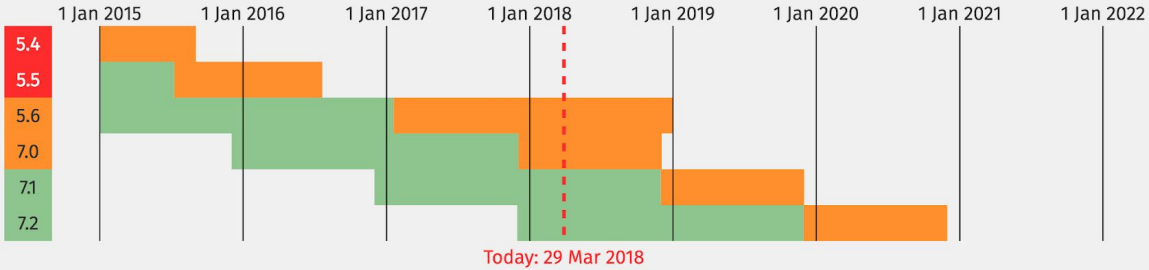
PHP 7.0 / 7.1 / 7.2 All New Features By Code

Daniel Schröder
skriptfabrik GmbH
@schroedan

Currently Supported Versions

Branch	Initial Release		Active Support Until		Security Support Until	
5.6 *	28 Aug 2014	*3 years, 7 months ago*	19 Jan 2017	*1 year, 2 months ago*	31 Dec 2018	*in 9 months*
7.0	3 Dec 2015	*2 years, 3 months ago*	3 Dec 2017	*3 months ago*	3 Dec 2018	*in 8 months*
7.1	1 Dec 2016	*1 year, 3 months ago*	1 Dec 2018	*in 8 months*	1 Dec 2019	*in 1 year, 8 months*
7.2	30 Nov 2017	*3 months ago*	30 Nov 2019	*in 1 year, 8 months*	30 Nov 2020	*in 2 years, 8 months*

Or, visualised as a calendar:



Key

Active support	A release that is being actively supported. Reported bugs and security issues are fixed and regular point releases are made.
Security fixes only	A release that is supported for critical security issues only. Releases are only made on an as-needed basis.
End of life	A release that is no longer supported. Users of this release should upgrade as soon as possible, as they may be exposed to unpatched security vulnerabilities.

Source: http://php.net/supported-versions.php (March 2018)

Declarations

Scalar type declarations, nullable types, iterable pseudo-type

```php
function printScalarTypes(string $string, int $int, float $float, bool $bool)
{
   print implode(' ', [gettype($string), gettype($int), gettype($float), gettype($bool)]);
}

function pintSumOfInts(int ...$ints)
{
   print array_sum($ints);
}

function examineString(?string $string)
{
   print 'is_' . strtolower(gettype($string));
}

// iterable = array || implement Traversable interface
function joinIterable(iterable $array = ['work', 'work', 'work'])
{
   print implode(' ', (array)$array);
}

```

7.0

7.1

7.2

Return type declarations, object type, void functions

```php
function sumOfInts(int ...$ints): int
{
    return array_sum($ints);
}

function maybeString(): ?string
{
    return array_rand(['ele', 'PHP', 'ant', null]);
}

function goToSleep(int $seconds): void
{
    sleep($seconds);
}

function createObject(): object
{
    return new stdClass();
}

```

7.0

7.1

7.2

Grouped use declarations

```php
// pre PHP 7 code
use Some\Namespace\ClassA;
use Some\Namespace\ClassB;
use Some\Namespace\ClassC as C;

use function Some\Namespace\fn_a;
use function Some\Namespace\fn_b;
use function Some\Namespace\fn_c;

use const Some\Namespace\ConstA;
use const Some\Namespace\ConstB;
use const Some\Namespace\ConstC;

```

```php
// PHP 7+ code
use Some\Namespace\{
    ClassA,
    ClassB,
    ClassC as C
};

use function Some\Namespace\{
    fn_a,
    fn_b,
    fn_c
};

use const Some\Namespace\{
    ConstA,
    ConstB,
    ConstC,
};

```

7.0

7.2

Classes

Anonymous classes

7.0

```php
interface Logger
{
    public function log(string $msg);
}

class Application
{
    private $logger;

    public function getLogger(): Logger
    {
        return $this->logger;
    }

    public function setLogger(Logger $logger)
    {
        $this->logger = $logger;
    }
}

$app = new Application();
$app->setLogger(new class implements Logger
{
    public function log(string $msg)
    {
        echo $msg;
    }
});

print get_class($app->getLogger());
// "class@anonymous"
```

Abstract method overriding, parameter type widening

7.2

```php
abstract class A
{
    abstract function test(string $s);
}

abstract class B extends A
{
    // overridden - still maintaining
    // contravariance for parameters and
    // covariance for return
    abstract function test($s) : int;
}

interface X
{
    public function test(array $input);
}

class Y implements X
{
    // type omitted for $input
    public function test($input){}
}
```

Class constant visibility, multi catch exceptions, IntlChar class

```php
class ConstDemo
{
    const PUBLIC_CONST_A = 1;
    public const PUBLIC_CONST_B = 2;
    protected const PROTECTED_CONST = 3;
    private const PRIVATE_CONST = 4;
}

try {
    // some code
} catch (FirstException | SecondException $e) {
    // handle first and second exceptions
}

```

7.1

```php
var_dump(IntlChar::charName('@'));
// string(13) "COMMERCIAL AT"

var_dump(IntlChar::ispunct('!'));
// bool(true)

```

7.0

Operations and operators

Integer division, Null coalescing and spaceship operator

7.0

```php
var_dump(intdiv(10, 3)); // int(3)

// fetches the value of $_GET['user'] and
// returns 'nobody' if it does not exist.
$username = $_GET['user'] ?? 'nobody';

// this is equivalent to:
$username = isset($_GET['user'])
    ? $_GET['user']
    : 'nobody';

```

```php
// integers
var_dump(1 <=> 1); // int(0)
var_dump(1 <=> 2); // int(-1)
var_dump(2 <=> 1); // int(1)

// floats
var_dump(1.5 <=> 1.5); // int(0)
var_dump(1.5 <=> 2.5); // int(-1)
var_dump(2.5 <=> 1.5); // int(1)

// strings
var_dump('a' <=> 'a'); // int(0)
var_dump('a' <=> 'b'); // int(-1)
var_dump('b' <=> 'a'); // int(1)

```

Symmetric array destructuring

7.1

```php
$data = [
    [1, 'Tom'],
    [2, 'Fred'],
];

// list() style
list($id1, $name1) = $data[0];

// [] style
[$id1, $name1] = $data[0];

// list() style
foreach ($data as list($id, $name)) {
    // logic here with $id and $name
}

// [] style
foreach ($data as [$id, $name]) {
    // logic here with $id and $name
}

```

… with support for keys in list

7.1

```php
$data = [
    ['id' => 1, 'name' => 'Tom'],
    ['id' => 2, 'name' => 'Fred'],
];

// list() style
list('id' => $id1, 'name' => $name1) = $data[0];

// [] style
['id' => $id1, 'name' => $name1] = $data[0];

// list() style
foreach ($data as list('id' => $id, 'name' => $name)) {
    // logic here with $id and $name
}

// [] style
foreach ($data as ['id' => $id, 'name' => $name]) {
    // logic here with $id and $name
}

```

Constant arrays, filtered unserialize, negative string offsets

```php
define('ANIMALS', [
    'dog',
    'cat',
    'bird'
]);

var_dump(ANIMALS[1]); // string(3) "cat"

// converts all objects into __PHP_Incomplete_Class object
$data = unserialize($foo, ['allowed_classes' => false]);

// converts all objects into __PHP_Incomplete_Class object except those of MyClass and MyClass2
$data = unserialize($foo, ['allowed_classes' => ['MyClass', 'MyClass2']]);

// default behaviour (same as omitting the second argument) that accepts all classes
$data = unserialize($foo, ['allowed_classes' => true]);

var_dump('abcdef'[-2]); // string(1) "e"

var_dump(strpos('aabbcc', 'b', -3)); // int(3)

```

7.0

7.1

Generators, closures, functions, ...

Generator return expressions, generator delegation

7.0

```php
$gen = (function () {
    yield 1;
    yield 2;

    return 3;
})();

foreach ($gen as $val) {
    var_dump($val);
}
// int(1)
// int(2)
// int(3)

var_dump($gen->getReturn()); // int(3)
print get_class($gen); // "Generator"

```

```php
function gen()
{
    yield 1;
    yield 2;
    yield from gen2();
}

function gen2()
{
    yield 3;
    yield 4;
}

foreach (gen() as $val) {
    var_dump($val);
}
// int(1)
// int(2)
// int(3)
// int(3)

```

Closure::call(), Closure::fromCallable()

```php
class A
{
    private $x = 1;
}

$getX = function () {
    return $this->x;
};

// pre PHP 7 code
$getXCB = $getX->bindTo(new A, 'A');
var_dump($getXCB()); // int(1)

// PHP 7+ code
var_dump($getX->call(new A)); // int(1)
```

7.0

```php
class A
{
    public function exposeFunction()
    {
        return Closure::fromCallable([
            $this,
            'privateFunction'
        ]);
    }

    private function privateFunction($param)
    {
        var_dump($param);
    }
}

$privFunc = (new A)->exposeFunction();
$privFunc('value'); // string(5) "value"
```

7.1

preg_replace_callback_array(), session options

7.0

```php
$subject = 'Aaaaaa Bbb';

preg_replace_callback_array(
    [
        '~[a]+~i' => function ($match) {
            print strlen($match[0]), ' matches for "a" found', PHP_EOL;
        },
        '~[b]+~i' => function ($match) {
            print strlen($match[0]), ' matches for "b" found', PHP_EOL;
        }
    ],
    $subject
);

// override the session configuration directives
session_start([
    'cache_limiter' => 'private',
    // can only be passed to session_start()
    'read_and_close' => true,
]);

```

Expectations, unicode codepoint escape syntax

7.0

```php
ini_set('assert.exception', 1);

class CustomError extends AssertionError
{
}

assert(false, new CustomError('Some error message'));
// PHP Fatal error:  Uncaught CustomError: Some error message in /path/test.php:9

var_dump("\u{aa}"); // string(2) "ª"
var_dump("\u{0000aa}"); // string(2) "ª"
var_dump("\u{9999}"); // string(3) "香"

```

What else?

Some more considerable enhancements

- two new functions have been added to generate cryptographically secure integers and strings in a cross platform way: `random_bytes()` and `random_int()` — 7.0
- `list()` function can always unpack objects implementing `ArrayAccess` — 7.0
- new function called `pcntl_async_signals()` has been introduced to enable asynchronous signal handling — 7.1
- HTTP/2 server push support in ext/curl can be leveraged through the `curl_multi_setopt()` function — 7.1
- `proc_nice()` function is now supported on Windows — 7.2
- `pack()` and `unpack()` functions now support float and double in both little and big endian — 7.2
- EXIF extension has been updated to support a much larger range of formats when parsing images with the `exif_read_data()` function — 7.2
- Enhancements to the PCRE, SQLite3, Oracle OCI8 and ZIP extension — 7.2

What’s next?

PHP 7.3 (features to be continued...)

- Release planned in late 2018
- Improved PHP GC
- Added `net_get_interfaces()` function
- Allow a trailing comma in function calls (https://wiki.php.net/rfc/trailing-comma-function-calls)
- PCRE2 migration (https://wiki.php.net/rfc/pcre2-migration)
- Added support for references in `list()` and array destructuring (https://wiki.php.net/rfc/list_reference_assignment)
- Added `is_countable()` function (https://wiki.php.net/rfc/is-countable)
- JSON: Added `JSON_THROW_ON_ERROR` flag (https://wiki.php.net/rfc/json_throw_on_error)
- Date: Added `DateTime::createFromImmutable()` method to create `DateTime` from `DateTimeImmutable` object
- GD: Added support for WebP in `imagecreatefromstring()`

PHP 8.0 (just some rumors)

- Release planned for November 2021
- New JIT (Just In Time) engine
- Arrays starting with a negative index (https://wiki.php.net/rfc/negative_array_index)
- Merge class member symbol tables (https://wiki.php.net/rfc/php8/merge_member_symbol_tables)
- Merge constant, function and class symbol tables https://wiki.php.net/rfc/php8/merge_symbol_tables)
- Unify behavior of userland and internal functions: In particular when internal functions fail to parse argument types correctly they fail by returning null, userland functions throw a TypeError
- Probably reserve new types related to variance such as `mixed`
- Extend instanceof to work with non-classes and add new functions for checking type and subtype relationships
- Unify class and abstract type error behaviors when method incompatibilities exists


NOW, ARE THERE
ANY QUESTIONS?

Thank you!

References

- http://php.net/supported-versions.php
- http://php.net/manual/en/appendices.php
- http://php.net/manual/en/migration70.new-features.php
- http://php.net/manual/en/migration71.new-features.php
- http://php.net/manual/en/migration72.new-features.php
- https://wiki.php.net/rfc#php_73
- https://github.com/php/php-src/blob/master/NEWS
- https://wiki.php.net/rfc/php8
- https://react-etc.net/entry/php-8-0-0-release-date-and-jit-status