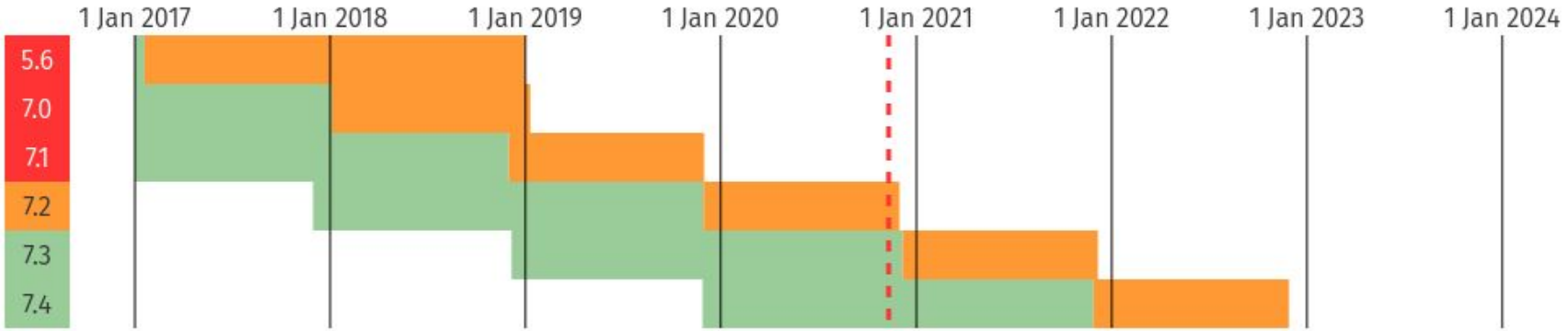
Introduction to PHP 8

Temuri Takalandze

@ABGEO

Timeline



https://www.jetbrains.com/lp/php-25/

About PHP 8

- General
- About types
- What's new in OOP?
- New built-in functions
- Breaking changes

General

JIT

RFC https://wiki.php.net/rfc/jit

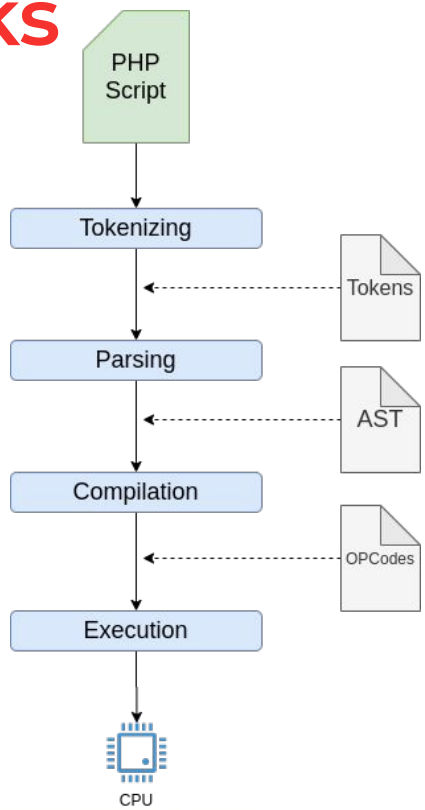
JIT - just in time - Compiler

“PHP JIT is implemented as an almost independent part of OPcache. It may be enabled/disabled at PHP compile time and at run-time. When enabled, native code of PHP files is stored in an additional region of the OPcache shared memory and op_array→opcodes[].handler(s) keep pointers to the entry points of JIT-ed code.”

How PHP works

- **Tokenizing**: First, the interpreter reads the PHP code and builds a set of tokens.
- **Parsing**: The interpreter checks if the script matches the syntax rules and uses tokens to build an AST.
- **Compilation**: The interpreter traverses the tree and translates AST nodes into low-level Zend opcodes.
- **Interpretation**: Opcodes are interpreted and run on the Zend VM.

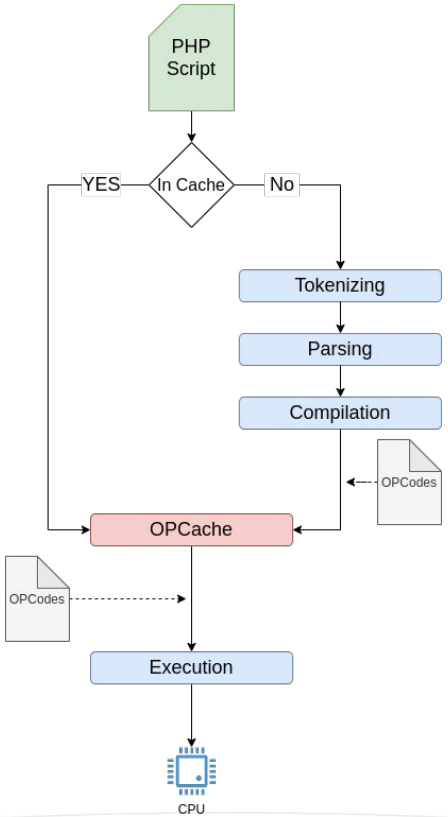
How PHP works



OPcache

“OPcache improves PHP performance by storing precompiled script bytecode in shared memory, thereby removing the need for PHP to load and parse scripts on each request.”

OPcache


PHP Script
In Cache
YES
No
Tokenizing
Parsing
Compilation
OPCodes
OPCache
OPCodes
Execution
CPU

Preloading

Allows you to tell PHP FPM to parse your codebase, transform it into Opcodes and cache them even before you execute anything.

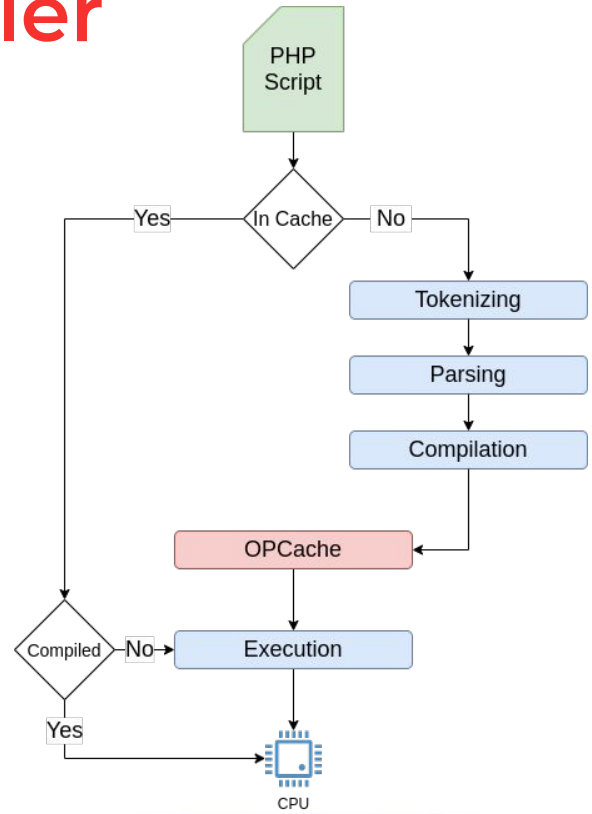
JIT Compiler

Even if opcodes are in the form of low-level intermediate representation, they still have to be compiled into machine code.

JIT uses DynASM (Dynamic Assembler for code generation engines) to generate native code directly from PHP byte-code.

With JIT enabled, the code wouldn’t be run by the Zend VM, but by the CPU itself, and this would improve speed in calculation.

JIT Compiler



Attributes

RFC https://wiki.php.net/rfc/attributes_v2

```php
#[Attribute]
class ExampleAttribute
{
    public $value;

    public function __construct($value)
    {
        $this->value = $value;
    }
}
```

```php
use App\Attributes\ExampleAttribute;

#[ExampleAttribute]
class Foo
{
    #[ExampleAttribute]
    public const FOO = 'foo';

    #[ExampleAttribute]
    public $foo;

    #[ExampleAttribute]
    public function bar(#[ExampleAttribute] $baz) { }
}
```

Named arguments

RFC https://wiki.php.net/rfc/named_params

```php
function foo(string $foo, ?string $bar = null, ?string $baz = null) {}

foo(
    foo: 'value of foo',
    baz: 'value of baz',
    bar: 'value of bar'
);
```

The nullsafe operator

RFC https://wiki.php.net/rfc/nullsafe_operator

```php
$formattedReturnDate = $order->returnDate ? $order->returnDate->format('d/m/Y') : null;

// VS

$formattedReturnDate = $order->returnDate?->format('d/m/Y');
```

Match expression

RFC https://wiki.php.net/rfc/match_expression_v2

```php
$dayType = match($day) {
    1, 2, 3, 4, 5 => 'Weekday',
    6, 7 => 'Weekend',
    default => 'Invalid day number',
};
```

Throw expression

RFC https://wiki.php.net/rfc/throw_expression

```php
$triggerError = fn () => throw new Exception();

// ------------------------------------------------------------------

$item = $data[$key] ?? throw new Exception("Invalid key '{$key}'!");
```

Non-capturing catches

RFC https://wiki.php.net/rfc/non-capturing_catches

```
try {
    // Something goes wrong.
} catch (Exception) {
    Loger::error('Something went wrong');
}
```

Trailing comma in parameter lists

RFC https://wiki.php.net/rfc/trailing_comma_in_parameter_list

```php
function doSomething(
    Foo $foo,
    Bar $bar,
    Baz $baz,
) {
    // Do something ...
}
```

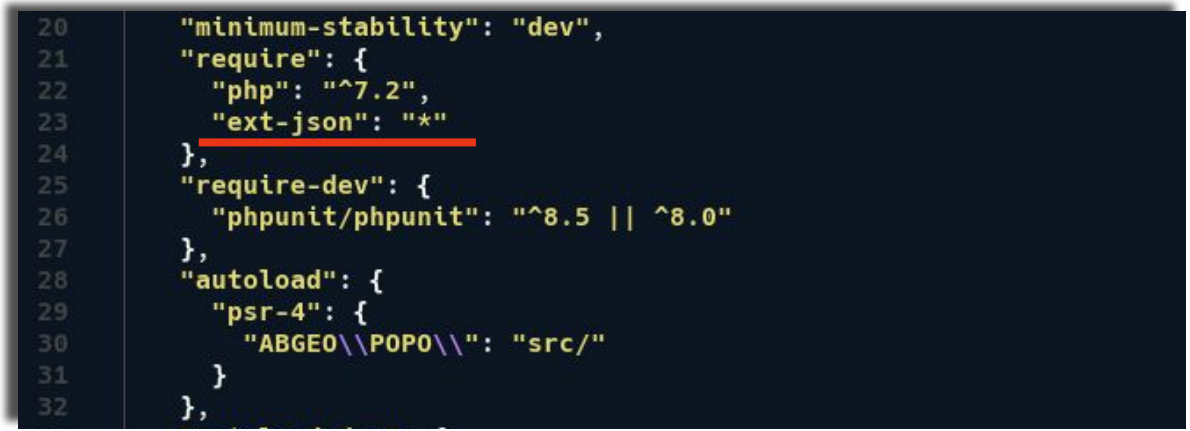
Concatenation precedence

RFC https://wiki.php.net/rfc/concatenation_precedence

```php
echo 'sum: ' . $a + $b;

echo ('sum: ' . $a) + $b;

// VS

echo 'sum: ' . ($a + $b);
```

ext-json always available

RFC https://wiki.php.net/rfc/always_enable_json



```
20    "minimum-stability": "dev",
21    "require": {
22      "php": "^7.2",
23      "ext-json": "*"
24    },
25    "require-dev": {
26      "phpunit/phpunit": "^8.5 || ^8.0"
27    },
28    "autoload": {
29      "psr-4": {
30        "ABGEO\\POPO\\": "src/"
31      }
32    },
```

About types

Union types

RFC https://wiki.php.net/rfc/union_types_v2

```php
public function foo(Foo|Bar $input): int|float {}
```

Mixed type

RFC https://wiki.php.net/rfc/mixed_type_v2

- array
- bool
- callable
- int
- float
- null
- object
- resource
- string

The mixed type can also be used as a parameter or property type, not just as a return type.

Since mixed already includes null, it's not allowed to make it nullable. The following will trigger an error:

Static return type

RFC https://wiki.php.net/rfc/static_return_type

```php
class Foo
{
    public function bar(): static
    {
        return new static();
    }
}
```

New Stringable interface

RFC https://wiki.php.net/rfc/stringable

```php
class Foo
{
    public function __toString(): string
    {
        return 'Hello from ' . __METHOD__;
    }
}

function doEcho(string|Stringable $stringable): void { /* echo */ }

doEcho('Hello');
doEcho(new Foo());
```

What’s new in OOP?

Constructor property promotion

RFC https://wiki.php.net/rfc/constructor_promotion

```php
class Person
{
    public function __construct(
        public string $firstName,
        public string $lastName,
    ) {}
}
```

Inheritance with private methods

RFC https://wiki.php.net/rfc/inheritance_private_methods

```php
class Foo
{
    private function baz() {}
}

class Bar extends Foo
{
    private function baz() {}
}
```

Allowing ::class on objects

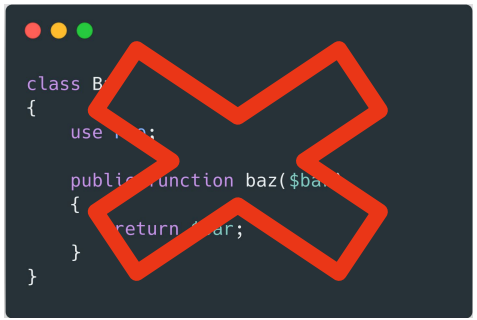
RFC https://wiki.php.net/rfc/class_name_literal_on_object

```php
$foo = new Foo();

var_dump(get_class($foo));

// VS

var_dump($foo::class);
```

Abstract trait method validation

RFC https://wiki.php.net/rfc/abstract_trait_method_validation

```php
trait Foo {
    abstract public function baz(int $bar): int;
}
```



```php
class Bar
{
    use Foo;

    public function baz(int $bar): int
    {
        return $bar;
    }
}
```

Weak maps

RFC https://wiki.php.net/rfc/weak_maps

```php
class Foo
{
    private WeakMap $cache;

    public function getSomethingWithCaching(object $obj)
    {
        return $this->cache[$obj] ??= $this->computeSomethingExpensive($obj);
    }

    // ...
}
```

Token as object

RFC https://wiki.php.net/rfc/token_as_object

```php
$tokens = token_get_all($code);

// VS

$tokens = PhpToken::tokenize($code);
```

New internal (built-in) functions

str_contains()

RFC https://wiki.php.net/rfc/str_contains

```php
if (strpos('Hello, World!', 'World') !== false) { /* ... */ }

// VS

if (str_contains('Hello, World!', 'World')) { /* ... */ }
```

str_starts_with() & str_ends_with()

RFC https://wiki.php.net/rfc/add_str_starts_with_and_ends_with_functions

```php
str_starts_with('haystack', 'hay');   // true
str_starts_with('haystack', 'stack'); // false

str_ends_with('haystack', 'hay');     // false
str_ends_with('haystack', 'stack');   // true
```

fdiv()

```php
$safeDivision = fdiv(5, 0);
```

get_debug_type()

RFC https://wiki.php.net/rfc/get_debug_type

```php
gettype($foo);

// VS

get_debug_type($foo);
```

get_resource_id()

```php
$resourceId = (int) $resource;

// VS

$resourceId = get_resource_id($resource);
```

Breaking changes

Breaking changes

- Methods with the same name as the class are no longer interpreted as constructors.
- Removed ability to call non-static methods statically.
- Removed each(). foreach or ArrayIterator should be used instead.
- Removed ability to use array_key_exists() with objects. Use one of isset() or property_exists() instead.

https://github.com/php/php-src/blob/PHP-8.0/UPGRADING#L20

Reclassified engine warnings

RFC https://wiki.php.net/rfc/engine_warnings

- Undefined variable: Error exception instead of notice;
- Undefined array index: warning instead of notice;
- Division by zero: DivisionByZeroError exception instead of warning;
- Undefined property: %s::$%s: warning instead of notice;
- Array to string conversion: warning instead of notice;

The default error reporting level is now E_ALL. This means that many errors might pop up which were previously silently ignored.

Questions ?

```php
function sayThanks(
    string $foo,
    string $bar,
): void {
    echo "{$foo} {$bar}!\n";
}

sayThanks(
    foo: 'Thank',
    bar: 'you',
);
```