**Self vs This vs Static**

self و $this و static في PHP بتستخدم في الـ classes عشان توصل للـ properties والـ methods في الـ class.

self بتستخدم لو عاوز توصل للـ static properties و static methods في الـ class اللي انت فيه دلوقتي.

$this بتستخدم لو عاوز توصل للـ properties و methods اللي في الـ instance اللي انت فيه دلوقتي.

static بتستخدم لو عاوز توصل للـ static properties و static methods في أي class، مش

بس الـ class اللي انت فيه دلوقتي.

class MyClass {

public static $staticVar = "Hello, I'm static!";

public $instanceVar = "Hello, I'm an instance!";

public function myMethod() {

echo self::$staticVar; // يطبع "Hello, I'm static!"

echo $this->instanceVar; // يطبع "Hello, I'm an instance!"

echo static::$staticVar; // يطبع "Hello, I'm static!"

}

}

**OOP vs functional Vs procedural programming**

**procedural**

هي طريقة تتميز بتتبع خطوات محددة ورا بعضاها عاديه الإجراءات اللي لازم نعملها لحل المشكلة المطلوبة. ويكون البرنامج مجموعة أوامر. مثال: لو كان لدينا برنامج يحسب مجموع 3 أعداد، فيتم البرمجة بهذه الطريقة

$a = 5;

$b = 10;

$c = 15;

$sum = $a + $b + $c;

echo "The sum is: " . $sum;

**functional**

بتركز على استخدام الدوال بشكل رئيسي في البرمجة. بتكون الدوال مستقلة عن البيانات التانيه ومبتحتاجش متغيرات عامة، وبتتعمل باستخدام مجموعة من الدوال لحل المشكلة المطلوبة. مثال: لو كان لدينا برنامج يحسب مجموع 3 أعداد، فيتم البرمجة بهذه الطريقة.

function calculateSum($a, $b, $c) {

return $a + $b + $c;

}

$sum = calculateSum(5, 10, 15);

echo "The sum is: " . $sum;

OOP

هي طريقة بتستخدم مفهوم الـ classes والـ objects للتعامل مع البيانات والعمليات، وبتقسم البرنامج إلى مجموعة من الـ classes اللي بتتعامل مع بعضها البعض، حيث يتم تعريف الـ classes بالـ attributes والـ methods الخاصة بها. مثال: لو كان عندنا برنامج يحسب مجموع 3 أعداد، فيتم البرمجة بهذه الطريقة.

class Calculator {

public function calculateSum($a, $b, $c) {

return $a + $b + $c;

}

}

$obj = new Calculator();

$sum =

**Cloning in php ( shallow copy and deep copy)**

الـ cloning في PHP هو عملية إنشاء نسخة من الـ object، ويوجد نوعان من الـ cloning: shallow copy و deep copy.

الـ shallow copy:

ده بينسخ الـ object الأصلي وينشئ نسخة منه، لكن الـ properties اللي هي objects هي نفس الـ reference اللي بيشير لهم الـ object الأصلي. يعني لو غيرنا حاجة في الـ property ده هيتغير في الـ object الأصلي والنسخة اللي اتعملت بعد الـ cloning.

الـ deep copy:

ده بينشئ نسخة جديدة من الـ object الأصلي ونسخ كل الـ properties اللي هي objects، لكنه ينشئ objects جدد بدلاً من استخدام الـ reference اللي بيشير لهم الـ object الأصلي. بالتالي، إذا قمنا بتعديل أي من الـ properties الجديدة المنشئة في الـ object المنسوخ، فلن يؤثر ذلك على الـ object الأصلي.

**Magic methods ( get - set - call - call static ...etc)**

الـ Magic methods ده هي دوال خاصة في PHP بتبدأ بالكلمة "\_\_"، وده بيعني إنهم مبنيين داخلياً في اللغة. بيستخدموا لحل مشاكل كتير في الـ Object Oriented Programming.

\_\_get و \_\_set:

ده بيسمح لينا بالتحكم في properties الـ object اللي ممكن نوصل ليهم من خلال الـ syntax العادية اللي هي object->property. يعني لو عندي class اسمها Person وفيها property اسمها "name"، وعايز أعدل فيها من براها، ممكن استخدم الـ \_\_get والـ \_\_set دول. مثال

class Person {

private $name;

public function \_\_get($property) {

if ($property === 'name') {

return $this->name;

}

}

public function \_\_set($property, $value) {

if ($property === 'name') {

$this->name = $value;

}

}

}

$person = new Person();

$person->name = 'Ahmed'; // \_\_set called

echo $person->name; // \_\_get called, output: Ahmed

\_\_call و \_\_callStatic:

ده بيسمح لينا بتحكم في methods الـ object اللي مش موجودة، يعني لو أحنا عندنا method مش موجودة في class، ممكن نستخدم الـ \_\_call والـ \_\_callStatic عشان نتحكم في الـ method ده. مثال:

class Person {

public function \_\_call($method, $arguments) {

if ($method === 'greet') {

return "Hello, " . $arguments[0];

}

}

public static function \_\_callStatic($method, $arguments) {

if ($method === 'greet') {

return "Hello, " . $arguments[0];

}

}

}

$person = new Person();

echo $person->greet('Ahmed'); // \_\_call called, output: Hello, Ahmed

echo Person::greet('Ali'); // \_\_callStatic called, output: Hello, Ali

بالتالي، الـ Magic methods ده بيعملوا حياتنا في PHP أسهل وأكتر منظمة في الـOOP.