Looking forward to PHP 8

Nikita Popov

JET
BRAINS

PS

Release schedule (tentative)

May ‘19	Now
June ‘19	PHP 7.4 Alpha 1
August ‘19	PHP 7.4 Beta 1 – Feature freeze
December ‘19	PHP 7.4 Release

Release schedule (tentative)

May ‘19	Now
June ‘19	PHP 7.4 Alpha 1
August ‘19	PHP 7.4 Beta 1 – Feature freeze
December ‘19	PHP 7.4 Release
December ‘20	PHP 8.0 Release

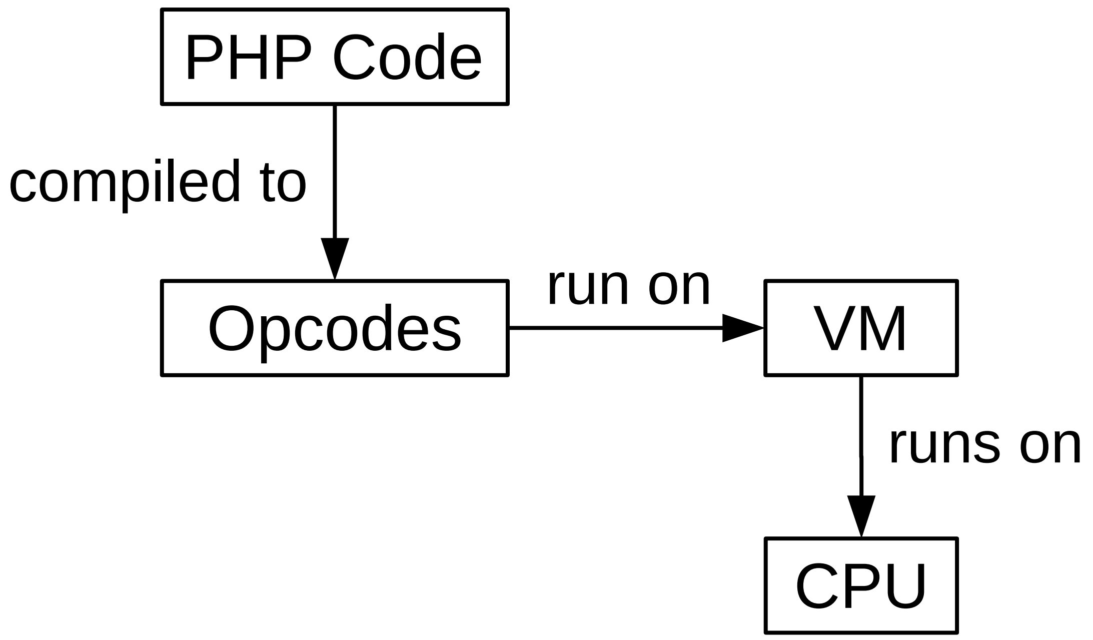
PHP 7.4

- Typed properties
- Arrow functions
- Restricting return types in child classes (covariance)
- Foreign Function Interface (FFI)
- Preloading
- WeakReference
- ??= operator
- … and more

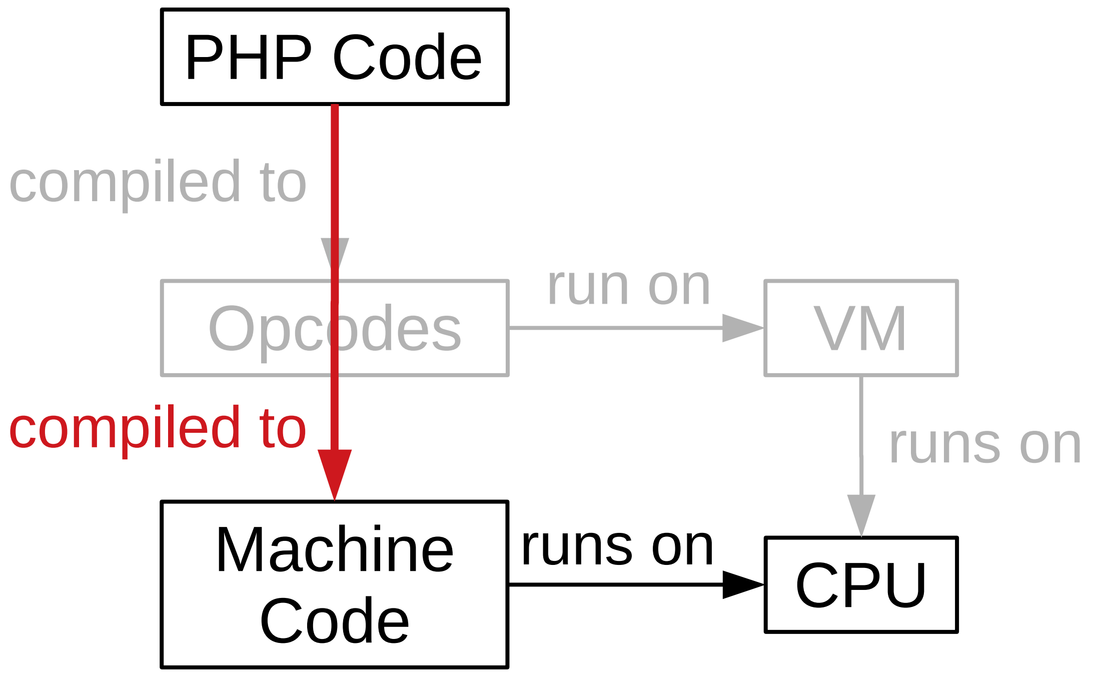
PHP 8.0

- Just-In-Time compiler (JIT)
- Removals + Cleanups
- ???

Just-In-Time (JIT) Compiler



Just-In-Time (JIT) Compiler



JIT Performance

- bench.php: 2.3x faster
- PHP-Parser: 30% faster
- WordPress: 3.5% faster

- JIT mainly benefits math heavy code
- Little impact on typical web code (currently)

Opcodes

```
function square_sum(int $n) {
    $sum = 0;
    for ($i = 0; $i < $n; $i++) {
        $sum += $i*$i;
    }
    return $sum;
}
```

Opcodes

```
function square_sum(int $n) {
    $sum = 0;
    for ($i = 0; $i < $n; $i++) {
        $sum += $i*$i;
    }
    return $sum;
}
```

Compilation to opcodes and optimization

```
    $n = RECV 1
    $sum = QM_ASSIGN int(0)
    $i = QM_ASSIGN int(0)
    JMP cond

loop:
    T0 = MUL $i, $i
    $sum = ADD $sum, T0
    PRE_INC $i

cond:
    T1 = IS_SMALLER $i, $n
    JMPNZ T1, loop
    RETURN $sum
```

Opcodes

```
function square_sum(int $n) {
    $sum = 0;
    for ($i = 0; $i < $n; $i++) {
        $sum += $i*$i;
    }
    return $sum;
}
```

Types:

```
$n   : int
$i   : int
T0   : int|float ($i*$i may overflow)
$sum : int|float
```

```
    $n = RECV 1
    $sum = QM_ASSIGN int(0)
    $i = QM_ASSIGN int(0)
    JMP cond

loop:
    T0 = MUL $i, $i
    $sum = ADD $sum, T0
    PRE_INC $i

cond:
    T1 = IS_SMALLER $i, $n
    JMPNZ T1, loop
    RETURN $sum
```

JIT Assembly

```
    ...
    xor %rdx, %rdx
    jmp .L6
.L3:
    ...
    mov %rdx, %rax
    imul %rax, %rax
    jo .L9
    mov %rax, 0x80(%r14)
    mov $0x4, 0x88(%r14)
.L4:
    cmp $0x4, 0x68(%r14)
    jnz .L12
    cmp $0x4, 0x88(%r14)
    jnz .L10
    mov 0x60(%r14), %rax
    add 0x80(%r14), %rax
    jo .L11
    mov %rax, 0x60(%r14)
.L5:
    add $0x1, %rdx
    ...
    ...
    xor %rdx, %rdx              ; $i = 0
    jmp .L6
.L3:
    ...
    mov %rdx, %rax
    imul %rax, %rax             ; %rax = $i*$i
    jo .L9
    mov %rax, 0x80(%r14)
    mov $0x4, 0x88(%r14)
.L4:
    cmp $0x4, 0x68(%r14)
    jnz .L12
    cmp $0x4, 0x88(%r14)
    jnz .L10
    mov 0x60(%r14), %rax
    add 0x80(%r14), %rax
    jo .L11
    mov %rax, 0x60(%r14)
.L5:
    add $0x1, %rdx              ; $i++
    ...
```

```
        ...
        xor %rdx, %rdx            ; $i = 0
        jmp .L6
    .L3:
        ...
        mov %rdx, %rax
        imul %rax, %rax           ; %rax = $i*$i
        jo .L9                    ; jump on overflow
        mov %rax, 0x80(%r14)      ; T0.value = %rax
        mov $0x4, 0x88(%r14)      ; T0.type  = int
    .L4:
        cmp $0x4, 0x68(%r14)
        jnz .L12
        cmp $0x4, 0x88(%r14)
        jnz .L10
        mov 0x60(%r14), %rax
        add 0x80(%r14), %rax
        jo .L11
        mov %rax, 0x60(%r14)
    .L5:
        add $0x1, %rdx            ; $i++
        ...
        ...
        xor %rdx, %rdx             ; $i = 0
        jmp .L6
    .L3:
        ...
        mov %rdx, %rax
        imul %rax, %rax            ; %rax = $i*$i
        jo .L9                     ; jump on overflow
        mov %rax, 0x80(%r14)       ; T0.value = %rax
        mov $0x4, 0x88(%r14)       ; T0.type  = int
    .L4:
        cmp $0x4, 0x68(%r14)       ; check if $sum.type == int
        jnz .L12
        cmp $0x4, 0x88(%r14)       ; check if T0.type == int
        jnz .L10
        mov 0x60(%r14), %rax
        add 0x80(%r14), %rax
        jo .L11
        mov %rax, 0x60(%r14)
    .L5:
        add $0x1, %rdx             ; $i++
        ...
        ...
        xor %rdx, %rdx              ; $i = 0
        jmp .L6
    .L3:

        ...
        mov %rdx, %rax
        imul %rax, %rax             ; %rax = $i*$i
        jo .L9                      ; jump on overflow
        mov %rax, 0x80(%r14)        ; T0.value = %rax
        mov $0x4, 0x88(%r14)        ; T0.type  = int
    .L4:
        cmp $0x4, 0x68(%r14)        ; check if $sum.type == int
        jnz .L12
        cmp $0x4, 0x88(%r14)        ; check if T0.type == int
        jnz .L10
        mov 0x60(%r14), %rax        ; load $sum.value
        add 0x80(%r14), %rax        ; %rax = $sum.value + T0.value
        jo .L11                     ; jump on overflow
        mov %rax, 0x60(%r14)        ; $sum.value = %rax
    .L5:
        add $0x1, %rdx              ; $i++
        ...
```

Improvement: Type Guards

- If the multiplication or addition overflows, fall back to the virtual machine
- → No unnecessary type checks
- → Everything kept in registers
- More generally: Runtime type profiling.

Removals

Anything deprecated in PHP <= 7.4 is no longer supported!

Full list of backwards incompatible changes:

`https://github.com/php/php-src/blob/master/UPGRADING`

PHP 4 Constructors

Now a normal method,
no longer a constructor

```
class Test {
    function Test() {
        /* … */
    }
}

class Test {
    function __construct() {
        /* … */
    }
}
```

Non-static method called statically

```
class Test {
    function method() {
        var_dump(isset($this)); // bool(false)
    }
}

Test::method();
// Deprecated: Non-static method Test::method()
// should not be called statically
```

Non-static method called statically

Guaranteed to exist

```
class Test {
    function method() {
        var_dump(isset($this));
    }
}

Test::method();
// Error: Non-static method Test::method()
// cannot not be called statically
```

Bareword fallback

```
var_dump(PHP_NIT_MAX); // string(11) "PHP_NIT_MAX"
// Warning: Use of undefined constant PHP_NIT_MAX -
// assumed 'PHP_NIT_MAX'
```

Bareword fallback

```
var_dump(PHP_NIT_MAX);
// Error: Undefined constant 'PHP_NIT_MAX'
```

Short Open Tags

- <? deprecated in 7.4, removed in 8.0
- Only <?php and <?= supported
- (???) short_open_tag default value from On to Off in 7.4

Disclaimer: RFC accepted, but much push-back after voting.

Many more...

- (unset)
- $php_errormsg
- Case-insensitive constants
- __autoload()
- assert() string args
- create_function()
- each()
- mbstring.func_overload
- ...

Concatenation Precedence

```php
$a = 1;
$b = 2;
echo "Sum: " . $a+$b;
// currently interpreted as
echo ("Sum: " . $a)+$b; // Prints "2"
```

Concatenation Precedence

```
$a = 1;
$b = 2;
echo "Sum: " . $a+$b;
// currently interpreted as
echo ("Sum: " . $a)+$b; // Prints "2"
// will become
echo "Sum: " . ($a+$b); // Prints "Sum: 3"
```

Disclaimer: Voting in progress, likely to pass.

Ternary Associativity

```
return $a == 1 ? 'one'
     : $a == 2 ? 'two'
               : 'other';
// was intended as:
return $a == 1  ? 'one'
     : ($a == 2 ? 'two'
                : 'other');
// but PHP interprets it as:
return ($a == 1 ? 'one'
     : $a == 2) ? 'two'
                : 'other';
```

Ternary Associativity

```php
return $a == 1 ? 'one'     // Deprecated in 7.4.
     : $a == 2 ? 'two'     // Compile error in 8.0.
               : 'other';
// was intended as:
return $a == 1  ? 'one'
     : ($a == 2 ? 'two'
                : 'other');
// but PHP interprets it as:
return ($a == 1 ? 'one'
     : $a == 2) ? 'two'
                : 'other';
```

Disclaimer: Voting in progress, likely to pass.

TypeErrors for internal functions

```php
function foo(int $bar) {}
foo("not an int");
// TypeError: Argument 1 passed to foo()
// must be of the type int, string given
```

TypeErrors for internal functions

```
function foo(int $bar) {}
foo("not an int");
// TypeError: Argument 1 passed to foo()
// must be of the type int, string given


var_dump(strlen(new stdClass)); // NULL
// Warning: strlen() expects parameter 1
// to be string, object given
```

TypeErrors for internal functions

```
function foo(int $bar) {}
foo("not an int");
// TypeError: Argument 1 passed to foo()
// must be of the type int, string given


var_dump(strlen(new stdClass)); // NULL
// Warning: strlen() expects parameter 1
// to be string, object given
```

Return type becomes `?int`

TypeErrors for internal functions

```
function foo(int $bar) {}
foo("not an int");
// TypeError: Argument 1 passed to foo()
// must be of the type int, string given


var_dump(strlen(new stdClass));
// TypeError: strlen() expects parameter 1
// to be string, object given
```

Number to string comparison

```
var_dump(0 == "foo"); // bool(true)
```

Number to string comparison

```
var_dump(0 == "foo"); // bool(true)

0 == "foo"
// evaluated as
0 == (int)"foo"
```

Number to string comparison

```
var_dump(0 == "foo"); // bool(true)

0 == "foo"
// evaluated as
0 == (int)"foo"
// but would be better as
(string)0 == "foo"
```

Disclaimer: Draft proposal, may not happen.

Number to string comparison

```
Comparison     | Before | After
------------------------------
 0 == "0"      | true   | true
 0 == "0.0"    | true   | true
 0 == "foo"    | true   | false
 0 == ""       | true   | false
42 == "   42"  | true   | true
42 == "42foo"  | true   | false
```

Disclaimer: Draft proposal, may not happen.

String to string comparison

```
Comparison                      | Result
--------------------------------------
"42" == "000042"                | true
"42" == "42.0"                  | true
"42.0" == "+42.0E0"             | true
"0" == "0e214987142012"         | true
```

Also weird :(

Features?

- Pure speculation ahead
- Things I would like to have and might work on personally

Property Accessors

```
class User {
    public int $age;
}
```

Property Accessors

```
class User {
    public int $age {
        get;
        set($age) {
            if ($age <= 0)
                throw new Exception("Must be positive");
            $this->age = $age;
        }
    }
}
```

Related: Read-only properties, properties in interfaces

Union Types

```
function mul(int|float $n1, int|float $n2) : int|float
{
    return $n1 * $n2;
}

function lookup(array $ary, int|string $key)
{
   return $ary[$key];
}
```

We’ll likely get this…

Generics

```
class Collection<K, V> implements ArrayAccess<K, V> {

    public function offsetGet(K $k): V {
        Return $this->data[$k];
    }

    public function offsetSet(K $k, V $v): void {
        $this->data[$k] = $v;
    }

    /* … */
}
```

This is going to take a lot of work...

Directory-scoped declares

```
directory_declare(__DIR__ . "/src", [
    "strict_types" => true,
]);
```

Seems to be quite controversial...

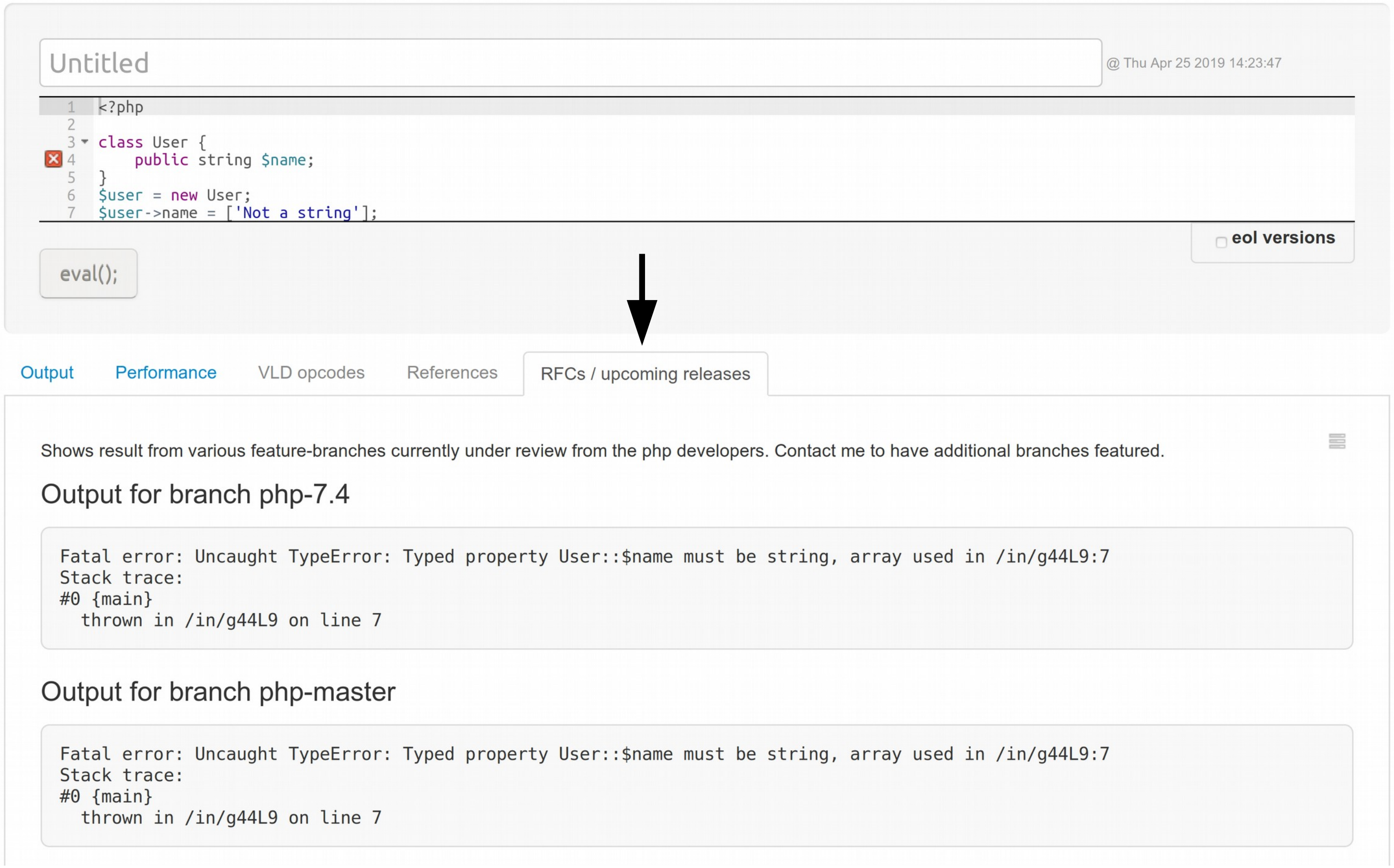
Directory-scoped declares

```
directory_declare(__DIR__ . "/src", [
    "strict_types" => true,
    "no_dynamic_properties" => true,
]);
```

Seems to be quite controversial...

What else?

3v4l.org



Untitled

@ Thu Apr 25 2019 14:23:47

```php
<?php

class User {
    public string $name;
}
$user = new User;
$user->name = ['Not a string'];
```

Shows result from various feature-branches currently under review from the php developers. Contact me to have additional branches featured.

Output for branch php-7.4

```
Fatal error: Uncaught TypeError: Typed property User::$name must be string, array used in /in/g44L9:7
Stack trace:
#0 {main}
  thrown in /in/g44L9 on line 7
```

Output for branch php-master

```
Fatal error: Uncaught TypeError: Typed property User::$name must be string, array used in /in/g44L9:7
Stack trace:
#0 {main}
  thrown in /in/g44L9 on line 7
```

Travis CI

```
php:
  - 7.3
  - 7.4snapshot
  - nightly  ← PHP 8

install:
  - composer install --ignore-platform-reqs
```

PHPUnit claims it is not PHP 8 compatible (it is)

Docker

- https://github.com/devilbox/docker-php-fpm-7.4
- https://github.com/devilbox/docker-php-fpm-8.0

Thank You!

Questions?