PHP 7: What To Expect

Lorna Mitchell, OSCON Europe

http://www.lornajane.net/resources
About PHP 7

PHP 7 is the next major release of PHP

- Follows PHP 5.6 (there is no PHP 6)
- Currently at RC5
- Due for final release in November 2015
- First major release of PHP since 2004
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/](http://gophp7.org/gophp7-ext/)

Rasmus' stats: [http://talks.php.net/fluent15#/6](http://talks.php.net/fluent15#/6)
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
```

Use it with numbers, strings and even arrays - but not objects.
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()`.

```
1 $b = 16;
2
3 echo $a ?? 2;  // 2
4 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.

```php
function sample(array $list, $length) {
    return array_slice($list, 0, $length);
}
```
Type Hints

If we use the wrong parameter types, it errors

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

PHP 5 error:

PHP 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:

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

And then call the method:

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

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

```
delare(strict_types=1);
```
Return Type Hints

We can also type hint for return values.

```php
function sample(array $list, int $length): array {
    if($length > 0) {
        return array_slice($list, 0, $length);
    }
    return false;
}
```

Beware that we can't return false or null.
This works:

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

This errors:

```php
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 inherit from Throwable.
Some errors are now catchable via the `Error` class.
Catching Exceptions and Errors

```php
function sample(array $list, int $length) {
    throw new Exception("You fail");
}

try {
    $a = sample(1,1);
} catch (Exception $e) {
    echo "you hit the exception line";
} catch (TypeError $e) {
    echo "you passed the wrong arguments";
}
```
Catch Method Calls on Non-Objects

Does this error look familiar?

```php
1 $a->grow();
```

PHP 5:

PHP Fatal error: Call to a member function grow() on null

PHP 7:

Fatal error: Uncaught Error: Call to a member function grow() on unknown
Catch Method Calls on Non-Objects

```php
try {
    $a->grow();
} catch (Error $e) {
    echo "(oops! " . $e->getMessage() . " )\n";
    // now take other evasive action
}
```
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
Upgrading to PHP 7
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.

I expect all modern applications to be upgradeable (and therefore upgraded!).
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
Static analyser: https://github.com/rlerdorf/phan
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](https://wiki.php.net/rfc/php7_foreach)
Deprecated Features

The majority of things that trigger E_DEPRECATION in older versions of PHP are now actually removed.

This includes the `mysqli_*` functions. PDO is great, I promise.
Upgrading to PHP 7

There are fabulous comprehensive instructions https://github.com/php/php-src/blob/PHP-7.0.0/UPGRADING

To make the business case:

• calculate hardware cost saving
• calculate developer time required

Done :)
PHP 7 and You
Use PHP 7

• Use Rasmus' dev box
  https://github.com/rlerdorf/php7dev
• Install using package manager (homebrew, PPAs)
• Use Zend's nightly builds for your platform
  http://php7.zend.com/
• Bitnami LAMP/MAMP stacks
  http://lrnja.net/1M5fEnH
• Compile new PHP yourself
  https://github.com/php/php-src/
How To Test

Get PHP 7 and then:

• Run your test suites (travis already has PHP 7 available)
• Then run your actual PHP 5 applications
• Narrow down good replication cases, report bugs to appropriate place

Tutorial for putting your project onto php7dev: http://lrnja.net/1MSlFkt
PHP 7 Is Coming

It's fast and ... fabulous
Questions?

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:

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

Group your imports, also supports aliases.
Hex Numbers in Strings

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

Start with this (normal) class:

```php
1 class Logger {
2     public function log($message) {
3         echo $message;
4     }
5 }
6
7 $log1 = new Logger();
```
Anonymous Classes

Now consider this anonymous class:

```php
$log2 = new class extends Logger {
    public function log($message) {
        echo $message . "\n";
    }
};
```
Anonymous Classes

Compare the two in use:

1 $log1->log("one line");
2 $log1->log("another line");
3 echo "----
";
4 $log2->log("one line");
5 $log2->log("another line");

one line another line ----
one line
another line