# Обектно ориентирано програмиране с


Максим Крижановски

# puzzle

```
$a = new Student;

$b = new $a;

$a();
```

# ще говорим за

- ✓ какво поддържа php
- ✓ магически методи
- ✓ динамични аспекти на ООП в php
- ✓ някои стандартни интерфейси
- ✓ някои design patterns
- ✓ какво да очакваме от PHP 5.5

# Обекти

✓ Винаги се предават по референция

```
class Student {
   public $name;
}

function setname($student) {
  $student->name = 'John';
}

$student = new Student;
setname($student);
var_dump($student->name);
```

# Обекти

✗ Това за & беше лъжа

```
$instance = new StdClass();

$assigned   =  $instance;
$reference  =& $instance;

$instance->var = '$assigned will have this value';

$instance = null; // $instance and $reference become null

var_dump($instance); // null
var_dump($reference); // null
var_dump($assigned); // object
```

# Обекти

✓ Могат да се клонират

```
$john = new Student;
$john->name = 'John';
$james = clone $john;
$james->name = 'James';
```

Какво се извиква при `clone`?

# Обекти

✓ Имат константи

```
class Student {
   consts university = 'SU';
}
var_dump(Student::university);
```

# $this

```
class A {
    public function test(){
        $this->called = true;
        var_dump($this);
    }

    public static function challenge(){
        var_dump($this);
    }
}
```

# Инстанциране

```php
$class = 'Student';
class Student {
   public static function getInstance()
   {
      return new static;
   }
}
$a = new Student;
$b = new $a;
$c = new $class;
$d = Student::getInstance();
```

# Сравняване

- ✓ ==
- ✓ ===

# self vs static

```
class Animal {
  public static function getInstance()
  {
    return new static;
  }
}

class Dog extends Animal{}
class Cat extends Animal{}

var_dump(Cat::getInstance());
```

# Динамични свойства и методи

```
class Student {
   public function __construct($data) {
     foreach ($data as $key => $val) {
        $this->$key = $val;
      }
   }
}
$s = new Student(['name' => 'John']);
$property = 'name';
var_dump($s->$property);
```

# Магически методи

- ✓ __get() и __set()
- ✓ __call()
- ✓ __clone()
- ✓ __sleep() и __wakeup()
- ✓ __invoke()

```
$student = new Student;
$student('John');
```

# Auto-loading

```
function __autoload($className) {
   include $className . ".php"
}
if (class_exists('Student')) {

}
```

Альтернатива:
spl_autoload_register

# Наследяване

- ✓ При предефиниране на методи, те трябва да имат същата сигнатура
- ✓ Не важи за конструктора, освен ако конструктура не е дефиниран от абстрактен клас

# Интерфейси

✗ Един клас не може да имплементира два интерфейса, които имат метод с едно и също име, но различна сигнатура

✓ Интерфейса може да дефинира константи

✓ Интерфейса подлежи на autoloading

✓ Може да се използва за проверка на типа

# Стандартни интерфейси

- ✓ Countable
- ✓ `ArrayAccess`

# Type hinting

```
function foo($s) {
   if ($s instanceof Student)
   }
}
function bar(Student $s) {
}
function foobar($s){
   switch (get_class($s)){
   }
}
```

# Reflection API

- ✓ ReflectionClass
- ✓ ReflectionMethod
- ✓ ReflectionParameter
- ✓ ReflectionObject

# Fluent interface

```
$obj->moveLeft()->moveRight()->stop();
```

# MVC*

- ✓ Model
- ✓ View
- ✓ Controller


# Traits

```
trait Fly {
  public function fly(){}
}

class Bird extends Animal {
  use Fly;
}
```

# PHP 5.5

- ✓ yield
- ✓ finally
- ✓ и много други :)

```
function getNumber(){
  $i = 0;
  while (true){
    yield ++$i;
  }
}
foreach ($number as getNumber()) { ... }
```

# Contacts

Maxim Krizhanovsky
darhazer@gmail.com
http://www.linkedin.com/in/darhazer

PHP Bulgaria @ Facebook
https://www.facebook.com/groups/459347284112503/

php.dev.bg