PHP 7: What To Expect

Lorna Mitchell, PHPUK 2016

Slides available online, help yourself:
http://www.lornajane.net/resources

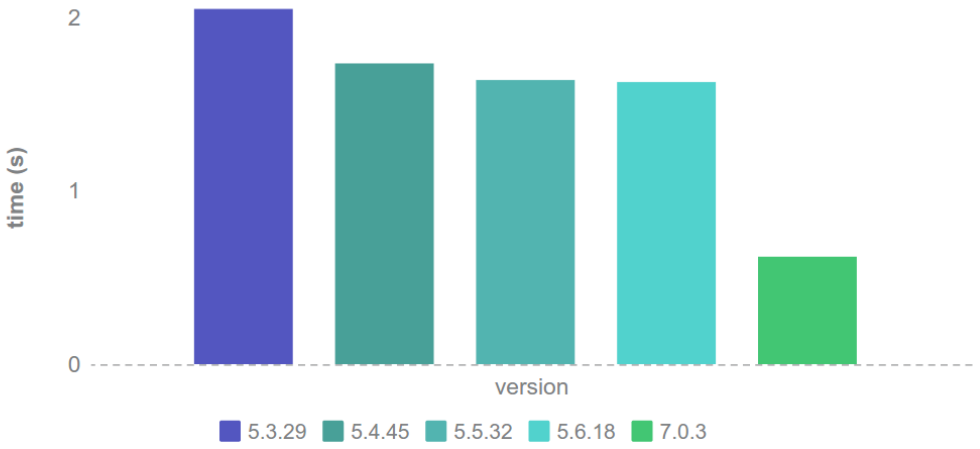
Versions of PHP

Version	Support until	Security fixes until
PHP 5.5	(expired)	10th July 2016
PHP 5.6	31st December 2016	31st December 2018
PHP 7.0	3rd December 2017	3rd December 2018

see also: http://php.net/supported-versions.php

PHP 7 Is Fast

PHP 7 Is Fast



Why PHP 7 Is Fast

- Grew from the phpng project
- Influenced by HHVM/Hacklang
- Major refactoring of the Zend Engine
- More compact data structures throughout
- As a result all extensions need updates
    - http://gophp7.org/gophp7-ext/

Rasmus' stats: http://talks.php.net/fluent15#/6

Abstract Syntax Trees

PHP 7 uses an additional AST step during compilation

This gives a performance boost and much nicer architecture

Abstract Syntax Trees

Example code:

```
1 $a = rand(0,1);
2
3 if($a) {
4     echo "Heads";
5 } else {
6     echo "Tails";
7 }
```

Abstract Syntax Trees

Tokenized PHP:

```
T_OPEN_TAG: <?php

T_VARIABLE: $a
T_WHITESPACE:
=
T_WHITESPACE:
T_STRING: rand
(
T_LNUMBER: 0
,
T_LNUMBER: 1
)
;
```

Abstract Syntax Trees

Abstract syntax tree representation:

```
AST_STMT_LIST
  AST_ASSIGN
    AST_VAR
      a
    AST_CALL
      AST_NAME
        rand
      AST_ARG_LIST
        0
        1
```

New Features

Combined Comparison Operator

The <=> "spaceship" operator is for quick greater/less than comparison.

```
1 echo 2 <=> 1; // 1
2 echo 2 <=> 3; // -1
3 echo 2 <=> 2; // 0
```

Ternary Shorthand

Refresher on this PHP 5 feature:

```
1 echo $count ? $count : 10; // 10
2 echo $count ?: 10; // 10
```

Null Coalesce Operator

Operator `??` is ternary shorthand (`?:`) but with `isset()`.

```php
$b = 16;

echo $a ?? 2; // 2
echo $a ?? $b ?? 7; // 16
```

Type Hints

PHP 5 has type hinting, allowing you to say what kind of parameter is acceptable in a method call.

```
1 function sample(array $list, $length) {
2     return array_slice($list, 0, $length);
3 }
```

Type Hints

If we use the wrong parameter types, it errors

```
1 print_r(sample(3, 3));
```

PHP 5 error:

```
Catchable fatal error: Argument 1 passed to sample() must be of the type array,
integer given
```

PHP 7 error:

```
Fatal error: Uncaught TypeError: Argument 1 passed to sample() must be of the type
array, integer given
```

Scalar Type Hints

PHP 7 lets us hint more datatypes:

- string
- int
- float
- bool

Scalar Type Hints

We can amend our code accordingly:

```
1 function sample(array $list, int $length) {
2     return array_slice($list, 0, $length);
3 }
```

And then call the method:

```
1 $moves = ['hop', 'skip', 'jump', 'tumble'];
2 print_r(sample($moves, "2")); // ['hop', 'skip']
```

Scalar Type Hints

To enable strict type check, add this line in the calling context:

```
declare(strict_types=1);
```

Return Type Hints

We can also type hint for return values.

```
1 function sample(array $list, int $length): array {
2     if($length > 0) {
3         return array_slice($list, 0, $length);
4     }
5     return false;
6 }
```

Beware that we can't return false or null.

Return Type Hints

This works:

```
1 $moves = ['hop', 'skip', 'jump', 'tumble'];
2 print_r(sample($moves, "2")); // ['hop', 'skip']
```

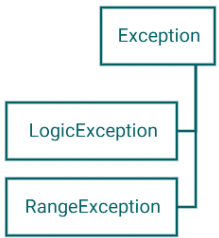
This errors:

```
1 $moves = ['hop', 'skip', 'jump', 'tumble'];
2 print_r(sample($moves, 0));
```

```
Fatal error: Uncaught TypeError: Return value of sample() must be of the type array,
boolean returned
```

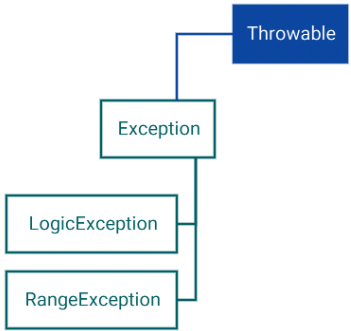
Exceptions and Errors

PHP 5 exceptions are alive, well, and excellent



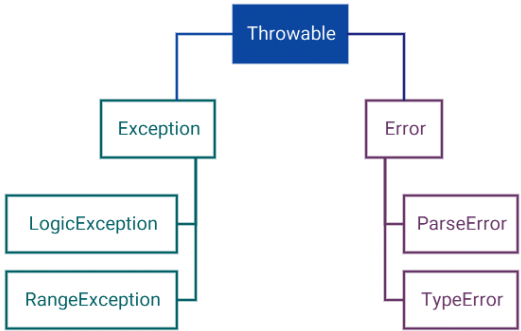
Exceptions in PHP 7

They now implement the `Throwable` interface



Errors in PHP 7

Some errors are now catchable via the `Error` class



Catching Exceptions and Errors

```
 1 function sample(array $list, int $length) {
 2     throw new Exception("You fail");
 3 }
 4
 5 try {
 6     $a = sample(1,1);
 7 } catch (Exception $e) {
 8     echo "you hit the exception line";
 9 } catch (TypeError $e) {
10     echo "you passed the wrong arguments"; }
```

Catch Method Calls on Non-Objects

Does this error look familiar?

```
1 $a = 6;
2 $a->grow();
```

PHP 5:

```
Fatal error: Call to a member function grow() on integer
```

PHP 7:

```
Fatal error: Uncaught Error: Call to a member function grow() on integer
```

Catch Method Calls on Non-Objects

PHP 7 allows us to catch Errors as well as Exceptions

```
1 try {
2     $a = 6;
3     $a->grow();
4 } catch (Error $e) {
5     echo "(oops! " . $e->getMessage() . ")\n";
6     // now take other evasive action
7 }
```

Newer bits of PHP will use this new Error mechanism

Anonymous Classes

Start with this (normal) class:

```
1 class Logger {
2     public function log($message) {
3         echo $message . "\n";
4     }
5 }
6
7 $log1 = new Logger();
```

Anonymous Classes

Now consider this anonymous class:

```
1 $log2 = new class extends Logger {
2     public function log($message) {
3         echo date('[d-M-Y] ')
4             . $message . "\n";
5     }
6 }
```

Anonymous Classes

Compare the two in use:

```
1 $log1->log("one line");
2 $log1->log("another line");
3 $log2->log("one line");
4 $log2->log("another line");
```

```
one line
another line
[18-Feb-2016] one line
[18-Feb-2016] another line
```

Random* Functions

PHP 7 introduces a couple of neat randomness functions:

- `random_bytes()` — Generates cryptographically secure pseudo-random bytes
- `random_int()` - Generates cryptographically secure pseudo-random integers

For PHP <7 use https://github.com/paragonie/random_compat

New JSON Extension

PHP 7 includes the JSOND extension.

No major changes but:

- has a friendly PHP-compatible license
- performs better than the alternatives

Upgrading to PHP 7

Uniform Variable Syntax

This is a feature as well as a gotcha.

- Good news: more consistent and complete variable syntax with fast parsing
- Bad news: some quite subtle changes from old syntax when dereferencing or using `$$`
- If in doubt, add more `{` and `}`

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

Phan

Static analyser: https://github.com/etsy/phan

- reads code and PHPDoc comments
- warns about BC breaks including uniform variable syntax issues
- warns you about undeclared things
- checks parameter types

Has a great guide to codebase wrangling:
http://lrnja.net/1W2Gjmb

Foreach

Check that you're not relying on any `foreach()` weirdnesses

- The array pointer will no longer move, look out for use of `current()` and `next()` inside a `foreach()` loop
- Don't assign to the thing you're looping over, the behaviour has changed

RFC: https://wiki.php.net/rfc/php7_foreach

Hex Numbers in Strings

PHP 7 doesn't detect hex numbers when casting strings to numeric values.

Deprecated Features

You should expect things that trigger `E_DEPRECATED` in older versions of PHP to be removed.

Caveats:

- The RFC to remove things was agreed but it hasn't been implemented yet
- The `mysql_*` functions really are removed
- PHP 4 constructors are less removed than you'd expect them to be

Upgrading to PHP 7

Step 1: Upgrade to PHP 5.5 or 5.6.

Upgrading to PHP 7

Step 1: Upgrade to PHP 5.5 or 5.6.

Most PHP 5 code will just work with a few pitfalls to look out for.

You probably want to run `composer update` while you're at it

Upgrading to PHP 7

There are fabulous comprehensive instructions
http://php.net/manual/en/migration70.php

Making the business case for PHP 7

- calculate hardware cost saving
- calculate developer time required

Done :)

Acquiring PHP 7

Windows users: get a new binary

Linux users:

- wait for your distro to update
- use an alternative source (e.g. http://lrnja.net/1PIPw2M)
- compile it yourself

The Future

The Future

- PHP 5.6 support has been extended
  - Support until 31st December 2016
  - Security fixes until 31st December 2018
- PHP 7.0 is safe to run
- PHP 7.1 looks even better

(see also http://php.net/supported-versions.php)

Questions?

Feedback please! https://joind.in/talk/0a501

Slides are on http://lornajane.net
(related blog posts are there too)

Contact me

- lorna@lornajane.net
- @lornajane

Bonus Content

No E_STRICT

Replaced with either `E_DEPRECATED` or `E_NOTICE` or `E_WARNING`

Simplifies error stuff in PHP 7

Multiple Import Declarations

Syntactic sugar perhaps, but very readable code. Start with:

```
1 use Symfony\Component\Form\Form;
2 use Symfony\Component\Form\FormError;
3 use Talk\TalkDb;
4 use Talk\TalkApi;
5 use User\UserDb;
6 use User\UserApi;
7
```

Multiple Import Declarations

Syntactic sugar perhaps, but very readable code. Now reads:

```
1 use Symfony\Component\Form\{Form, FormError};
2 use Talk\{TalkDb, TalkApi};
3 use User\{UserDb, UserApi};
4
```

Group your imports, also supports aliases.