الاسم: الحسن عبد الحميد مقبل المليكي

تحت إشراف : م- أ / إبراهيم الشامي

PHP 8، الذي تم إصداره في 26 نوفمبر 2020، جاء بالعديد من الميزات الجديدة والتحسينات التي تهدف إلى تحسين الأداء، وتسهيل كتابة الكود، وإضافة ميزات جديدة للغة.

فيما يلي نظرة مفصلة على أبرز الإضافات والمميزات في PHP 8 مع أمثلة لكل خاصية:

1- JIT (Just-In-Time) Compilation

- الوصف: JIT هي تقنية تحويل الكود البرمجي إلى كود آلي (machine code) في وقت التنفيذ، مما يحسن أداء التطبيقات التي تعتمد على العمليات الحسابية المكثفة.

- الأهمية: يحسن أداء PHP بشكل ملحوظ، خاصة في التطبيقات التي تتطلب معالجة كبيرة.

- مثال:

```php
// لا يوجد تغيير في كتابة الكود، ولكن الأداء يتحسن تلقائيًا.
function calculate($a, $b) {
    return $a * $b + $a / $b;
}
echo calculate(10, 5); // الناتج: 52
```

2- Union Types

- الوصف: إمكانية تحديد أكثر من نوع للبيانات (type) للمتغيرات أو معاملات الدوال.

- الأهمية: يزيد من مرونة كتابة الكود ويقلل الحاجة إلى التحقق من الأنواع يدويًا.

- مثال:

```php
  function processInput(int|string $input): void {

    echo "Input: " . $input;
}
processInput(42);      // الناتج: Input: 42
processInput("Hello"); // الناتج: Input: Hello
```

3- Named Arguments

- الوصف: إمكانية تمرير المعاملات (arguments) للدوال باستخدام أسماء المعاملات بدلًا من الاعتماد على الترتيب.

- الأهمية: يجعل الكود أكثر قابلية للقراءة ويقلل الأخطاء عند استدعاء الدوال.

- مثال:

```php
function createUser(string $name, int $age, bool $isAdmin = false): void {
    echo "User: $name, Age: $age, Admin: " . ($isAdmin ? 'Yes' : 'No');
}
createUser(age: 23, name: "Hasan"); // الناتج: User: Hasan, Age: 23, Admin: No
```

4- Attributes (السمات)

- الوصف: إضافة بيانات وصفية (metadata) للفئات، الدوال، أو الخصائص باستخدام صيغة جديدة بدلًا من التعليقات.

- الأهمية: يحسن تنظيم الكود ويجعل البيانات الوصفية قابلة للاستخدام برمجيًا.

- مثال:

```php
   #[Attribute]
class Route {
    public function __construct(public string $path) {}
}

#[Route('/home')]
class HomeController {
    // ...
}
```

5- Constructor Property Promotion

- الوصف: اختصار لتعريف الخصائص (properties) وتعيينها في المُنشئ (constructor) في سطر واحد.
- الأهمية: يقلل من تكرار الكود ويجعل تعريف الخصائص أكثر كفاءة.
- مثال:

```php
class User {
    public function __construct(
        private string $name,
        private int $age
    ) {}
}
$user = new User("Hasan", 23);
```

6- Match Expression

- الوصف: بديل أكثر قوة ومرونة لعبارة `switch`، مع إرجاع قيمة مباشرة.
- الأهمية: يجعل الكود أكثر وضوحًا ويقلل الأخطاء.
- مثال:

```php
 $status = 200;

$message = match ($status) {
    200, 201 => 'Success',
    404 => 'Not Found',
    500 => 'Server Error',
    default => 'Unknown',
};
echo $message; // الناتج: Success
```

7- Nullsafe Operator

- الوصف: يسمح بالوصول إلى خصائص أو استدعاء دوال على كائن مع التحقق من وجوده (null) دون الحاجة إلى تحقق يدوي.

- الأهمية: يقلل من تعقيد الكود عند التعامل مع القيم الفارغة.

- مثال:

```
$user = null;
echo $user?->profile?->name ?? 'Guest'; // الناتج: Guest
```

8- تحسينات في نظام الأنواع (Type System)

- الوصف: إضافة أنواع جديدة مثل ``mixed، static``، و`return`، وتحسين دعم الأنواع الحالية.

- الأهمية: يجعل نظام الأنواع أكثر قوة ومرونة.

- مثال:

```
   function process(mixed $input): void {

    if (is_string($input)) {
        echo "String: $input";
    } elseif (is_int($input)) {
        echo "Integer: $input";
    }
}
process("Hello"); // الناتج: String: Hello
process(42);      // الناتج: Integer: 42
```

9- تحسينات في الأخطاء والاستثناءات

- الوصف: تحويل العديد من الأخطاء إلى استثناءات (exceptions) مثل `TypeError` و`ValueError`.

- الأهمية: يجعل التعامل مع الأخطاء أكثر مرونة.

- مثال:

```
try {
    $result = substr(null, 0, 5);
} catch (TypeError $e) {
    echo "Error: " . $e->getMessage(); // الناتج: Error: substr(): Argument #1
($string) must be of type string, null given
}
```

10- تحسينات في الأداء

- الوصف: تحسينات عامة في أداء اللغة، بما في ذلك تحسينات في إدارة الذاكرة وتنفيذ الكود.
- الأهمية: زيادة سرعة تنفيذ التطبيقات.
- مثال:

```
// لا يوجد تغيير في الكود، ولكن الأداء يتحسن تلقائيًا.
for ($i = 0; $i < 1000000; $i++) {
    // عمليات مكثفة
}
```

11- تحسينات في الدوال المضمنة (Built-in Functions)

- الوصف: إضافة دوال جديدة مثل ``str_contains``، ``str_starts_with`` و`str_ends_with`.
- الأهمية: يسهل التعامل مع النصوص.
- مثال:

```
if (str_contains("Hello World", "World")) {
    echo "Found!"; // الناتج: Found!
}
```

12- تحسينات في التعامل مع السلاسل النصية

- الوصف: تحسينات في معالجة النصوص، مثل دعم Unicode بشكل أفضل.
- الأهمية: يجعل التعامل مع النصوص أكثر قوة.
- مثال:

```
echo strtoupper('i love php'); // الناتج: I LOVE PHP
```

13- تحسينات في التعامل مع المصفوفات

- الوصف: إضافة دوال جديدة مثل `array_is_list` للتحقق من أن المصفوفة هي قائمة (list).
- الأهمية: يسهل التعامل مع المصفوفات.
- مثال:

```
$array = [1, 2, 3];
var_dump(array_is_list($array)); // الناتج: bool(true)
```

14- تحسينات في التعامل مع التواريخ والوقت

- الوصف: تحسينات في دوال مثل `DateTime` و`DateTimeImmutable`.
- الأهمية: يجعل التعامل مع التواريخ أكثر مرونة.
- مثال:

```
$date = new DateTime('2023-10-01');
echo $date->format('Y-m-d'); // 01-10-2023 :الناتج
```

15- تحسينات في التعامل مع الملفات

- الوصف:تحسينات في دوال مثل `fopen` و`file_get_contents`.
- الأهمية: يجعل التعامل مع الملفات أكثر كفاءة.
- مثال:

```
$content = file_get_contents('example.txt');
echo $content;
```

16- تحسينات في التعامل مع قواعد البيانات

- الوصف: تحسينات في دوال مثل `mysqli` و`PDO`.
- الأهمية: يجعل التعامل مع قواعد البيانات أكثر أمانًا وكفاءة.
- مثال:

```
$pdo = new PDO('mysql:host=localhost;dbname=test', 'user', 'pass');
$stmt = $pdo->query('SELECT * FROM users');
```

17- تحسينات في التعامل مع الشبكات

- الوصف: تحسينات في دوال مثل `curl` و`socket`.

- الأهمية: يجعل التعامل مع الشبكات أكثر مرونة.

- مثال:

```
$ch = curl_init('https://example.com');
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
$response = curl_exec($ch);
curl_close($ch);
echo $response;
```

18- تحسينات في التعامل مع الذاكرة

- الوصف: تحسينات في إدارة الذاكرة، مثل تقليل استخدام الذاكرة في بعض الحالات.
- الأهمية: يجعل التطبيقات أكثر كفاءة.
- مثال:

```
// لا يوجد تغيير في الكود، ولكن استخدام الذاكرة يتحسن تلقائيًا.
$array = range(1, 1000000);
```

19- تحسينات في التعامل مع الأخطاء

- الوصف: تحسينات في التعامل مع الأخطاء، مثل تحسين رسائل الأخطاء.
- الأهمية: يجعل تصحيح الأخطاء أسهل.
- مثال:

```
try {
    $result = 10 / 0;
} catch (DivisionByZeroError $e) {
    echo "Error: " . $e->getMessage(); // الناتج: Error: Division by zero
}
```

20- تحسينات في التعامل مع الاستثناءات

- الوصف: تحسينات في التعامل مع الاستثناءات، مثل تحسين رسائل الاستثناءات.
- الأهمية: يجعل التعامل مع الاستثناءات أكثر مرونة.
- مثال:

```
try {
    throw new Exception('Something went wrong');
} catch (Exception $e) {
    echo "Exception: " . $e->getMessage(); // الناتج: Exception: Something went
wrong
}
```

دوال الوقت والتاريخ في PHP

1- date()

- الوصف: تُستخدم لعرض التاريخ والوقت بتنسيق معين.
- المعاملات:
- `format`: التنسيق المطلوب (مثل `Y-m-d` لعرض السنة-الشهر-اليوم).
- مثال:

```
echo date('Y-m-d H:i:s'); // 14:30:00 01-10-2023 :الناتج
```

2- time()

- الوصف:تُرجع الطابع الزمني الحالي (عدد الثواني منذ 1 يناير 1970).
- مثال:

```
echo time(); // (على سبيل المثال) 1696162200 :الناتج
```

3- strtotime()

- الوصف: تحويل نص تاريخي إلى طابع زمني.
- مثال:

```
echo strtotime('next Monday'); // الناتج: الطابع الزمني ليوم الاثنين القادم
```

4- mktime()

- الوصف: إنشاء طابع زمني من قيم محددة (ساعة، دقيقة، ثانية، شهر، يوم، سنة).

- مثال:

```
echo mktime(0, 0, 0, 10, 1, 2023); // 00:00:00 01-10-2023 الناتج: الطابع الزمني لتاريخ
```

5- DateTime` Class

- الوصف: يوفر كائن `DateTime` طريقة كائنية التوجه للتعامل مع التواريخ والأوقات.

- مثال:

```
$date = new DateTime('2023-10-01');
echo $date->format('Y-m-d H:i:s'); // 00:00:00 01-10-2023 :الناتج
```

6- date_diff()

- الوصف: حساب الفرق بين تاريخين.

- مثال:

```
$date1 = new DateTime('2023-10-01');
$date2 = new DateTime('2023-10-10');
$interval = date_diff($date1, $date2);
echo $interval->days; // 9 :الناتج
```

7- date_add() و date_sub()`

- الوصف: إضافة أو طرح فترة زمنية من تاريخ.

- مثال:

```
$date = new DateTime('2023-10-01');
$date->add(new DateInterval('P10D')); // إضافة 10 أيام
echo $date->format('Y-m-d'); // 11-10-2023 :الناتج
```

8- strftime()

- الوصف: تنسيق التاريخ والوقت بناءً على الإعدادات المحلية.

- مثال:

```
setlocale(LC_TIME, 'ar_AE.utf8'); // تعيين اللغة العربية
echo strftime('%A %d %B %Y'); // الناتج: الأحد 01 أكتوبر 2023
```

9- checkdate()

- الوصف: التحقق من صحة تاريخ معين.

- مثال:

```
var_dump(checkdate(2, 29, 2023)); // الناتج: bool(false) لأن 2023 ليست سنة كبيسة
```

10- date_default_timezone_set()

- الوصف: تعيين المنطقة الزمنية الافتراضية.

- مثال:

```
  date_default_timezone_set('Asia/Riyadh');

echo date('Y-m-d H:i:s'); // الناتج: الوقت الحالي في الرياض
```

التعابير النظامية (Regular Expressions) في PHP

التعابير النظامية هي أدوات قوية لمطابقة النصوص واستخراجها بناءً على أنماط معينة. في PHP، يمكن استخدام التعابير النظامية عبر دوال مثل ``()preg_match`، `()preg_replace`، وغيرها.

1- preg_match()

- الوصف: البحث عن نمط معين في نص.
- مثال:

```
   $text = "Hello, World!";

if (preg_match('/World/', $text)) {
    echo "Match found!";
} else {
    echo "No match!";
}
// الناتج: Match found!
```

2- preg_replace()

- الوصف: استبدال نص مطابق لنمط معين.
- مثال:

```
$text = "Hello, World!";
$newText = preg_replace('/World/', 'PHP', $text);
echo $newText; // الناتج: Hello, PHP!
```

3- preg_split()

- الوصف: تقسيم نص إلى مصفوفة بناءً على نمط معين.

- مثال:

```php
$text = "apple,banana,orange";
$fruits = preg_split('/,/', $text);
print_r($fruits);
// الناتج: Array ( [0] => apple [1] => banana [2] => orange )
```

4- preg_match_all()

- الوصف: البحث عن جميع التطابقات لنمط معين في نص.

- مثال:

```php
$text = "cats and dogs";
preg_match_all('/\w+/', $text, $matches);
print_r($matches);
// الناتج: Array ( [0] => Array ( [0] => cats [1] => and [2] => dogs ) )
```

5- preg_quote()

- الوصف: إضافة إشارات الهروب إلى الأحرف الخاصة في النص لجعله آمنًا للاستخدام في التعابير النظامية.

- مثال:

```php
$text = "Hello. World!";
echo preg_quote($text); // الناتج: Hello\. World\!
```

6- أنماط شائعة في التعابير النظامية

- `\d`: أي رقم (0-9).
- `\w`: أي حرف أو رقم أو شرطة سفلية.
- `\s`: أي مسافة بيضاء (فراغ، تبويب، سطر جديد).
- `+`: تطابق واحد أو أكثر من العنصر السابق.
- `*`: تطابق صفر أو أكثر من العنصر السابق.
- `?`: تطابق صفر أو واحد من العنصر السابق.
- `{n}`: تطابق عدد محدد من العناصر (مثل `{3}` لتطابق 3 عناصر).