# OOP IN PHP

## تكليف تطوير الويب متقدم عملي


**الدكتور**

**إبراهيم الشامي**

**سمر عزي محمد الكرعمي**

**تقنية معلومات**

**مستوى ثالث**

# البرمجة كائنية التوجه فيPHP

بدءًا من الإصدار الخامس من اللغة، أعيدت كتابة نموذج الكائنات للحصول على أداء أفضل وإضافة المزيد من الميزات، وقد كان هذا من التغييرات الكبيرة في اللغة، إذ قدّم الإصدار الخامس نموذج مكتملًا للكائنات.

ومن الميزات التي قدّمها الإصدار الخامس قابلية الرؤية (visibility)، الأصناف والتوابع المجرّدة (abstract) والنهائية (final)، المزيد من التوابع السحرية ( magical methods)، الواجهات (interfaces)، الاستنساخ (cloning) والإشارة إلى الأنواع (typehinting).

تعامل PHP الكائنات معاملة المراجع (references) أو المقابض (handles)، بمعنى أنّ كلّ متغيّر يتضمن كائنًا مرجعيًا وليس نسخة من الكائن كلّه. انظر الكائنات والمراجع.

نصيحة: راجع أيضًا دليل تسمية Userland.

## الأساسيات

### الصنف

تبدأ الصيغة البسيطة للتعريف عن الصنف بالكلمة المفتاحية class، يتبعها اسم الصنف ويتبعه زوج من الأقواس المعقوفة التي تحيط بالعبارات المسؤولة عن تعريف الخصائص والتوابع المُنتمية إلى الصنف.

يمكن أن يحمل الصنف أي اسم بشرط أن لا يكون كلمةً محجوزةً في PHP. يبدأ اسم الصنف النظامي بحرف أو شرطة سفلية، تتبعها أيّ عددٍ من الحروف أو الأرقام أو الشرطات السفلية. ويمكن التعبير عن هذه الصيغة باستخدام التعابير النمطية بالصورة التالية: `^[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*$`.

يمكن أن يحتوي الصنف على ثوابت ومتغيرات (تسمّى "خصائص" [properties]) ودوالّ (تسمى "توابع" [methods]) خاصة به.

المثال ١: مثال عن تعريف بسيط للصنف

```
<?php
class SimpleClass
{
// التصريح عن الخاصّية
public $var = 'a default value';
```

```
// التصريح عن التابع
public function displayVar() {
echo $this->var;
}
}
?>
```

يكون المتغيّر الزائف `$this` متاحًا عند استدعاء التابع من داخل سياق الكائن.
يعدّ `$this` مرجعًا للكائن المستدعي (عادة ما يكون الكائن الذي ينتمي إليه التابع، ولكن قد يكون كائنًا آخر، وذلك عندما تستدعى الدالة سكونيّاً [statically] ضمن سياق خاصّ بكائن آخر). منذ الإصدار ٧,٠,٠ من اللغة يؤدي الاستدعاء الساكن لتابع غير ساكن يؤدي إلى عدم التعرّف على $this داخل التابع. منذ الإصدار ٥,٦,٠ من اللغة أصبحت عملية الاستدعاء الساكن لتابع غير ساكن من سياق غير متوافق عملية مهملة. أما في الإصدار ٧,٠,٠ فقد أصبحت عملية الاستدعاء الساكن لتابغ غير ساكن مهملة بصورة عامّة (حتى لو استدعي التابع من سياق متوافق). يؤدي هذا النوع من الاستدعاء إلى إطلاق ملاحظة من نوع strict في الإصدارات السابقة للإصدار ٥,٦,٠.

المثال ٢: بعض الأمثلة على المتغير الزائف `$this`

سنفترض في هذا المثال أنّ خاصية الإبلاغ عن الأخطار `error_reporting` معطّلة، وإلا ستطلق الشيفرة التالية ملاحظات من نوع deprecated و strict على التوالي، حسب إصدار اللغة المستخدم.

```
<?php
class A
{
function foo()
{
if (isset($this)) {
echo '$this is defined (';
echo get_class($this);
echo ")\n";
} else {
echo "\$this is not defined.\n";
}
}
}

class B
{
```

```php
function bar()
{
A::foo();
}
}

$a = new A();
$a->foo();

A::foo();
$b = new B();
$b->bar();

B::bar();
?>
```

تعطي الشيفرة السابقة النتيجة التالية في الإصدار ٥ من اللغة:

```
$this is defined (A)
$this is not defined.
$this is defined (B)
$this is not defined.
```

أما الإصدار ٧ من اللغة فيعطي النتائج التالية:

```
$this is defined (A)
$this is not defined.
$this is not defined.
$this is not defined.
```

### الكلمة المفتاحية new

لإنشاء نسخة (instance) من الصنف يجب استخدام الكلمة المفتاحية new، ودائمًا ما يتمّ إنشاء الكائن إلا إذا كان يمتلك دالة بانية constructor ترمي استثناءً عند وقوع خطأ ما. يستحسن تعريف الأصناف قبل تهيئتها instantiation (وفي بعض الأحيان يكون الأمر مطلوبًا).

في حال استخدام سلسلة نصّية تحتوي اسم أحد الأصناف مع الكلمة المفتاحية `new`، فإن اللغة تنشئ نسخة جديدة من ذلك الصنف، وإن كان الصنف ضمن مجال أسماء معيّن، فيجب استخدام اسمه المؤهّل بالكامل (fully qualified name).

المثال ٣: إنشاء نسخة من الصنف

```php
<?php
$instance = new SimpleClass();

// يمكن القيام بهذا ضمن متغيّر
$className = 'SimpleClass';
$instance = new $className(); // new SimpleClass()
?>
```

من الممكن إنشاء كائن جديد ضمن سياق الصنف وذلك باستخدام العبارتين `new self` و `new parent`.

عند إسناد نسخة من الصنف إلى متغيّر جديد يصبح بمقدور المتغير الوصول إلى النسخة ذاتها كما هو حال الكائن الذي أُسند إليه، وتسلك عملية تمرير النسخ إلى الدوالّ السلوك ذاته. يمكن إنشاء نسخة من عنصر تمّ إنشاؤه مسبقًا عن طريق استنساخه.

المثال ٤: إسناد الكائنات

```php
<?php

$instance = new SimpleClass();

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

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

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

var_dump($instance);
var_dump($reference);
var_dump($assigned);
?>
```

يعطي المثال السابق النتيجة التالية:

```
NULL
NULL
```

```
object(SimpleClass)#1 (1) {
["var"]=>
string(30) "$assigned will have this value"
}
```

قدّم الإصدار ٥,٣,٠ من اللغة طريقتين جديدتين لإنشاء نسخ من الكائنات: المثال ٥: إنشاء كائنات جديدة

```
<?php
class Test
{
static public function getNew()
{
return new static;
}
}

class Child extends Test
{}

$obj1 = new Test();
$obj2 = new $obj1;
var_dump($obj1 !== $obj2);

$obj3 = Test::getNew();
var_dump($obj3 instanceof Test);

$obj4 = Child::getNew();
var_dump($obj4 instanceof Child);
?>
```

تعطي الشيفرة السابقة المخرجات التالية:

```
bool(true)
bool(true)
bool(true)
```

قدّم الإصدار ٥,٤,٠ من اللغة إمكانية الوصول إلى أحد أعضاء (أي الخصائص أو التوابع) الكائنات الجديدة باستخدام تعبير واحد: المثال ٦: الوصول إلى أحد أعضاء كائن جديد

```php
echo (new DateTime())->format('Y');
?>
```

تعطي الشيفرة السابقة المخرجات التالية:

```
2016
```

## الخصائص والتوابع

تستقرّ خصائص وتوابع الصنف في مجال أسماء منفصل؛ لذا يمكن أن يحمل صنف وتابع الاسم ذاته. تقدّم اللغة صيغة واحدة للوصول إلى الخصائص أو التوابع، وتعتمد هذه العملية على السياق فقط، بمعنى أنّه هل أن طريقة الاستخدام هي الوصول إلى متغير أو استدعاء دالة.

المثال ٧: الوصول إلى خاصية مقابل استدعاء تابع

```php
<?php
class Foo
{
public $bar = 'property';

public function bar() {
return 'method';
}
}

$obj = new Foo();
echo $obj->bar, PHP_EOL, $obj->bar(), PHP_EOL;
```

يعطي المثال السابق النتائج التالية:

```
property
method
```

هذا يعني أنّه ليس بالإمكان استدعاء دالة مجهولة مُسندة إلى خاصية بصورة مباشرة، بل يجب إسناد الخاصية إلى متغير في البداية على سبيل المثال. أما في الإصدار ٧,٠,٠ فقد أصبح بالإمكان استدعاء مثل هذه الخاصّية مباشرة وذلك بإحاطتها بالأقواس. المثال ٨: استدعاء دالة مجهولة مخزّنة في خاصّية

```php
<?php
class Foo
{
public $bar;

public function __construct() {
$this->bar = function() {
return 42;
};
}
}

$obj = new Foo();

// منذ الإصدار
// PHP 5.3.0:
$func = $obj->bar;
echo $func(), PHP_EOL;

// يمكن استخدام هذه الصيغة في الإصدار
// PHP 7.0.0:
echo ($obj->bar)(), PHP_EOL;
```

تعطي الشيفرة السابقة النتيجة التالية:

```
42
```

### الكلمة المفتاحية extends

يمكن للصنف أن يرث توابع وخصائص صنف آخر باستخدام الكلمة المفتاحية `extends` عند التصريح عن الصنف، ولا تتيح اللغة الوراثة من أكثر من صنف واحد.

يمكن تجاوز التوابع والخصائص الموروثة عن طريق إعادة التصريح عنها باستخدام نفس الأسماء التي تحملها في الصنف الأب. ولكن إن كان أحد التوابع معرّفًا في الصنف الأب باستخدام الكلمة المفتاحية `final`، فلا يمكن حينئذٍ تجاوز ذلك التابع. يمكن الوصول إلى التوابع المُتجاوز عليها أو الخصائص الساكنة في الصنف الأب عن طريق `parent::`.

يجب أن لا يتغير توقيع المعامل [parameter signature] عند تجاوز (override، أي إعادة تعريف) التوابع، وإلا ستطلق اللغة خطأً من المستوى E_STRICT، ولكن لا ينطبق هذا على الدالة البانية والتي تسمح بالتجاوز باستخدام معاملات مختلفة.

المثال ٩: وراثة الأصناف

```php
<?php
class ExtendClass extends SimpleClass
{
// إعادة تعريف التابع الذي ينتمي للصنف الأب
function displayVar()
{
echo "Extending class\n";
parent::displayVar();
}
}

$extended = new ExtendClass();
$extended->displayVar();
?>
```

تعطي الشيفرة السابقة النتيجة التالية:

```
Extending class
a default value
```

### الكلمة المفتاحية class::

أصبح بالإمكان استخدام الكلمة المفتاحية `class` في الإصدار ٥,٥ من اللغة للحصول على اسم الصنف. يمكن الحصول على سلسلة نصّية تتضمن الاسم المؤهّل بالكامل للصنف وذلك باستخدام الصيغة `ClassName::class`، وهذه الميزة مفيدة جدًّا عند التعامل مع الأصناف المنتمية إلى مجالات أسماء مختلفة.

المثال ١٠: الحصول على اسم الصنف

```php
<?php
namespace NS {
class ClassName {
}

echo ClassName::class;
}
```

```
?>
```

تعطي الشيفرة السابقة النتيجة التالية:

```
NS\ClassName
```

ملاحظة: تجري عملية الحصول على اسم الصنف باستخدام `::class` في وقت التصريف. هذا يعني أنّه عند إنشاء السلسلة النصّية التي تحمل اسم الصنف فإنّ عملية التحميل التلقائي (autoloading) لم تحدث بعد، ونتيجة لذلك، تتوسّع أسماء الأصناف حتى لو كانت الصنف غير معرّفًا، ولن يحدث أيّ خطأً في هذه الحالة.

## الخاصيات

تطلق تسمية "الخاصيات properties" على المتغيّرات المعرّفة ضمن الأصناف، وهناك تسميات أخرى مثل "المعاملات" أو "الحقول"، ولكن سنستخدم التسمية الأولى "الخاصيات" في هذا الدليل. تعرّف الخاصيات باستخدام إحدى الكلمات المفتاحية public، أو protected، أو private تتبعها عبارة تصريح عن متغير اعتيادي. يمكن أن يتضمّن هذا التصريح عملية تهيئة initialization، ولكن يجب أن تكون هذه التهيئة ذات قيمة ثابتة، بمعنى أنّه يجب أن تكون قابلة للمعالجة في وقت التصريف ويجب أن لا تعتمد على المعلومات المتاحة في وقت التشغيل لغرض المعالجة.

## ثوابت الأصناف

يمكن تعريف قيم ثابتة لا يمكن تعديلها ولكل صنفٍ على حدة. تختلف الثوابت عن المتغيرات الاعتيادية في عدم استخدام العلامة `$` للتصريح عنها أو استخدامها. تمتلك ثوابت الأصناف قابلية رؤية من نوع public.

## التحميل التلقائي للأصناف

يُنشئ أغلب المطوّرين الذي يكتبون تطبيقات كائنية التوجه ملف PHP واحد لكل صنف. وهنا تظهر مشكلة مزعجة للغاية، وهي الحاجة إلى كتابة قائمة طويلة بالأصناف التي يجب تضمينها للبدء بتنفيذ الشيفرة (قائمة لكل صنف).

لم يعد هذا الأمر ضروريًا في الإصدار الخامس من اللغة، إذ تسجّل الدالة `spl_autoload_register()` عددًا غير محدّدٍ من المحمّلات التلقائية لتتيح إمكانية التحميل التلقائي للأصناف والواجهات غير المعرّفة في ملف الشيفرة. يمنح تسجيلُ المحمّلات التلقائية اللغةَ فرصة أخيرة لتحميل الصنف أو الواجهة قبل أن تتوقف الشيفرة عن العمل وتطلق خطأً.

## التوابع البانية والهادمة

تتيح لغة PHP للمطورين التصريح عن تابع بانٍ في الأصناف. كل صنف يمتلك تابعًا بانيًا يستدعي هذا التابع مع كل كائن جديد يُنشأ من ذلك الصنف؛ لهذا يعد هذا التابع ملائمًا لعمليات التهيئة التي قد يحتاج لها الكائن قبل استخدامه.

تقدّم PHP 5 مفهوم الدوال الهادمة وهو مفهوم شائع في لغات البرمجة كائنية التوجّه مثل C++. تُستدعى الدالة الهادمة عندما لا يكون هناك أي مرجع لكائن معين.

## قابلية الرؤية

يمكن تعريف قابلية رؤية خاصية أو تابع أو (بدءًا من الإصدار ٧,١,٠ من اللغة) ثابت بأنّها إلحاق صيغة التصريح بإحدى الكلمات المفتاحية public، أو protected، أو private. يتيح التصريح من نوع public الوصول إلى أعضاء الصنف من أي مكان، ويتيح التصريح من نوع protected الوصول إلى الأعضاء من داخل الصنف نفسه أو الأصناف التي ترث الصنف أو تورّثه، أما التصريح من نوع private فيتيح الوصول إلى الأعضاء من داخل الصنف الذي عرّف فيه ذلك العضو حصرًا.

## وراثة الكائنات

وراثة الكائنات من المبادئ المعروفة في البرمجة كائنية التوجه وتستخدمه PHP في نموذج الكائنات الخاصّ بها. يؤثّر هذا المبدأ على العلاقة التي تربط بين الأصناف والكائنات.

## عامل تحليل النطاق (::)

يسمح عامل تحليل النطاق (Scope Resolution Operator) بالوصول إلى الخاصيات والتوابع من نوع static، أو المتجاوز عليها أو الثوابت الخاصة بصنف معين.

## الكلمة المفتاحية Static

يمكن استخدام static أيضًا لتعريف المتغيرات الساكنة والروابط الساكنة اللاحقة ( late static bindings).

## تجريد الأصناف

يقدّم الإصدار الخامس من اللغة الأصناف والتوابع المجرّدة. لا يمكن تهيئة الأصناف المجرّدة، وإذا احتوى الصنف على تابع مجرّد واحدٍ على الأقل فيجب أن يكون الصنف مجردًا أيضًا. تصرّح الأصناف المجرّدة ببساطة عن توقيع التابع ( method's signature)، ولا يمكنها تعريف الاستخدام (implementation).

## واجهات الكائنات

تتيح واجهات الكائنات إنشاء شيفرة تحدّد التوابع التي يجب أن يتضمّنها الصنف دون الحاجة إلى تعريف آلية التحكّم في هذه التوابع.

تعرّف الواجهات بنفس طريقة تعريف الأصناف، ولكن باستخدام الكلمة المفتاحية interface بدل الكلمة المفتاحية class ودون تعريف محتوى أيّ تابعٍ في الواجهة.

## السمات

تعدّ السمات طريقة لإعادة استخدام الشيفرة في اللغات التي لا تدعم الوراثة المتعددة مثل PHP. وتهدف السمات إلى إزالة بعض القيود التي تفرضها الوراثة المفردة وذلك بتمكين المطوّر من استخدام مجموعة من التوابع بحرية في عدد من الأصناف المستقلة عن بعضها في هيكلية أصناف مختلفة. وتحمل كلّ من السمات والأصناف دلالات تهدف إلى التقليل من التعقيد وتجنب المشاكل الشائعة التي ترتبط بالوراثة المتعددة والدوال المساعدة Mixins.

## الأصناف المجهولة

أضيف دعم الأصناف المجهولة إلى الإصدار السابع من PHP، وهذا النوع من الأصناف مفيد عند الحاجة إلى إنشاء نسخة واحدة بسيطة من الصنف.

## التحميل الزائد

تقديم ميزة التحميل الزائد في PHP القدرة على إنشاء الخصائص والتوابع بصورة ديناميكية، وتعالج هذه العناصر الديناميكية بواسطة التوابع السحرية (magic methods) التي يمكن استخدامها في الصنف لأداء العديد من الوظائف.

## Object Iteration

يقدّم الإصدار الخامس من اللغة طريقة لتعريف العناصر ليكون بالإمكان المرور على قائمة من العناصر باستخدام العبارة `foreach` على سبيل المثال، وتستخدم جميع الخصائص المرئية بصورة افتراضية لعملية المرور على العناصر.

## التوابع السحرية

التوابع السحرية في أصناف PHP هي `__construct()` و `__destruct()` و `__call()` و `__callStatic()` و `__get()` و `__set()` و `__isset()` و `__unset()` و `__sleep()` و `__wakeup()` و `__toString()` و `__invoke()` و `__set_state()` و

`__clone()` و `__debugInfo()`. ولا يمكنك استخدام دوالّ تحمل هذه الأسماء في أي صنف ما لم تكن ترغب في الاستفادة من الوظيفة السحرية التي تتمتّع بها.

## الكلمة المفتاحية final

قدّم الإصدار الخامس من اللغة الكلمة المفتاحية final والتي تمنع الأصناف الأبناء من التجاوز على توابع الصنف الأب وذلك بإضافة الكلمة المفتاحية قبل عبارة تعريف التابع. إذا استُخدمت الكلمة المفتاحية `final` في تعريف الصنف فإنّه يصبح غير قابل للتوسع.

## استنساخ الكائنات

أي نسخ الكائن مع خاصياته كلها.

## مقارنة الكائنات

يمكن إجراء مقارنة بسيطة بين متغيرات الكائنات باستخدام عامل المقارنة (==)، وتكون نسختا الكائن متساويتين إن كانتا تملكان نفس المعاملات ونفس القيم (تقارن القيم باستخدام ==) وكانتا نسختين لنفس الصنف.

عند استخدام عامل التطابق (===) تكون متغيرات الكائنات متطابقة عندما تشير فقط وفقط إذا إلى النسخة نفسها من الصنف نفسه.

## التلميح عن الأنواع

تسمح خاصية التصريح عن الأنواع للدوال بطلب النوع المحدّد من المعاملات في وقت الاستدعاء، وإن كانت القيمة المستدعاة ذات نوع مغاير، تطلق اللغة خطأ من نوع recoverable fatal في الإصدار ٥، أما في الإصدار ٧ فترمي اللغة استثناءً من نوع `TypeError`.

ولتحديد النوع الذي ترغب في التصريح عنه، يجب إضافة اسم النوع قبل اسم المعامل، ويمكن قبول قيم NULL وذلك بتعيين NULL كقيمة افتراضية للمعاملات.

## الروابط الساكنة المتأخرة

تضمن الإصدار ٥,٣,٠ من PHP خاصية تدعى بالروابط الساكنة المتأخرة ( late static bindings) والتي يمكن استخدامها للإشارة إلى الصنف المستدعى في سياق وراثة ساكنة.

## الكائنات والمراجع

المرجع في PHP هو اختصار يتيح لمتغيرين مختلفين الكتابة إلى القيمة ذاتها. ومنذ الإصدار الخامس من اللغة، لم يعد متغير الكائن يتضمّن الكائن نفسه كقيمة على الإطلاق، بل يتضمّن فقط معرّفًا للكائن يتيح الوصول إلى الكائن الحقيقي. وعند تمرير الكائن كمعامل أو إعادته أو

إسناده إلى متغير آخر، لا تكون هذه المتغيرات أسماء بديلة (aliases) بل تحمل نسخة من المعرّف الذي يشير إلى الكائن نفسه.

## سَلسَلة الكائن

تعيد الدالة `serialize()` سلسلة نصية تتضمّن تمثيلًا لتدفق البايتات لأي قيمة يمكن تخزينها في PHP، ويمكن للدالة `unseriazlie()` استخدام هذه السلسلة النصية لإعادة إنشاء القيمة الأصلية للمتغيّر. يؤدي تطبيق الدالة `serialize()` على كائن ما لحفظ جميع المتغيرات الخاصة بذلك الكائن، أما التوابع فلا تحفظ وإنّما يحفظ اسم الصنف فقط.

## سجل التغييرات

يعرض الجدول التالي سجلّ التغييرات الحاصلة على نموذج البرمجة كائنية التوجه في الإصدار الخامس من اللغة. يمكنك أن تجد التفاصيل والملاحظات المتعلّقة بهذه الميزات في الأقسام الخاصة بالبرمجة كائنية التوجّه في هذا الدليل.

| الإصدار | الوصف |
| --- | --- |
| 7.0.0 | لا يتسبب تعريف الخواص (المتوافقة) في سمتين مستخدمتين في إطلاق خطأ. |
| 5.6.0 | إضافة التابع `__debugInfo()` |
| 5.5.0 | إضافة الثابت السحري `::class` |
| 5.5.0 | إضافة الكلمة المفتاحية `finally` للتعامل مع الاستثناءات |
| 5.4.0 | إضافة السمات.(traits) |
| 5.4.0 | تعديل: إنْ عرّفَ صنفٌ مجردٌ توقيعًا للدالة البانية constructor فإنّها ستفرض الآن. |

| | |
|---|---|
| 5.3.3 | تعديل: لا تعامل التوابع التي تحمل نفس اسم العنصر الأخير لصنف ينتمي إلى مجال أسماء كدالة بانية. لا يؤثر هذا التعديل على الأصناف غير المنتمية إلى مجال أسماء. |
| 5.3.0 | تعديل: لم تعد الأصناف التي تطبق واجهات مع توابع تمتلك قيمًا افتراضية في النموذج الأولي مطلوبة لمطابقة القيمة الافتراضية للواجهة. |
| 5.3.0 | تعديل: أصبح بالإمكان الإشارة إلى صنف بواسطة متغير (مثال `echo $classname::constant;`). لا يمكن أن تكون قيمة المتغير هذا كلمة مفتاحية (مثل `self` ، أو `parent` ، أو `static`). |
| 5.3.0 | تعديل: ينطلق خطأ من نوع E_WARNING عند تعريف توابع إعادة التعريف السحرية كتوابع ساكنة .static كذلك تصبح قابلية الرؤية من نوع public أمرًا إلزاميًا. |
| 5.3.0 | تعديل: قبل هذا الإصدار لم تكن الاستثناءات المرمية في دالة `__autoload()` تلتقط في كتلة `catch` ، وكانت تؤدي إلى حدوث خطأ من نوع .fatal أصبحت الاستثناءات المرمية في دالة `__autoload` تلتقط في كتلة `catch` بشرط واحد. في حال رمي استثناء خاص، فإنّ صنف الاستثناء الخاص يجب أن يكون معرّفًا. يمكن استخدام الدالة `__autoload` تعاوديًا لتحميل صنف الاستثناء الخاص تلقائيًا. |
| 5.3.0 | إضافة التابع `__callStatic`. |
| 5.3.0 | إضافة دعم كتلة heredoc و nowdoc في تعريف ثوابت وخواص الأصناف.<br>ملاحظة: يجب أن تتبع قيم heredoc نفس القواعد المفروضة على السلاسل النصّية المحاطة بعلامات اقتباس مزدوجة (مثل عدم استخدام المتغيرات). |
| 5.3.0 | إضافة الروابط الساكنة اللاحقة |
| 5.3.0 | إضافة التابع `__invoke()` |
| 5.2.0 | تعديل: كان التابع `__toString()` يستدعى فقط عند دمجه مباشرة مع `echo` أو `print`. ولكن أصبح بالإمكان الآن استدعاؤه ضمن أي سياق نصّي (مثال `printf()`: مع المعدِّل `%s`) ولكن ليس ضمن أي سياق من نوع آخر (مثال: المعدّل `%d`). منذ الإصدار ٥,٢,٠ من اللغة يؤدي تحويل كائن لا يتضمّن التابع |

| | |
|---|---|
| | `__toString` إلى سلسلة نصية إلى إطلاق خطأ من المستوى E_RECOVERABLE_ERROR. |
| 5.1.3 | تعديل: أهملت الإصدارات السابقة للإصدار الخامس من اللغة استخدام الكلمة المفتاحية `var` ، ويؤدي استخدامها إلى إطلاق خطأ من المستوى E_STRICT. لم يعد استخدام هذه الكلمة مهملًا في هذا الإصدار ولن يؤدي استخدامها إلى إطلاق خطأ. |
| 5.1.0 | تعديل: يستدعى التابع الساكن `__set_state()` للأصناف المصدّرة بواسطة الدالة `var_export()`. |
| 5.1.0 | إضافة التابعين `__isset()` و `__unset()` |

## مصادر

- صفحة OOP في توثيق PHP الرسمي.
- صفحة OOP Introduction في توثيق PHP الرسمي.