DRUPALCON VIENNA

PHP 7+

The why's and the how's

Ayesh Karunaratne

DRUPALCON VIENNA

Hallo!

Ayesh Karunaratne

- “Ayesh” on drupal.org and Github
- Drupal.org project review admin
- Drupal Sinhalese translation admin
- Drupal Sri Lanka group admin
- Drupal/Wordress module maintainer


DRUPALCON
VIENNA
WHY?

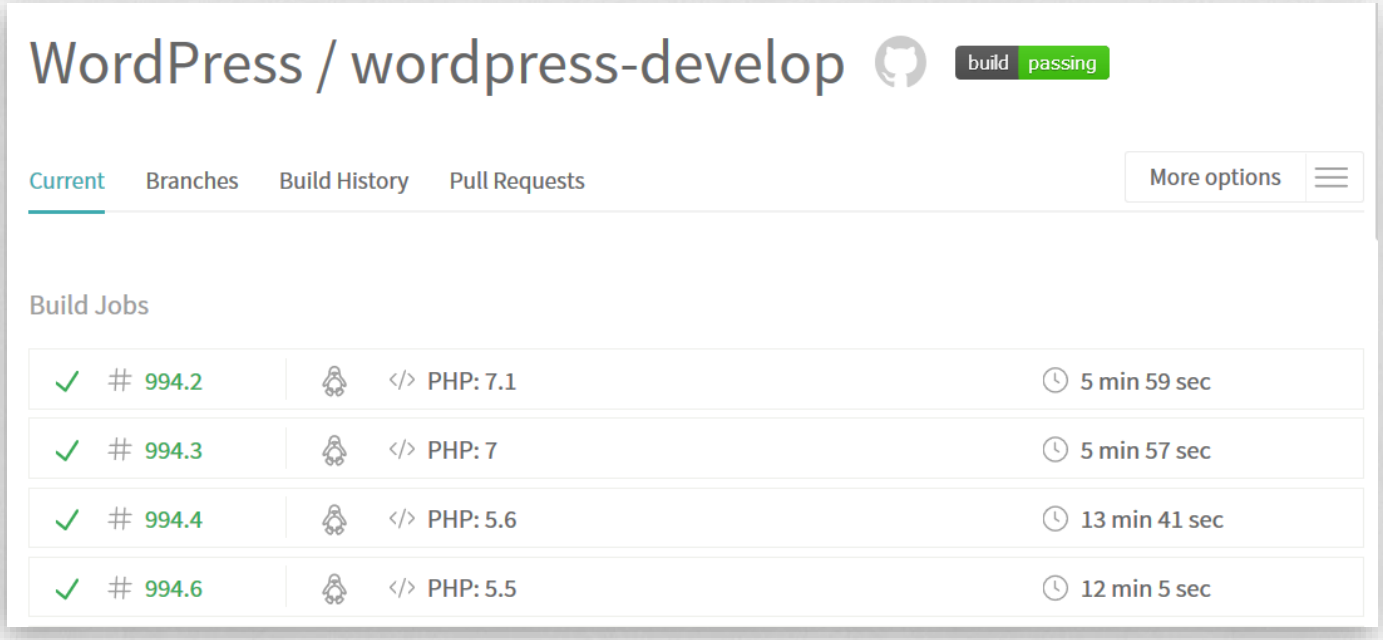
PHP 5.2: 2006
PHP 5.3: 2009
PHP 5.4: 2012
PHP 5.5: 2013

PHP 5.6: 2014

PHP 7.0: 2015
PHP 7.1: 2016
PHP 7.2: 2017

If it works, why bother?

PHP 7 is faster.



Many popular frameworks require PHP 7.

- Laravel 5.5
- Symfony 4 & components – Twig, console, Finder, etc
- Doctrine 2.8 – PHP 7.1
- PHPUnit 6
- Your packages too.

Require PHP 7 in your projects

Drupal modules

`module.info`

```
php=7.0
```

`module.info.yml`

```
php: 7.0
```



Composer packages

```
composer.json

"require": {
    "php": "^7.0"
}
```

Wordpress Plugins

readme.txt

```
=== Plugin Name ===
Tags: comments, spam
Requires at least: 4.8
Requires PHP: 5.6
```

Features before PHP 7.0

- Namespaces
- Closures
- Late static binding
- Traits
- Short array syntax
- `finally`
- Generators
- Built-in password hashing


wow
Such typing
Much OOP

PHP 7.0, 7.1, 7.2 in summary

- Parameter and return type improvements.
- Improved exceptions and exception handling.
- Security improvements
- Asynchronous signal handling
- SAPI and legacy extensions removed, new ones added.

Abstract Syntax Tree

- Intermediary structure in compilation process
- Decoupled parser and compiler
- Uniform variable syntax
- AST-based implementations are 10-20% faster
- Uses more memory (~10%)

https://wiki.php.net/rfc/abstract_syntax_tree

Null coalesce operator

```
if (isset($_GET['foo'])) {
  return $_GET['foo'];
}
return 'default';
```

```
return $_GET['foo'] ?? 'default';
```

Types

Scalar type hints

```
function increase_volume(int $increment) {
    // do something.
}
```

Return types

```
function get_score(): int {
    return 110;
}
```

- Must return an integer.
- Can't return `null`.

Void return type (PHP 7.1)

```
function change_something(): void {
  return;
}
```

- Must not return anything.
- `return null;` is not allowed.

Nullable Return types (PHP 7.1)

```
function get_score(): ?int {
   return 110;
   return null; // or like this
}
```

- Must return an `integer` or explicit `null`.
- `return;` is not allowed.

Nullable function arguments (7.1)

```
function increase_volume(?int $increment) {
    // do something.
}

increase_volume(); // \ArgumentCountError
Increase_volume(null); // Works.
```

Iterable pseudo type (PHP 7.1)

```
function iterate(iterable $var) {
    foreach ($var) {
        // do something.
    }
}
```

Object pseudo type (PHP 7.2)

```
function iterate(object $var) {
  return $var->key;
}
```

Exceptions

Exception hierarchy

- New `\Throwable` base interface
- Cannot directly implement `\Throwable`
- Most fatal errors are now catch-able exceptions
- `ParseError`, `TypeError`, etc

Exception hierarchy

\Throwable

\Error
- ArithmeticError
- DivisionByZeroError
- AssertionError
- ParseError
- TypeError
- ArgumentCountError

\Exception
- ErrorException
- LogicException …
- RuntimeException

Catch multiple exceptions (PHP 7.1)

```
try {
  something_nasty();
}
catch (\PDOException | \FooExceotion) {
  // deal with this.
}
```

Be considerate.

DRUPALCON VIENNA

OOP Improvements

Anonymous classes

```
$foo = new class extends someClass implements someThingElse{
  public function bar() {}
}
```

- Quick mockups in tests
- Cleaner closures

Class constant visibility

```
Class Foo {
  private const BAR = ‘baz’
}
```

Parameter no-type variance (PHP 7.2)

```
class Foo {
  public function setId(int $id) {}
}

class Bar extends Foo {
  public function setId($id) {}
}
```

Security

CSPRNG

Cryptographically Secure Pseudo Random Number Generator

- `random_int()`
- `random_bytes()`

Filtered `unserialize()`

Prevent `unserialize()` function from instantiating classes or whitelist certain classes.

```
unserialize(['allowed_classes' => false]);
```

No not pass user-input to unserialize() even with this!

Deprecations & Removals

Removed extensions

- `mysql` – Old and insecure
- `mcrypt` – Not maintained for years
- `ereg` – Replaced with `preg_*`.
- `call_user_method()`, `call_user_method_array()` - Use `call_user_func()`, `call_user_func_array()`, or variadic parameters.

Deprecations

- Accessing variables without `$` sign.
- `__autoload()` – Use SPL autoloader.
- `count($var)` on uncountable variables.
- `each()` – Use foreach(), key(), or current().
- `create_function()` – Use closures.

DRUPALCON VIENNA

More!

Uniform Variable Syntax

```
$foo->$bar['baz']->something[0]->something_else

($foo)
  ($bar)
    ('baz')
      (something)
        (0)
          (something_else)
```

Always left to right
Use {curly braces}.

Asynchronous signal handling (7.1)

For command-line executions

```
declare(ticks = 1);
pcntl_signal(SIGTERM, function ($signal) {
            echo ‘Good bye’;
});

--

pcntl_async_signals(true);
pcntl_signal(SIGTERM, function ($signal) {
            echo ‘Good bye’;
});
```

Danke.
Fragen?

DRUPALCON VIENNA

JOIN US FOR CONTRIBUTION SPRINT

Friday, 29 September, 2017

Mentored Core Sprint

9:00-18:00
Room: Stolz 2

First time Sprinter Workshop

9:00-12:00
Room: Lehar 1 - Lehar 2

General Sprint

9:00-18:00
Room: Mall

#drupalsprints


WHAT DID YOU THINK?

Locate this session at the DrupalCon Vienna website:

http://vienna2017.drupal.org/schedule

Take the survey!

https://www.surveymonkey.com/r/drupalconvienna