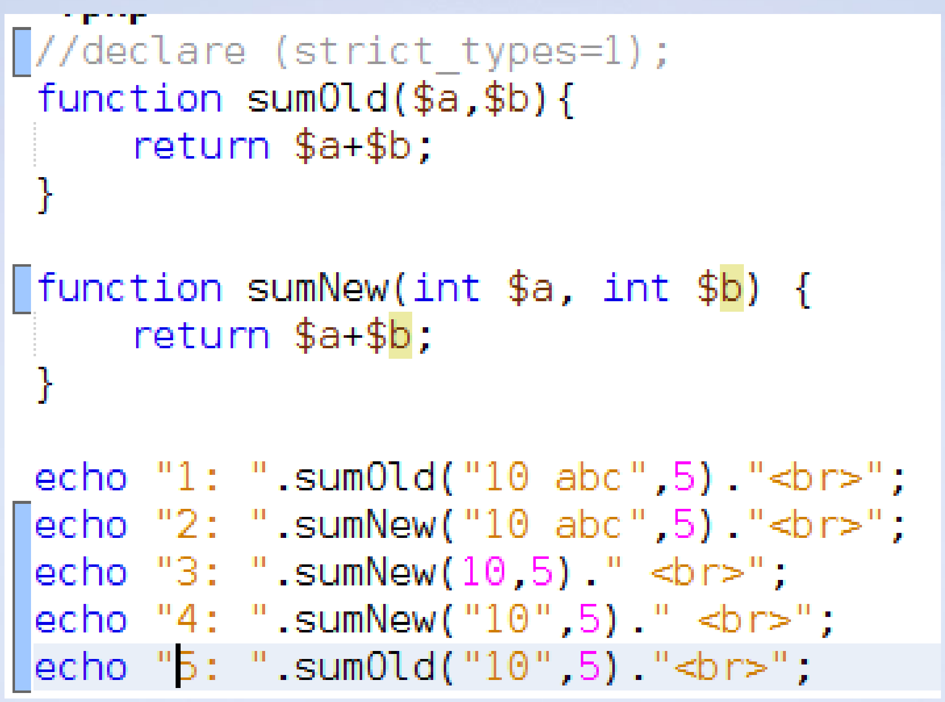
- strings (string);
- integers (int);
- floating-point numbers (float);
- and booleans (bool);

Scalar type declarations

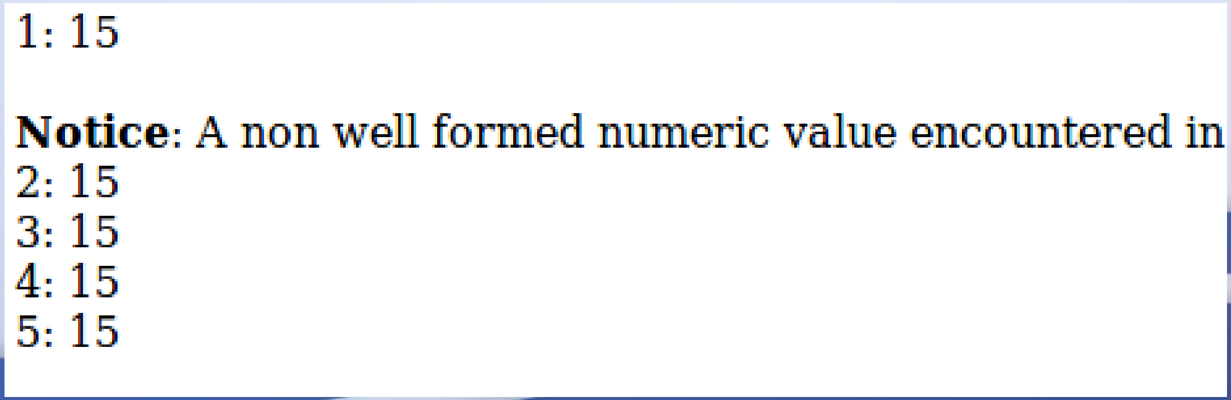
They augment the other types introduced in PHP 5:

- class names;
- interfaces;
- array;
- callable;

Scalar type declarations

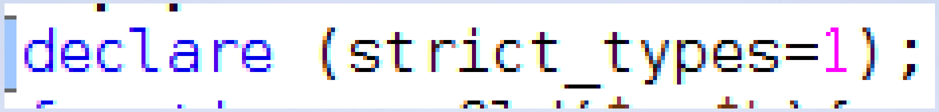


```
//declare (strict_types=1);
function sumOld($a,$b){
    return $a+$b;
}

function sumNew(int $a, int $b) {
    return $a+$b;
}

echo "1: ".sumOld("10 abc",5)."<br>";
echo "2: ".sumNew("10 abc",5)."<br>";
echo "3: ".sumNew(10,5)." <br>";
echo "4: ".sumNew("10",5)." <br>";
echo "5: ".sumOld("10",5)."<br>";
```

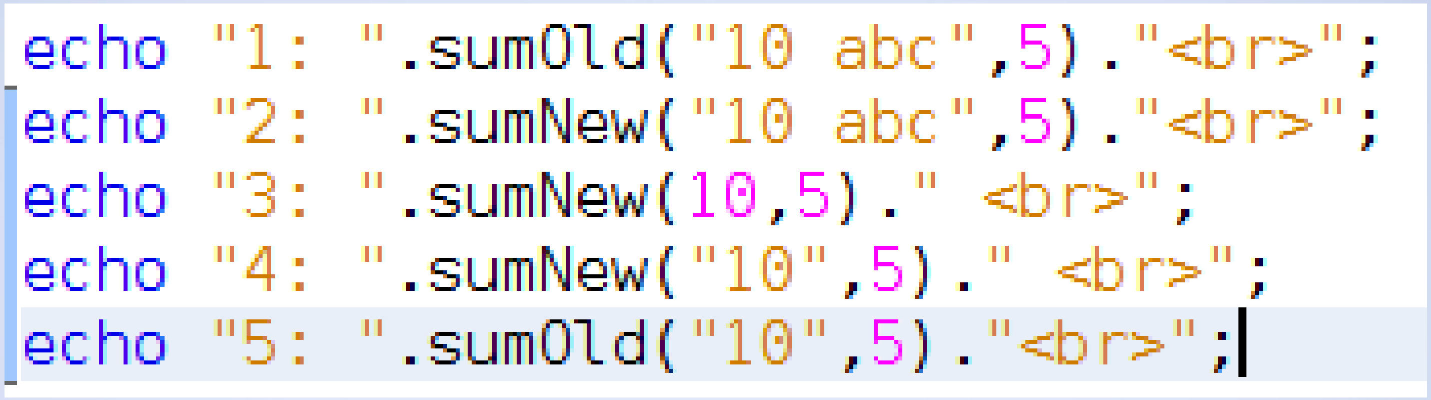


```
1: 15

Notice: A non well formed numeric value encountered in
2: 15
3: 15
4: 15
5: 15
```

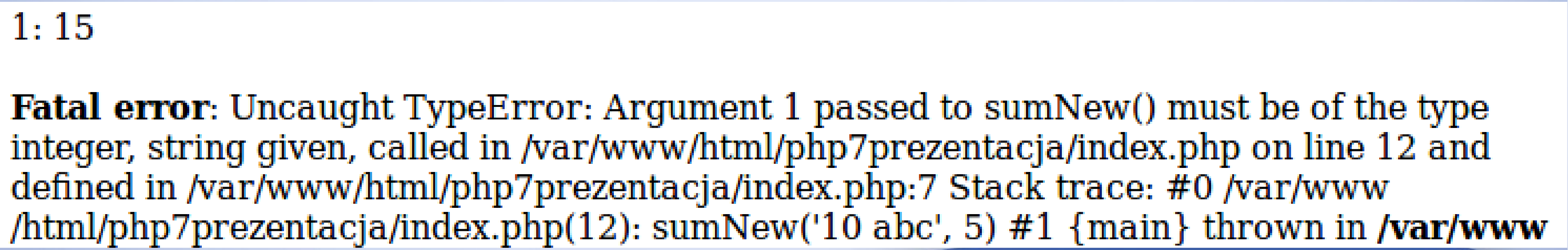
Scalar type declarations



```
declare (strict_types=1);
```

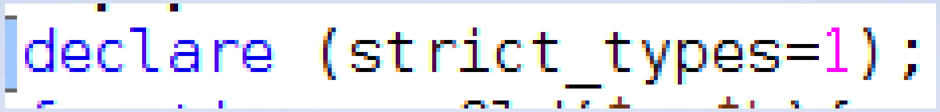


```
echo "1: ".sumOld("10 abc",5)."<br>";
echo "2: ".sumNew("10 abc",5)."<br>";
echo "3: ".sumNew(10,5)." <br>";
echo "4: ".sumNew("10",5)." <br>";
echo "5: ".sumOld("10",5)."<br>";
```

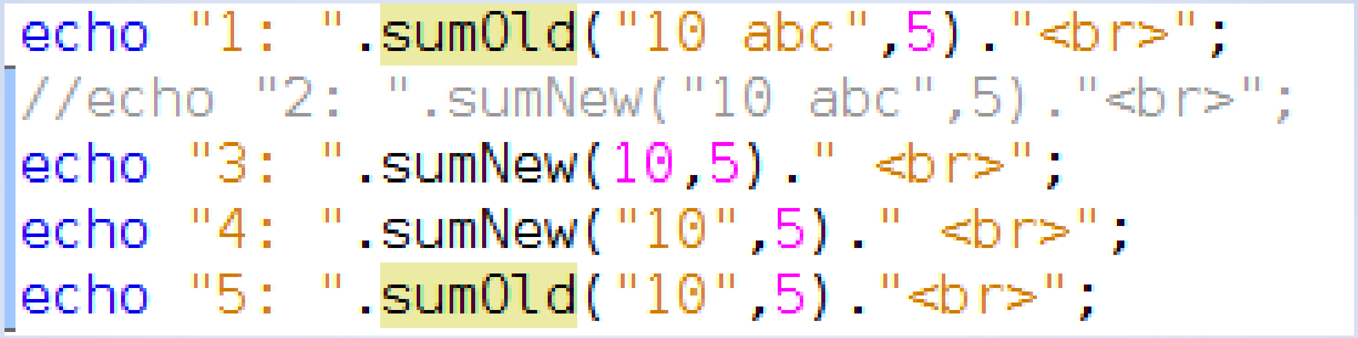


```
1: 15

Fatal error: Uncaught TypeError: Argument 1 passed to sumNew() must be of the type
integer, string given, called in /var/www/html/php7prezentacja/index.php on line 12 and
defined in /var/www/html/php7prezentacja/index.php:7 Stack trace: #0 /var/www
/html/php7prezentacja/index.php(12): sumNew('10 abc', 5) #1 {main} thrown in /var/www
```

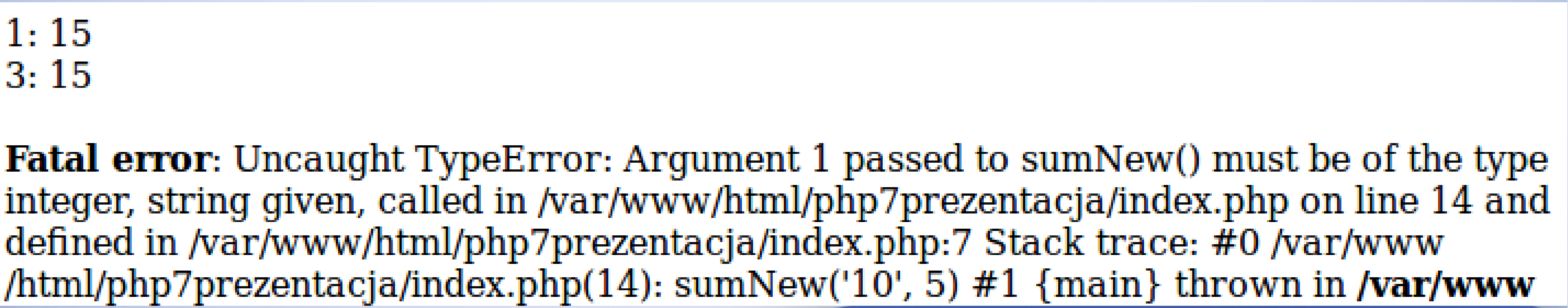
Scalar type declarations



```
declare (strict_types=1);
```



```
echo "1: ".sumOld("10 abc",5)."<br>";
//echo "2: ".sumNew("10 abc",5)."<br>";
echo "3: ".sumNew(10,5)." <br>";
echo "4: ".sumNew("10",5)." <br>";
echo "5: ".sumOld("10",5)."<br>";
```

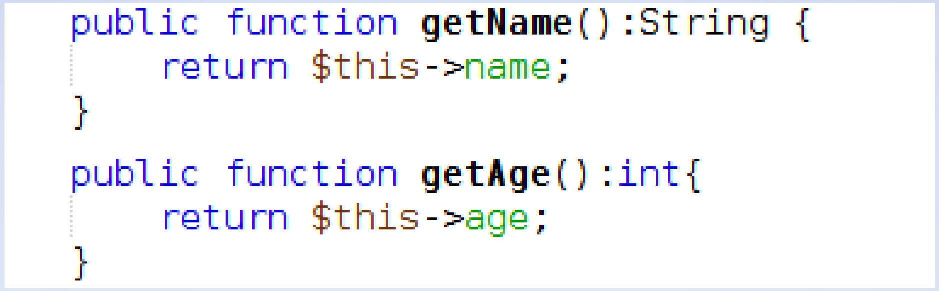


```
1: 15
3: 15

Fatal error: Uncaught TypeError: Argument 1 passed to sumNew() must be of the type
integer, string given, called in /var/www/html/php7prezentacja/index.php on line 14 and
defined in /var/www/html/php7prezentacja/index.php:7 Stack trace: #0 /var/www
/html/php7prezentacja/index.php(14): sumNew('10', 5) #1 {main} thrown in /var/www
```

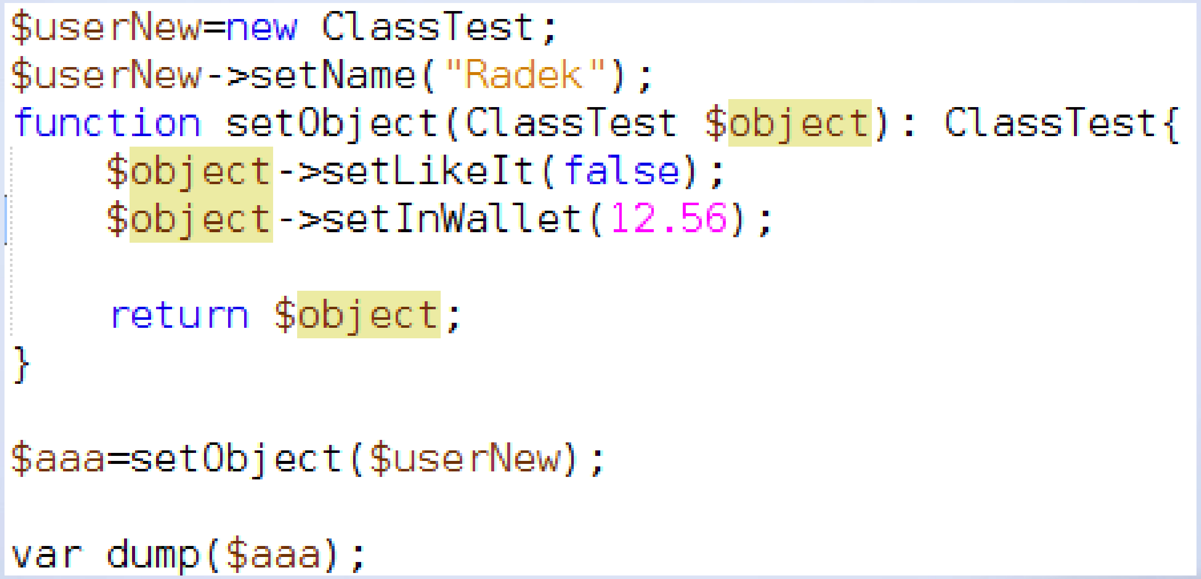
Return type declarations

- PHP 7 adds support for return type declarations.
- return type declarations specify the type of the value that will be returned from a function.
- The same types are available for return type declarations as are available for argument type declarations.

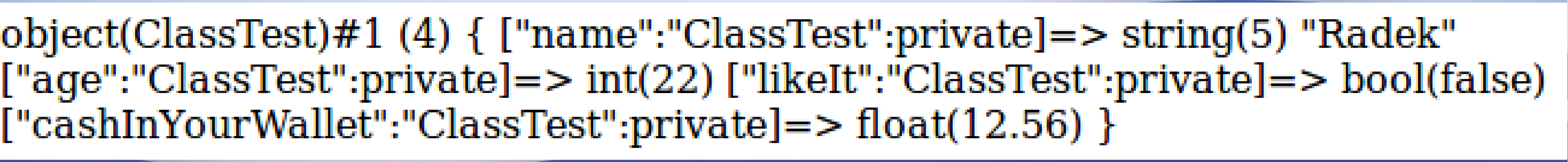
Return type declarations



```
public function getName():String {
    return $this->name;
}

public function getAge():int{
    return $this->age;
}
```

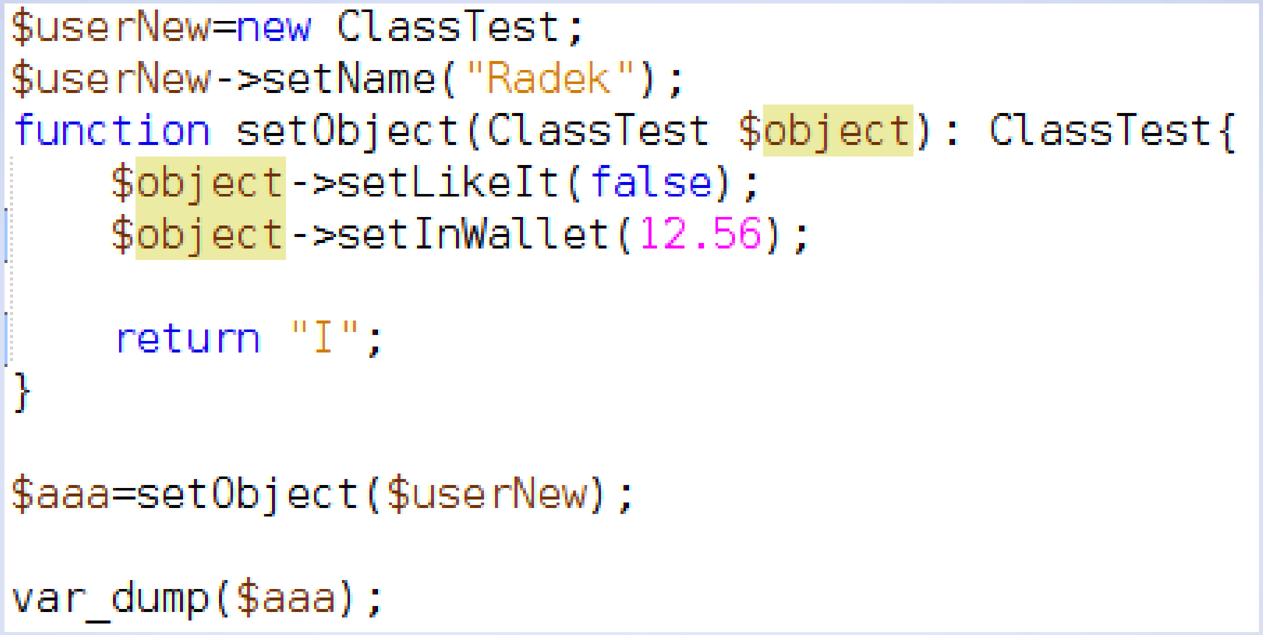


```
$userNew=new ClassTest;
$userNew->setName("Radek");
function setObject(ClassTest $object): ClassTest{
    $object->setLikeIt(false);
    $object->setInWallet(12.56);

    return $object;
}

$aaa=setObject($userNew);

var dump($aaa);
```

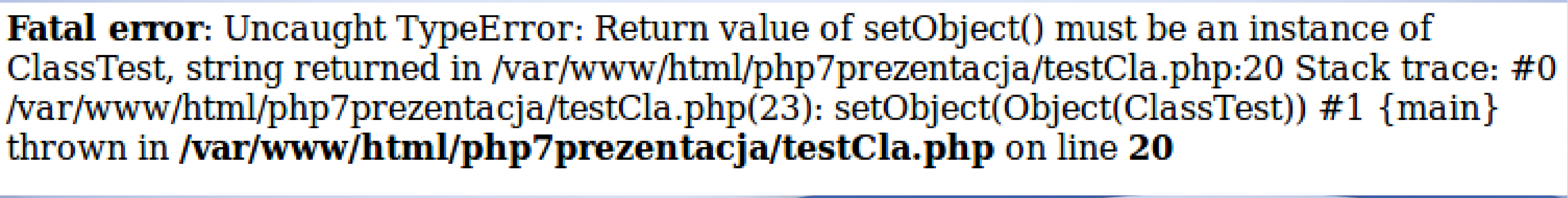


```
object(ClassTest)#1 (4) { ["name":"ClassTest":private]=> string(5) "Radek"
["age":"ClassTest":private]=> int(22) ["likeIt":"ClassTest":private]=> bool(false)
["cashInYourWallet":"ClassTest":private]=> float(12.56) }
```

Return type declarations



```
$userNew=new ClassTest;
$userNew->setName("Radek");
function setObject(ClassTest $object): ClassTest{
    $object->setLikeIt(false);
    $object->setInWallet(12.56);

    return "I";
}

$aaa=setObject($userNew);

var_dump($aaa);
```

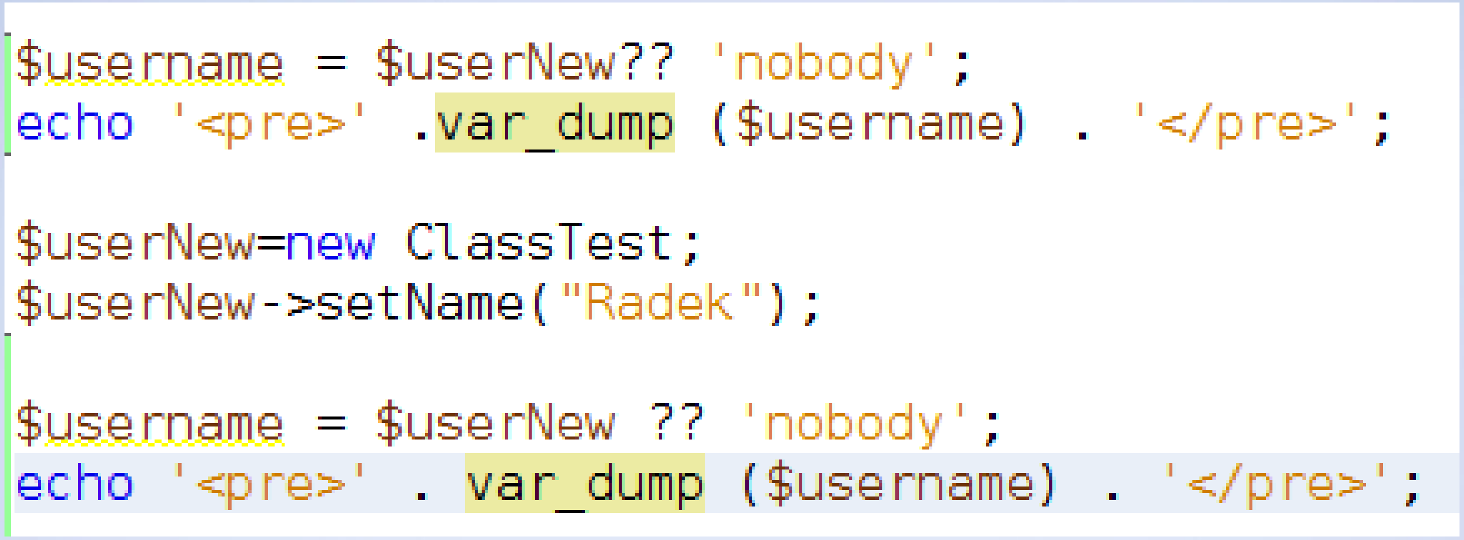


```
Fatal error: Uncaught TypeError: Return value of setObject() must be an instance of ClassTest, string returned in /var/www/html/php7prezentacja/testCla.php:20 Stack trace: #0 /var/www/html/php7prezentacja/testCla.php(23): setObject(Object(ClassTest)) #1 {main} thrown in /var/www/html/php7prezentacja/testCla.php on line 20
```

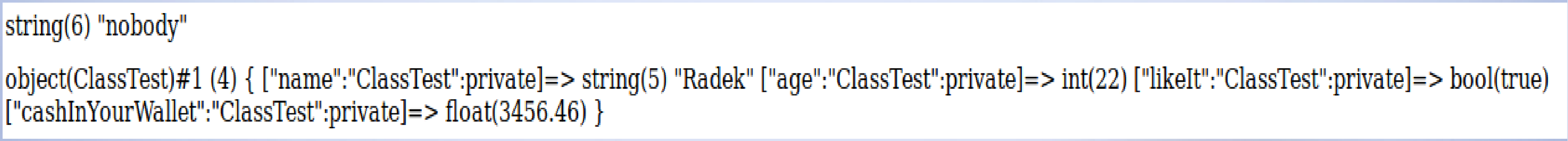
Null coalescing operator

- The null coalescing operator (??) has been added as syntactic sugar for the common case of needing to use a ternary in conjunction with isset().
- It returns its first operand if it exists and is not NULL; otherwise it returns its second operand.

Null coalescing operator

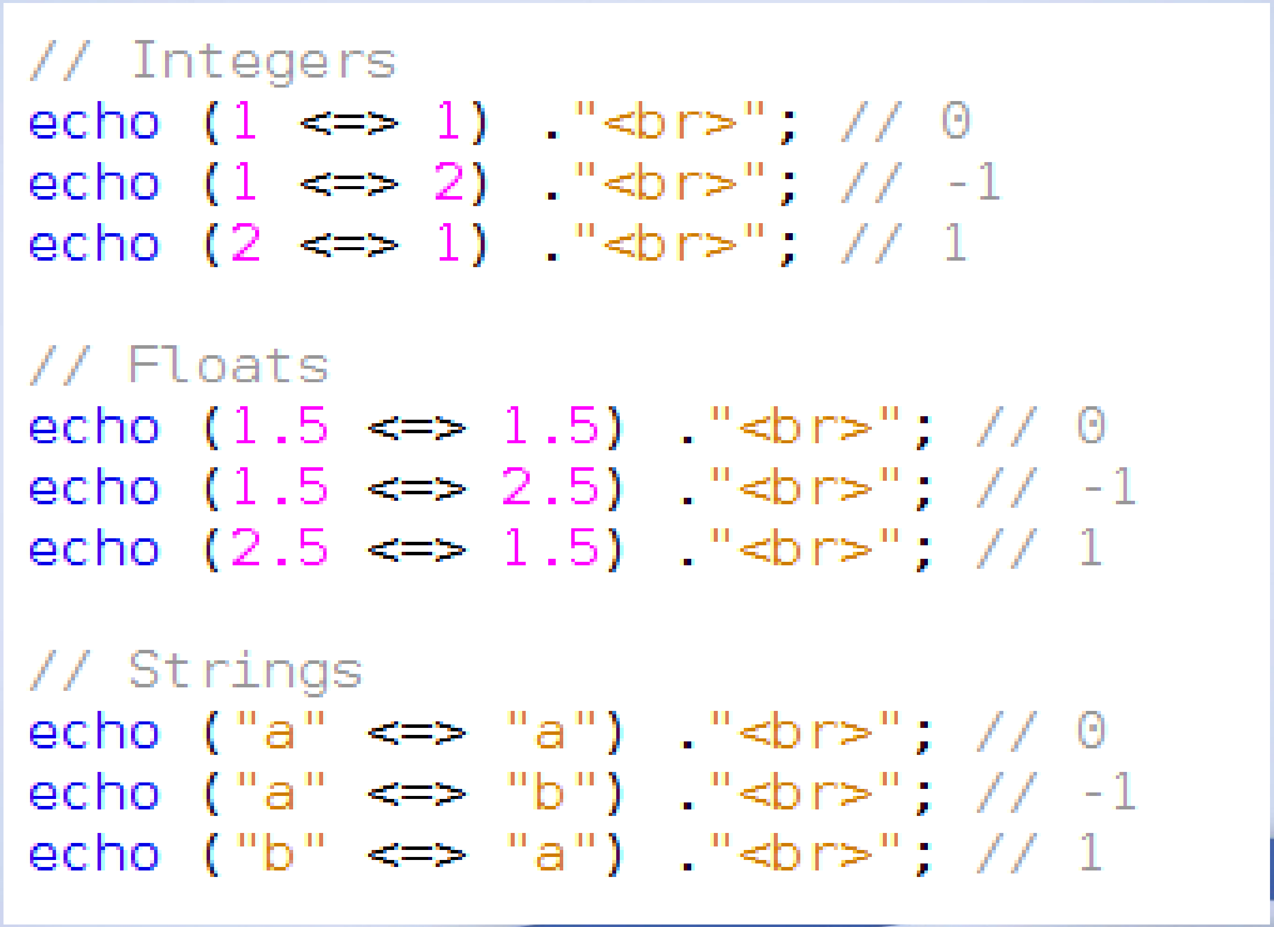


```
$username = $userNew?? 'nobody';
echo '<pre>' .var_dump ($username) . '</pre>';

$userNew=new ClassTest;
$userNew->setName("Radek");

$username = $userNew ?? 'nobody';
echo '<pre>' . var_dump ($username) . '</pre>';
```



```
string(6) "nobody"

object(ClassTest)#1 (4) { ["name":"ClassTest":private]=> string(5) "Radek" ["age":"ClassTest":private]=> int(22) ["likeIt":"ClassTest":private]=> bool(true)
["cashInYourWallet":"ClassTest":private]=> float(3456.46) }
```

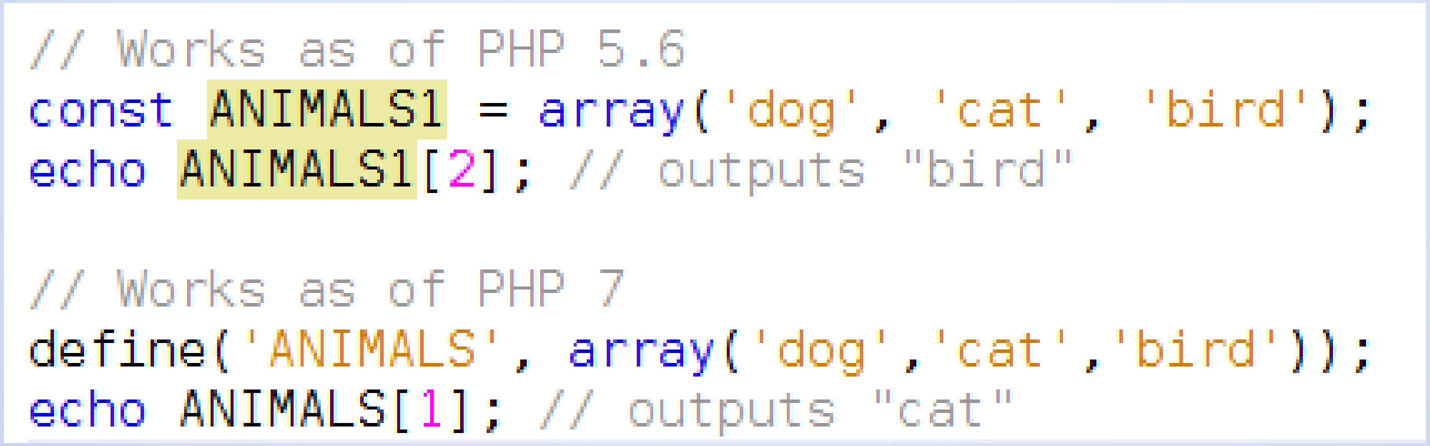
Spaceship operator



```
// Integers
echo (1 <=> 1) ."<br>"; // 0
echo (1 <=> 2) ."<br>"; // -1
echo (2 <=> 1) ."<br>"; // 1

// Floats
echo (1.5 <=> 1.5) ."<br>"; // 0
echo (1.5 <=> 2.5) ."<br>"; // -1
echo (2.5 <=> 1.5) ."<br>"; // 1

// Strings
echo ("a" <=> "a") ."<br>"; // 0
echo ("a" <=> "b") ."<br>"; // -1
echo ("b" <=> "a") ."<br>"; // 1
```

Constant arrays using define()

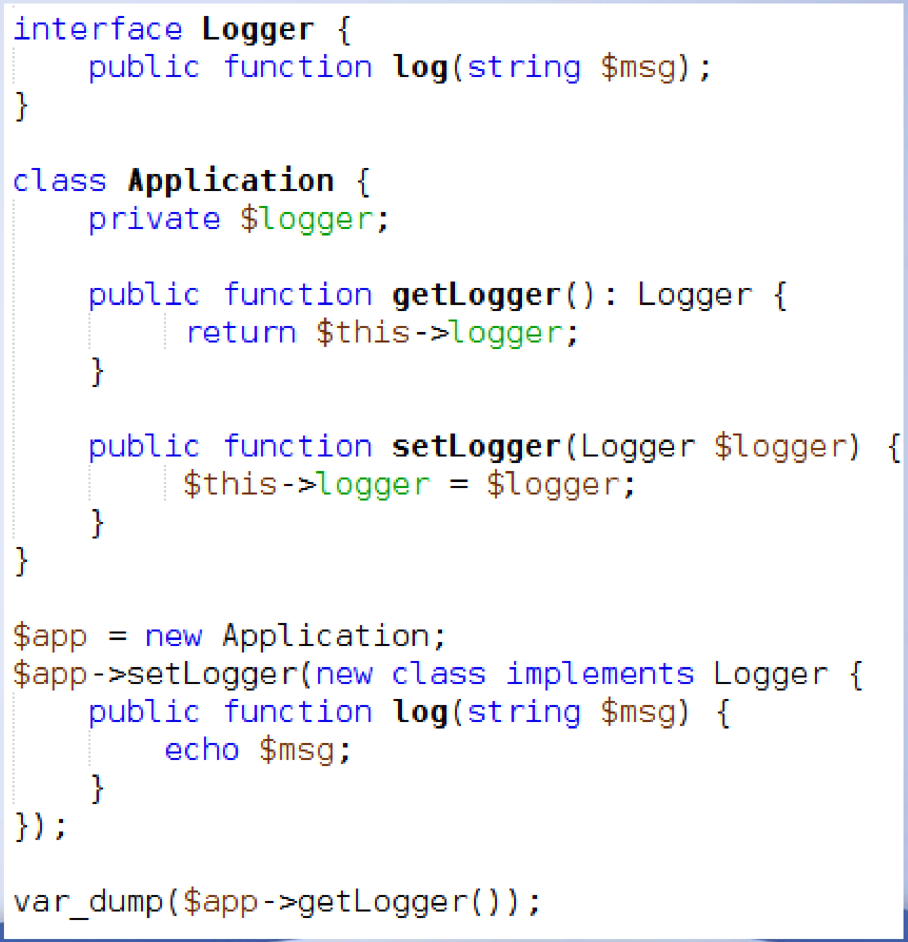
Array constants can now be defined with define(). In PHP 5.6, they could only be defined with const.



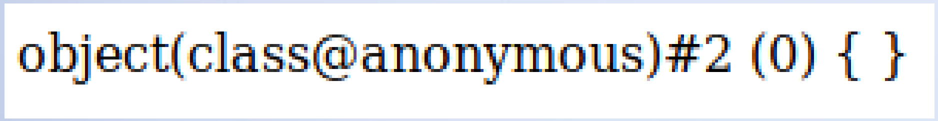
```
// Works as of PHP 5.6
const ANIMALS1 = array('dog', 'cat', 'bird');
echo ANIMALS1[2]; // outputs "bird"

// Works as of PHP 7
define('ANIMALS', array('dog','cat','bird'));
echo ANIMALS[1]; // outputs "cat"
```

Anonymous classes

Anonymous classes are useful when simple, one-off objects need to be created.



```php
interface Logger {
    public function log(string $msg);
}

class Application {
    private $logger;

    public function getLogger(): Logger {
         return $this->logger;
    }

    public function setLogger(Logger $logger) {
         $this->logger = $logger;
    }
}

$app = new Application;
$app->setLogger(new class implements Logger {
    public function log(string $msg) {
        echo $msg;
    }
});

var_dump($app->getLogger());
```



```
object(class@anonymous)#2 (0) { }
```

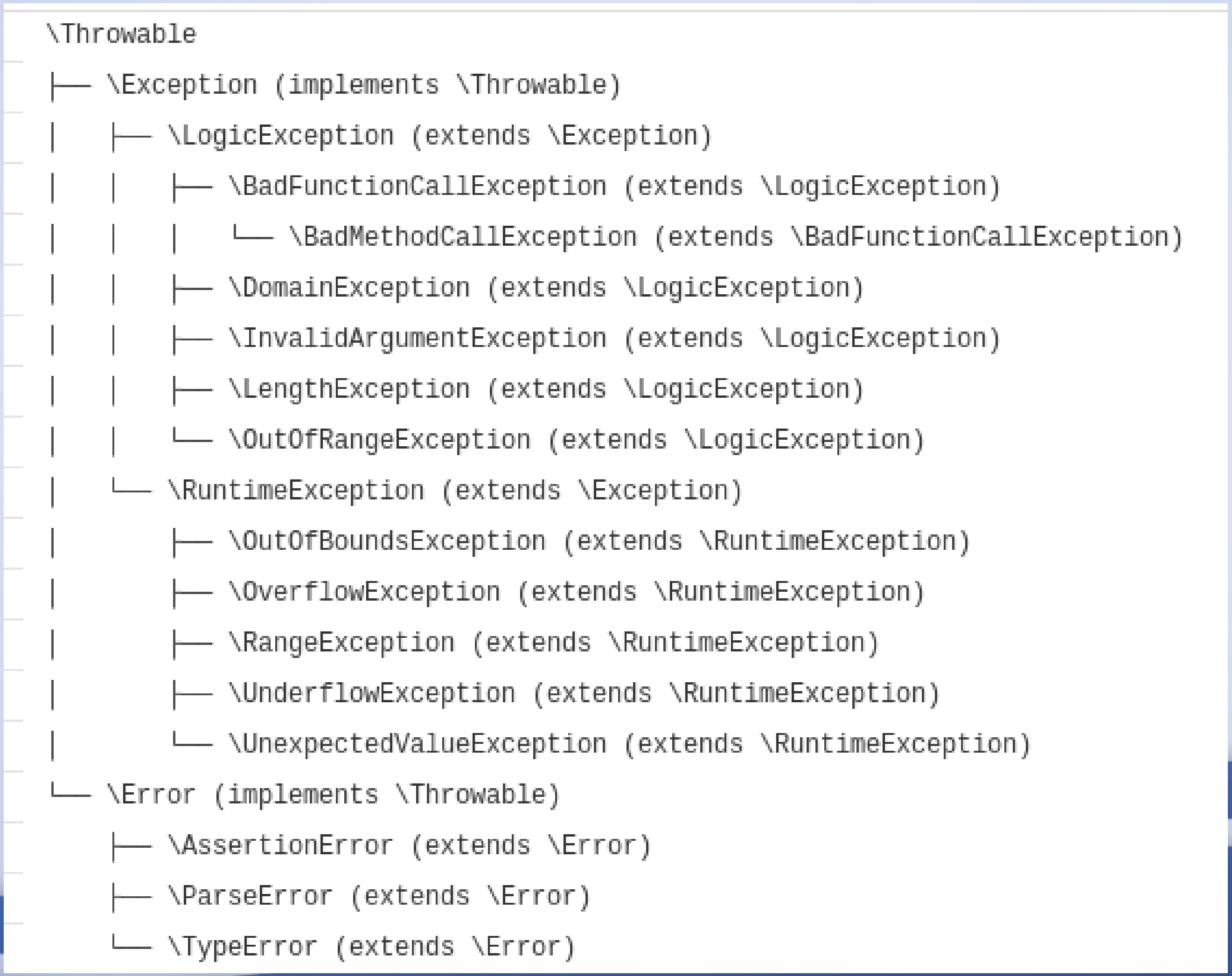
Engine Exceptions

Now we have access to exceptions, such as:

- \ ParseException
- \ BaseException
- \ TypeException
- \ EngineException

Engine Exceptions

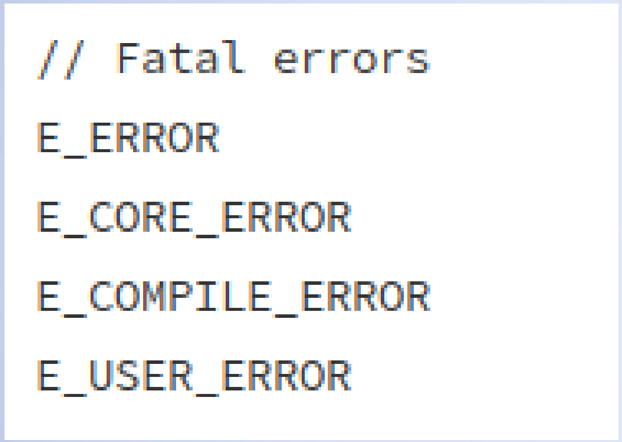
New exception hierarchy



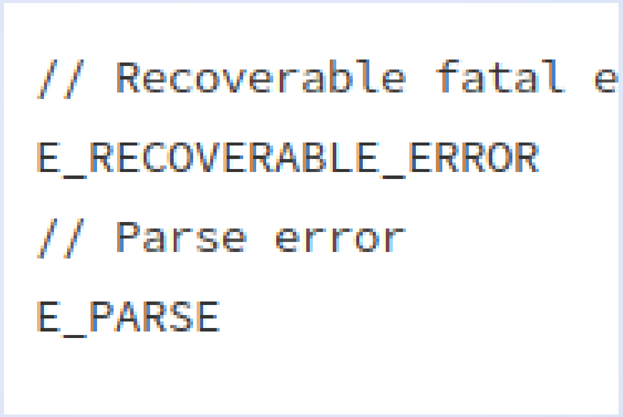
```
\Throwable
├── \Exception (implements \Throwable)
│   ├── \LogicException (extends \Exception)
│   │   ├── \BadFunctionCallException (extends \LogicException)
│   │   │   └── \BadMethodCallException (extends \BadFunctionCallException)
│   │   ├── \DomainException (extends \LogicException)
│   │   ├── \InvalidArgumentException (extends \LogicException)
│   │   ├── \LengthException (extends \LogicException)
│   │   └── \OutOfRangeException (extends \LogicException)
│   └── \RuntimeException (extends \Exception)
│       ├── \OutOfBoundsException (extends \RuntimeException)
│       ├── \OverflowException (extends \RuntimeException)
│       ├── \RangeException (extends \RuntimeException)
│       ├── \UnderflowException (extends \RuntimeException)
│       └── \UnexpectedValueException (extends \RuntimeException)
└── \Error (implements \Throwable)
    ├── \AssertionError (extends \Error)
    ├── \ParseError (extends \Error)
    └── \TypeError (extends \Error)
```

Engine Exceptions

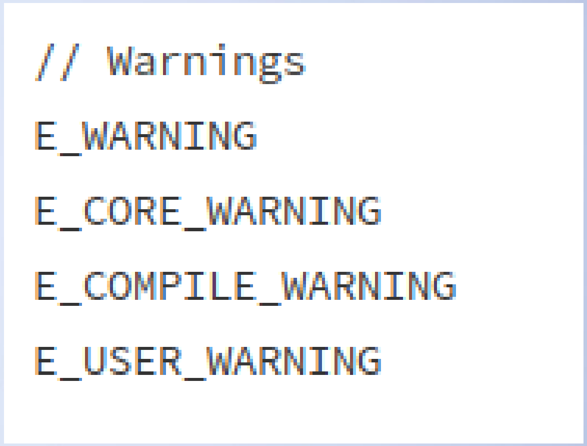
PHP currently supports 16 different error types which are listed below, grouped by severity:



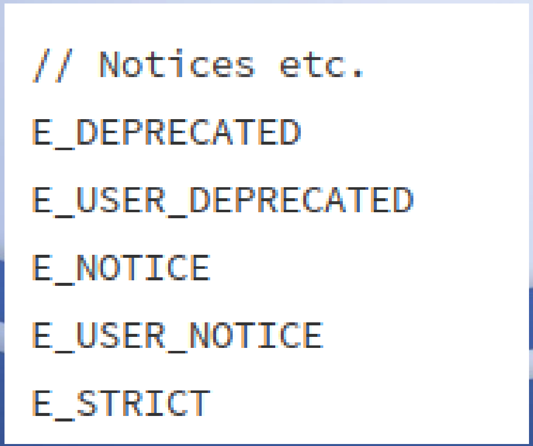
```
// Fatal errors
E_ERROR
E_CORE_ERROR
E_COMPILE_ERROR
E_USER_ERROR
```



```
// Recoverable fatal e
E_RECOVERABLE_ERROR
// Parse error
E_PARSE
```



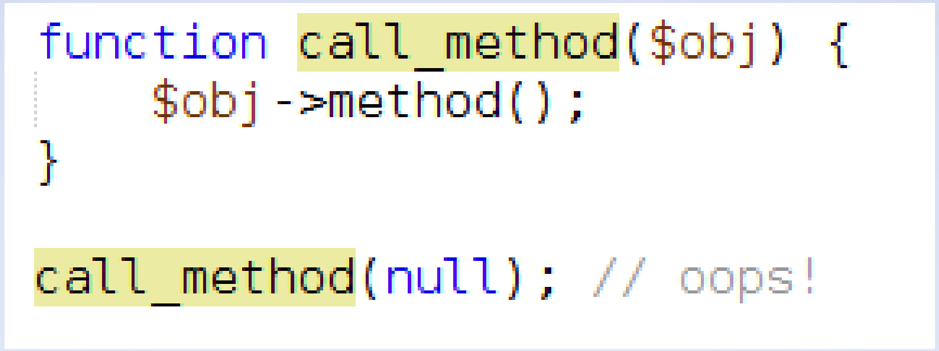
```
// Warnings
E_WARNING
E_CORE_WARNING
E_COMPILE_WARNING
E_USER_WARNING
```



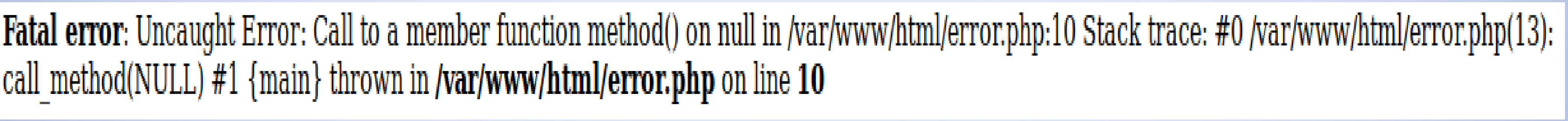
```
// Notices etc.
E_DEPRECATED
E_USER_DEPRECATED
E_NOTICE
E_USER_NOTICE
E_STRICT
```

https://wiki.php.net/rfc/engine_exceptions_for_php7

Engine Exceptions

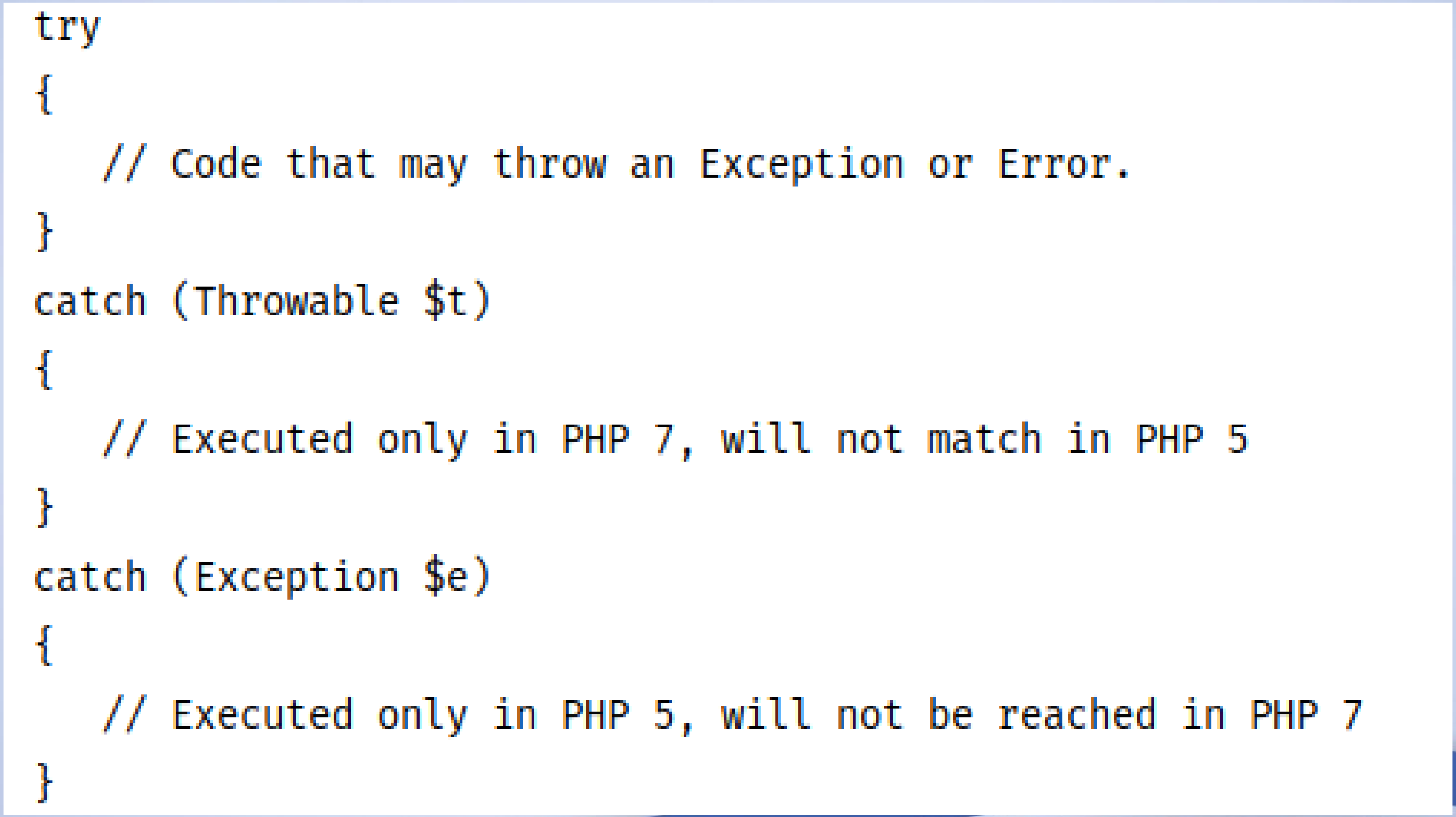


```php
function call_method($obj) {
    $obj->method();
}

call_method(null); // oops!
```



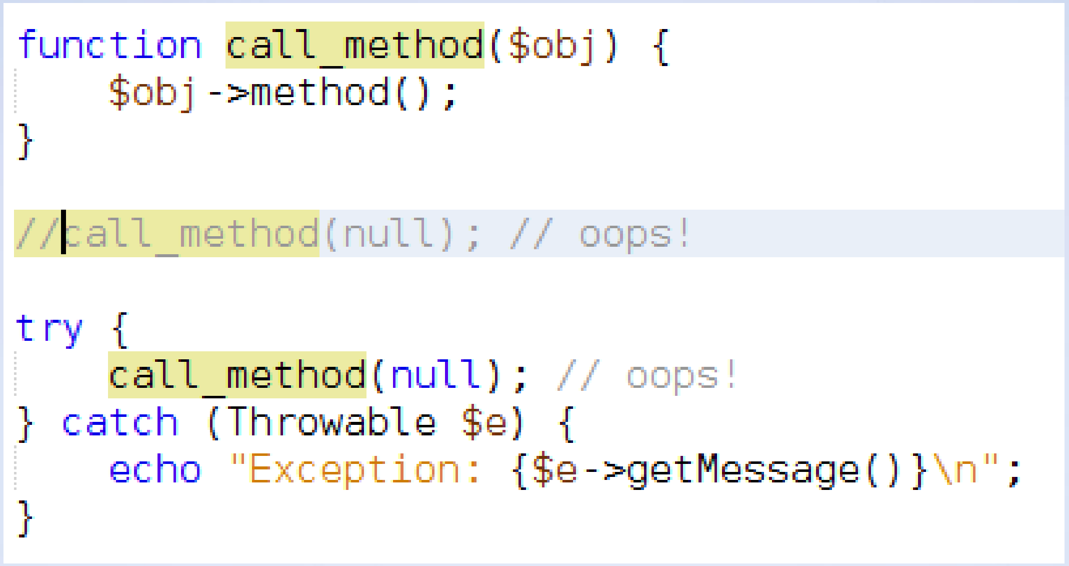
```
Fatal error: Uncaught Error: Call to a member function method() on null in /var/www/html/error.php:10 Stack trace: #0 /var/www/html/error.php(13):
call_method(NULL) #1 {main} thrown in /var/www/html/error.php on line 10
```

Engine Exceptions

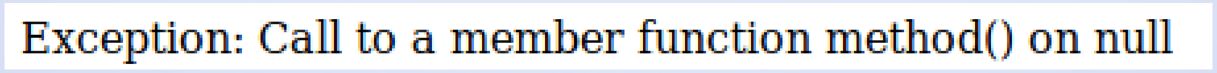


```
try
{
   // Code that may throw an Exception or Error.
}
catch (Throwable $t)
{
   // Executed only in PHP 7, will not match in PHP 5
}
catch (Exception $e)
{
   // Executed only in PHP 5, will not be reached in PHP 7
}
```

Engine Exceptions



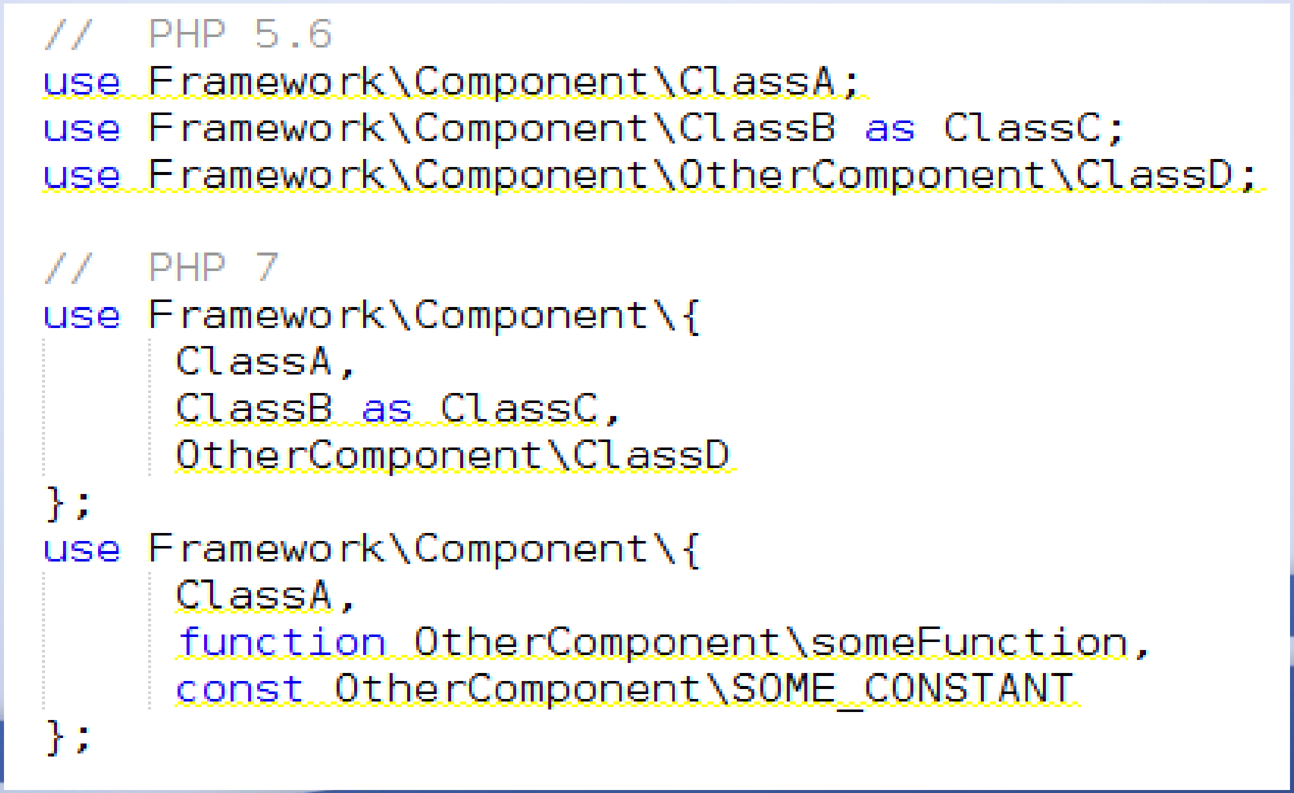
```php
function call_method($obj) {
    $obj->method();
}

//call_method(null); // oops!

try {
    call_method(null); // oops!
} catch (Throwable $e) {
    echo "Exception: {$e->getMessage()}\n";
}
```



```
Exception: Call to a member function method() on null
```

Group use declarations

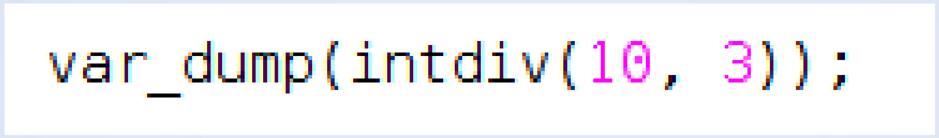
Classes, functions and constants being imported from the same namespace can now be grouped together in a single use statement.



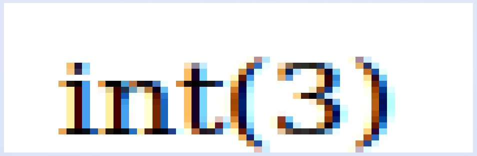
```
//  PHP 5.6
use Framework\Component\ClassA;
use Framework\Component\ClassB as ClassC;
use Framework\Component\OtherComponent\ClassD;

//  PHP 7
use Framework\Component\{
     ClassA,
     ClassB as ClassC,
     OtherComponent\ClassD
};
use Framework\Component\{
     ClassA,
     function OtherComponent\someFunction,
     const OtherComponent\SOME_CONSTANT
};
```

Integer division with intdiv()

The new intdiv() function performs an integer division of its operands and returns it.



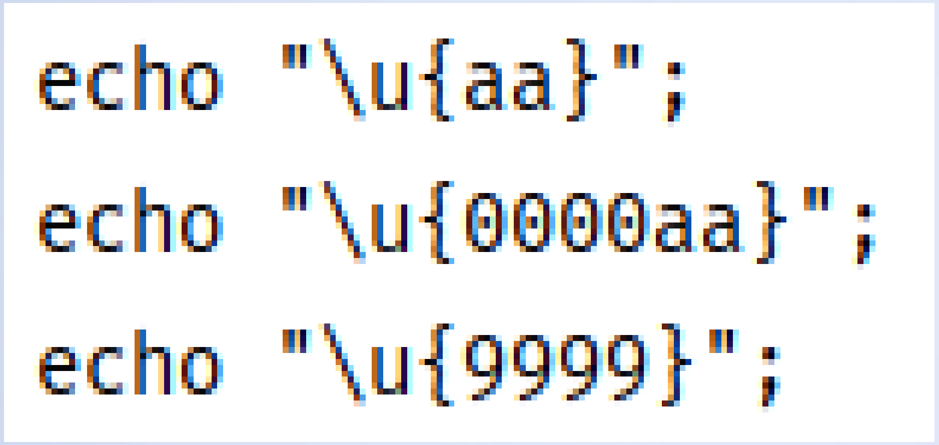
```
var_dump(intdiv(10, 3));
```



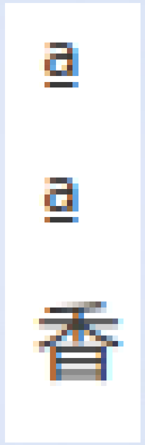
```
int(3)
```

Unicode codepoint escape syntax

UTF-8



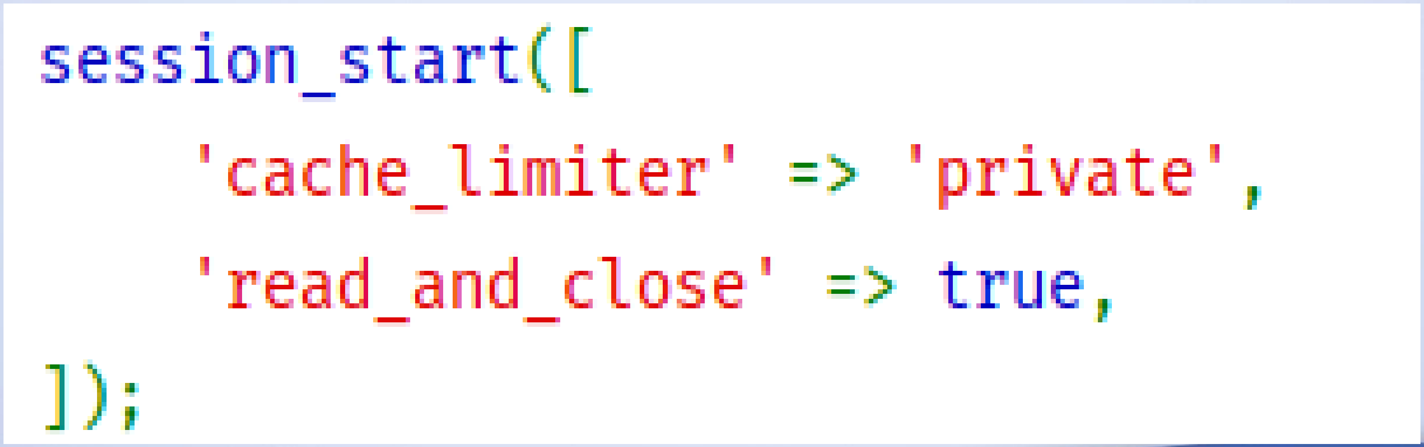
```
echo "\u{aa}";
echo "\u{0000aa}";
echo "\u{9999}";
```



```
ª
ª
香
```

Session options

session_start() now accepts an array of options that override the session configuration directives normally set in php.ini.



```
session_start([
    'cache_limiter' => 'private',
    'read_and_close' => true,
]);
```

Changed functions

- debug_zval_dump() now prints "int" instead of "long", and "float" instead of "double"
- dirname() now optionally takes a second parameter, depth, to get the name of the directory depth levels up from the current directory.
- getrusage() is now supported on Windows.
- mktime() and gmmktime() functions no longer accept is_dst parameter.
- preg_replace() function no longer supports "\e" (PREG_REPLACE_EVAL). preg_replace_callback() should be used instead.

Changed functions

- setlocale() function no longer accepts category passed as string. LC_* constants must be used instead.
- exec(), system() and passthru() functions have NULL byte protection now.
- shmop_open() now returns a resource instead of an int, which has to be passed to shmop_size(), shmop_write(), shmop_read(), shmop_close() and shmop_delete().
- substr() and iconv_substr() now return an empty string, if string is equal to start characters long.
- xml_set_object() now requires to manually unset the $parser when finished, to avoid memory leaks.

Removed Extensions

- ereg
- mssql
- mysql
- sybase_ct

Removed SAPIs

- aolserver
- apache
- apache_hooks
- apache2filter
- caudum
- continuity
- isapi
- milter
- nsapi
- phttpd
- pi3web
- roxen
- thttpd
- tux
- webjames

Thank you for your attention ;)

THANK YOU FOR YOUR ATTENTION
PLEASE CLAP AND DON'T ASK TOUGH QUESTIONS
memegenerator.net