What's new in *PHP 7.3*

PHP User Group Rheinhessen #55 (16.10.2018)

Matthias Gutjahr (@mattsches)

PHP 7.3 Timeline

Alpha 1 am 7. Juni 2018, Beta 1 am 2. August 2018, RC 1 am 13. September 2018

Geplantes Release-Datum: *13. Dezember 2018*

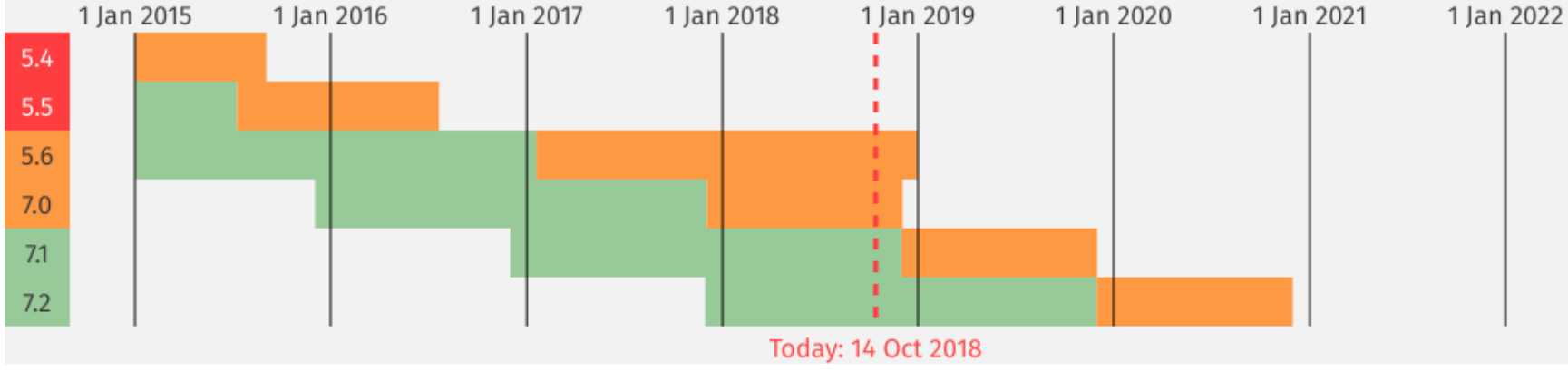
Release-Manager: Christoph M. Becker, Stanislav Malyshev

Aktuelle Versionen (Stand: 14.10.2018)

Currently Supported Versions

Branch	Initial Release		Active Support Until		Security Support Until	
5.6 *	28 Aug 2014	*4 years, 1 month ago*	19 Jan 2017	*1 year, 8 months ago*	31 Dec 2018	*in 2 months*
7.0	3 Dec 2015	*2 years, 10 months ago*	3 Dec 2017	*10 months ago*	3 Dec 2018	*in 1 month*
7.1	1 Dec 2016	*1 year, 10 months ago*	1 Dec 2018	*in 1 month*	1 Dec 2019	*in 1 year, 1 month*
7.2	30 Nov 2017	*10 months ago*	30 Nov 2019	*in 1 year, 1 month*	30 Nov 2020	*in 2 years, 1 month*

Or, visualised as a calendar:



Running/testing PHP 7.3 in Docker

```
$ docker run -it --rm -v "$PWD":/app -w /app \
php:7.3.0RC3-cli php index.php
```

Heredoc/Nowdoc Closing Marker Indentation

```php
<?php
class HeredocExample {
    public $content = <<<EOT
      Ich bin
     ein wichtiger
    Text
    EOT;
}
$ex = new HeredocExample();
echo $ex->content;

  Ich bin
 ein wichtiger
Text
```

Heredoc/Nowdoc Closing Marker New Line

```
<?php
$values = [<<<END
a
b
c
END, 'd e f'];
```

Allow a trailing comma in function calls

Yes 30 : 10 No

```php
<?php
$bar = function(...$args) {
  //
};
$bar(
  'closure',
  'bar',
);
// oder:
$foo = new Foo(
  'constructor',
  'bar',
);
// oder:
var_dump(
    $whatIsInThere,
    $probablyABugInThisOne,
    $oneMoreToCheck,
);
```

JSON_THROW_ON_ERROR

> This RFC instead proposes adding a new option flag value for json_decode() and json_encode(), JSON_THROW_ON_ERROR. When passed this flag, the error behaviour of these functions is changed. The global error state is left untouched, and if an error occurs that would otherwise set it, these functions instead throw a JsonException with the message and code set to whatever json_last_error() and json_last_error_msg() would otherwise be respectively.

```
try {
    json_decode("{", false, 512, JSON_THROW_ON_ERROR);
} catch (\JsonException $exception) {
    echo $exception->getMessage();
}
```

list() Reference Assignment

```php
$array = [1, 2];
list($a, &$b) = $array;
```

macht das gleiche wie:

```php
$array = [1, 2];
$a = $array[0];
$b =& $array[1];
```

is_countable

`bool is_countable(mixed $var)`

Beispiel:

```php
// bisher
if (is_array($foo) || $foo instanceof Countable) {
    // $foo is countable
}

// PHP 7.3
if (is_countable($foo)) {
    // $foo is countable
}
```

array_key_first(), array_key_last()

```php
$array = ['a' => 1, 'b' => 2, 'c' => 3];

$firstKey = array_key_first($array);
$lastKey = array_key_last($array);
assert($firstKey === 'a');
assert($lastKey === 'c');
```

Deprecations 1/2

- Undocumented mbstring function aliases: `mbereg_replace()` -> `mb_ereg_replace()`
- String search functions with integer needle

```
$str = "There are 10 apples";
var_dump(strpos($str, "10")); // int(10)
var_dump(strpos($str, 10));   // bool(false)
```

- fgetss() function and string.strip_tags filter:

> fgetss â€” Gets line from file pointer and strip HTML tags

- Defining a free-standing assert() function

Deprecations 2/2

- `FILTER_FLAG_SCHEME_REQUIRED` and `FILTER_FLAG_HOST_REQUIRED`
- pdo_odbc.db2_instance_name php.ini directive
- `each()` - use `foreach` instead (it is more than 10 times faster)
- `assert()` with string argument - it is using `eval()` internally!
- `$errcontext` argument of error handler - use `debug_backtrace()` instead

Ausblick auf PHP 7.4: Typed Properties

```php
class Example {
    // All types with the exception of "void" and "callabl
    public int $scalarType;
    protected ClassName $classType;
    private ?ClassName $nullableClassType;

    // Types are also legal on static properties
    public static iterable $staticProp;

    // Types can also be used with the "var" notation
    var bool $flag;

    // Typed properties may have default values
    public string $str = "foo";
    public ?string $nullableStr = null;

    // The type applies to all properties in one declarati
    public float $x, $y;
    // equivalent to:
    public float $x;
    public float $y;
}
```

Community-driven PHP 8 Wish List Top 10

1. Generics
2. Arrow functions aka short lambdas
3. Multi-threading
4. Scalar objects
5. Method overloading
6. Async programming (ansync/await, promises, event loops)
7. Static typing (typed everything)
8. Remove `<?php` and `declare_strict` statement
9. Boot-time preloading
10. Property accessors

Thanks!