PHP 8

What's New and Changed

DrupalCon EUROPE 2020 DECEMBER 8-11

Ayesh Karunaratne | https://ayesh.me/talk/php8-dceu

What's New and Changed

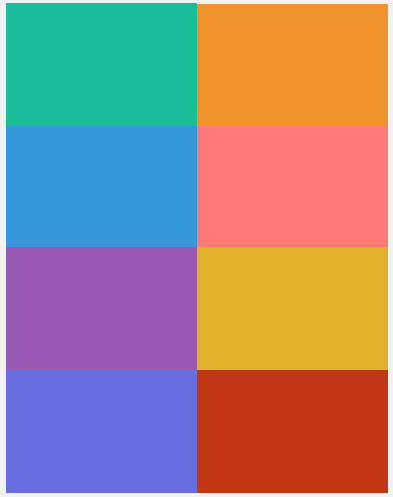
https://ayesh.me/talk/php8-dceu

php8 What's New and Changed

- Major New Features
- New Functions and Classes
- OOP Improvements
- Type System Improvements
- Error Handling Improvements
- Resource to Object Migration
- String Related Changes
- Removed Functionality

https://ayesh.me/talk/php8-dceu

php8 What's New and Changed



https://ayesh.me/talk/php8-dceu

Ayesh Karunaratne

Freelance Software Developer, Security Researcher, Full-time traveler

Kandy, Sri Lanka - Everywhere

https://ayesh.me | https://php.watch

Ayesh

@Ayeshlive | @phpwch

ayesh@ayesh.me

What's New and Changed

- Major New Features
- New Functions and Classes
- OOP Improvements
- Type System Improvements
- Error Handling Improvements
- Resource to Object Migration
- String Related Changes
- Removed Functionality

php8 Major New Features

Named Params

Named parameters in function/method calls in addition to traditional positional parameters.

Named Params

Named parameters in function/method calls in addition to traditional positional parameters.

```
function str_replace($search, $replace, $subject) {
}
```

```
str_replace('Foo', 'Bar', 'Baz');
```

```
str_replace(search: 'Foo', replace: 'Bar', subject: 'Baz');
str_replace(replace: 'Bar', subject: 'Baz', search: 'Foo');
```

Named Params

Named parameters in function/method calls in addition to traditional positional parameters.

```
function setcookie($name, $value = "", $expires = 0, $path, $domain, $secure, $httponly) {
}
```

```
setcookie('foo', 'Bar', 0, '', '', true, true);
```

```
setcookie('foo', 'Bar',
        path: '',
        domain: '',
        secure: true,
        httponly: true
);
```

Named Params

Named parameters in function/method calls in addition to traditional positional parameters.

https://php.watch/versions/8.0/named-parameters

```
setcookie('foo', 'Bar',
        path: '',
        domain: '',
        secure: true,
        httponly: true
);
```

Attributes

Declare **meta-data** for functions, classes, properties, parameters, and constants

Attributes

Declare **meta-data** for functions, classes, properties, parameters, and constants

```
#[Deprecated]
function drupal_set_message(string $message) {
}
```

Attributes

Declare **meta-data** for functions, classes, properties, parameters, and constants

```
#[Deprecated]
function drupal_set_message(string $message) {
}

#[Deprecated]
Class MailManager {

        #[Foo(42)]
        public string $test;

        #[Bar, Baz]
        public function doSomething(): {}

        public function doAnother(#[Bar, Baz] $foo) {}
}
```

Attributes

Declare **meta-data** for functions, classes, properties, parameters, and constants

```php
/**
 * Provides a 'Hello' Block.
 *
 * @Block(
 *   id = "hello_block",
 *   label = "Hello block",
 * )
 */
class HelloBlock extends BlockBase {
```

```php
/**
 * Provides a 'Hello' Block.
 */
#[Block(
    id: "hello_block",
    label: "Hello block",
)]
class HelloBlock extends BlockBase {
```

Attributes

Declare **meta-data** for functions, classes, properties, parameters, and constants

```php
/**
 * Provides a 'Hello' Block.
 */
#[Block(
    id: "hello_block",
    label: "Hello block",
)]
class HelloBlock extends BlockBase {
```

```php
#[Attribute]
Class Block {
    public string $id;
    public string $label;
    public function __construct($id, $label) {
        $this->id = $id;
        $this->label = $label;
    }
}
```

Attributes

Declare **meta-data** for functions, classes, properties, parameters, and constants

```
$reflector = new \ReflectionClass(Foo::class);
$reflector->getAttributes();
$reflector->getArguments();
$reflector->newInstance();
```

Attributes

Declare **meta-data** for functions, classes, properties, parameters, and constants

https://php.watch/versions/8.0/attributes
https://php.watch/articles/php-attributes

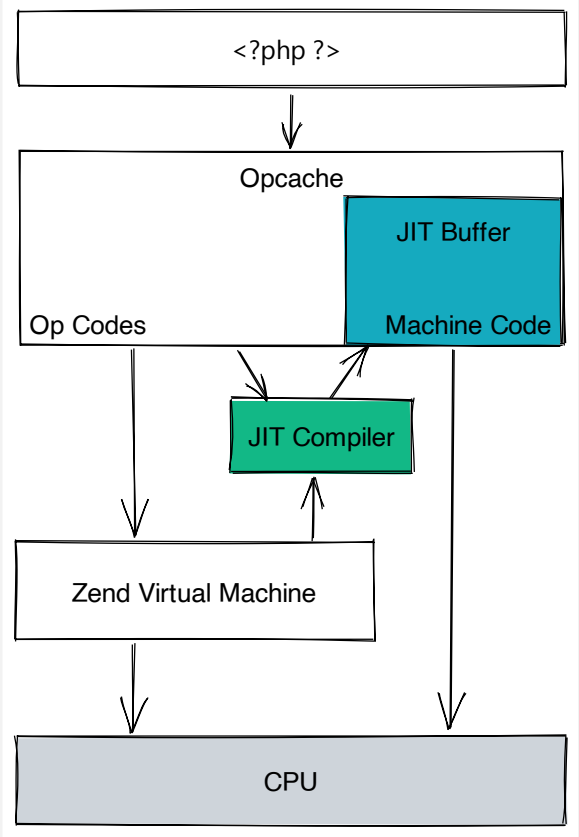
```
#[Deprecated]
function drupal_set_message(string $message) {
}

$reflector = new \ReflectionClass(Foo::class);
$reflector->getAttributes();
$reflector->getArguments();
$reflector->newInstance();
```

Just-In-Time (JIT) Compiler

Compile and cache CPU machine code, and skipping PHP virtual machine altogether.

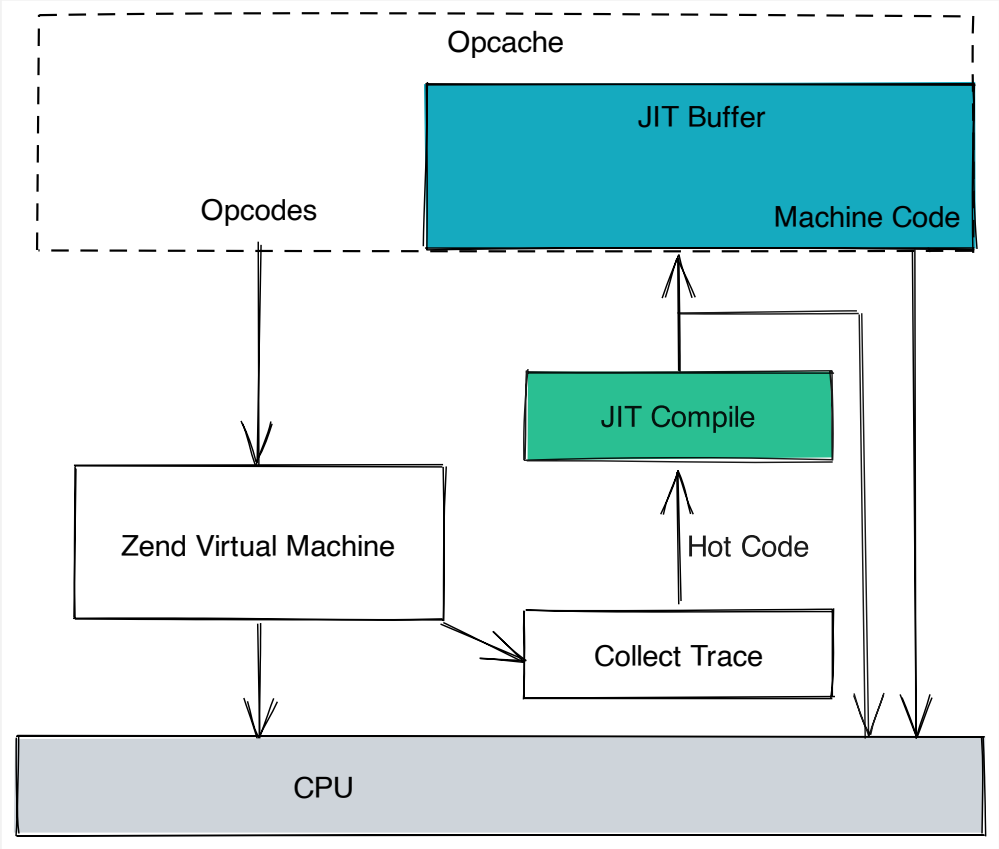
Just-In-Time (JIT) Compiler

Compile and cache CPU machine code, and skipping PHP virtual machine altogether.



Just-In-Time (JIT) Compiler

Compile and cache CPU machine code, and skipping PHP virtual machine altogether.



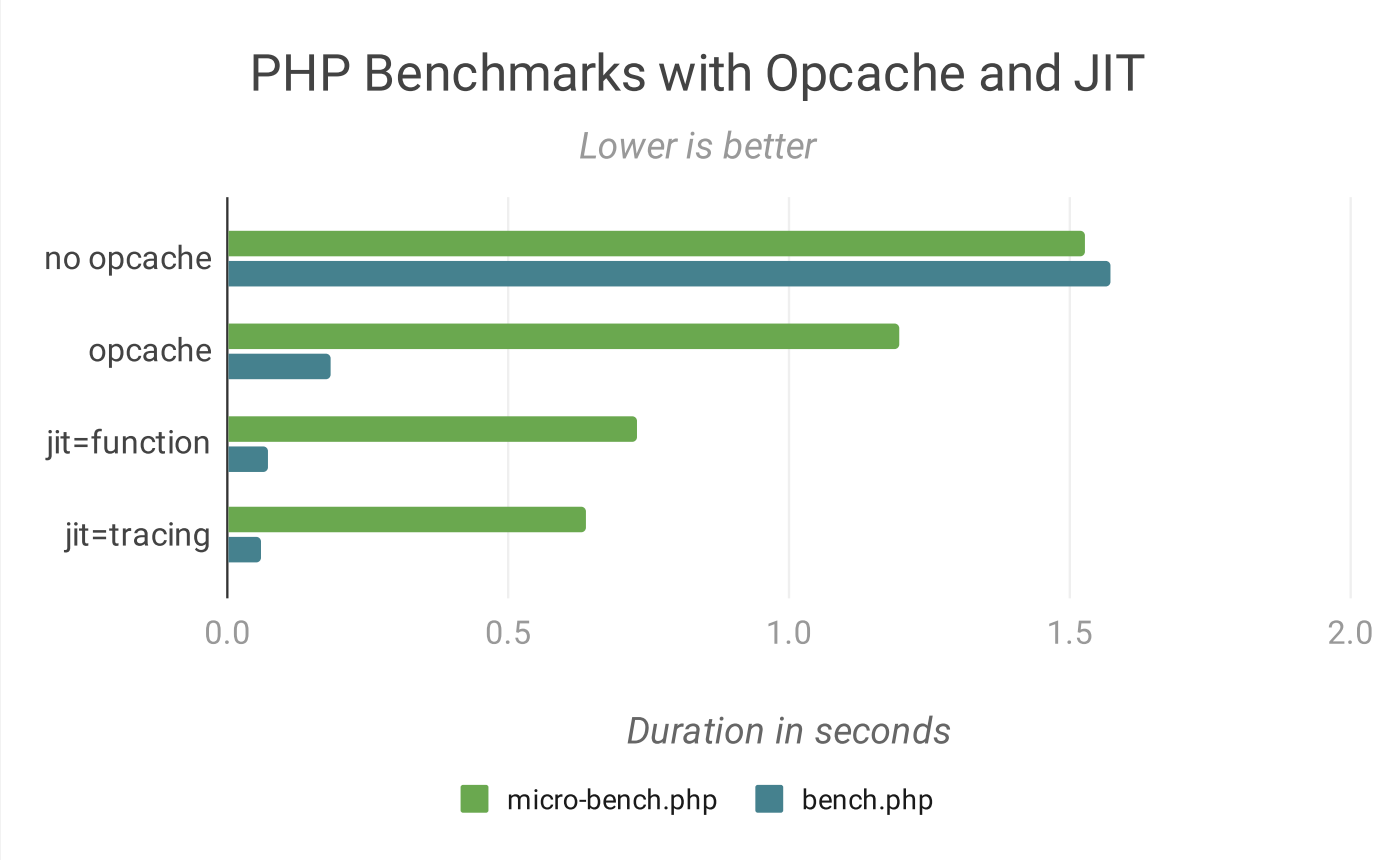
Just-In-Time (JIT) Compiler

Compile and cache CPU machine code, and skipping PHP virtual machine altogether.

```
[opcache]
opcache.enable=1
opcache.jit_buffer_size=100M
```

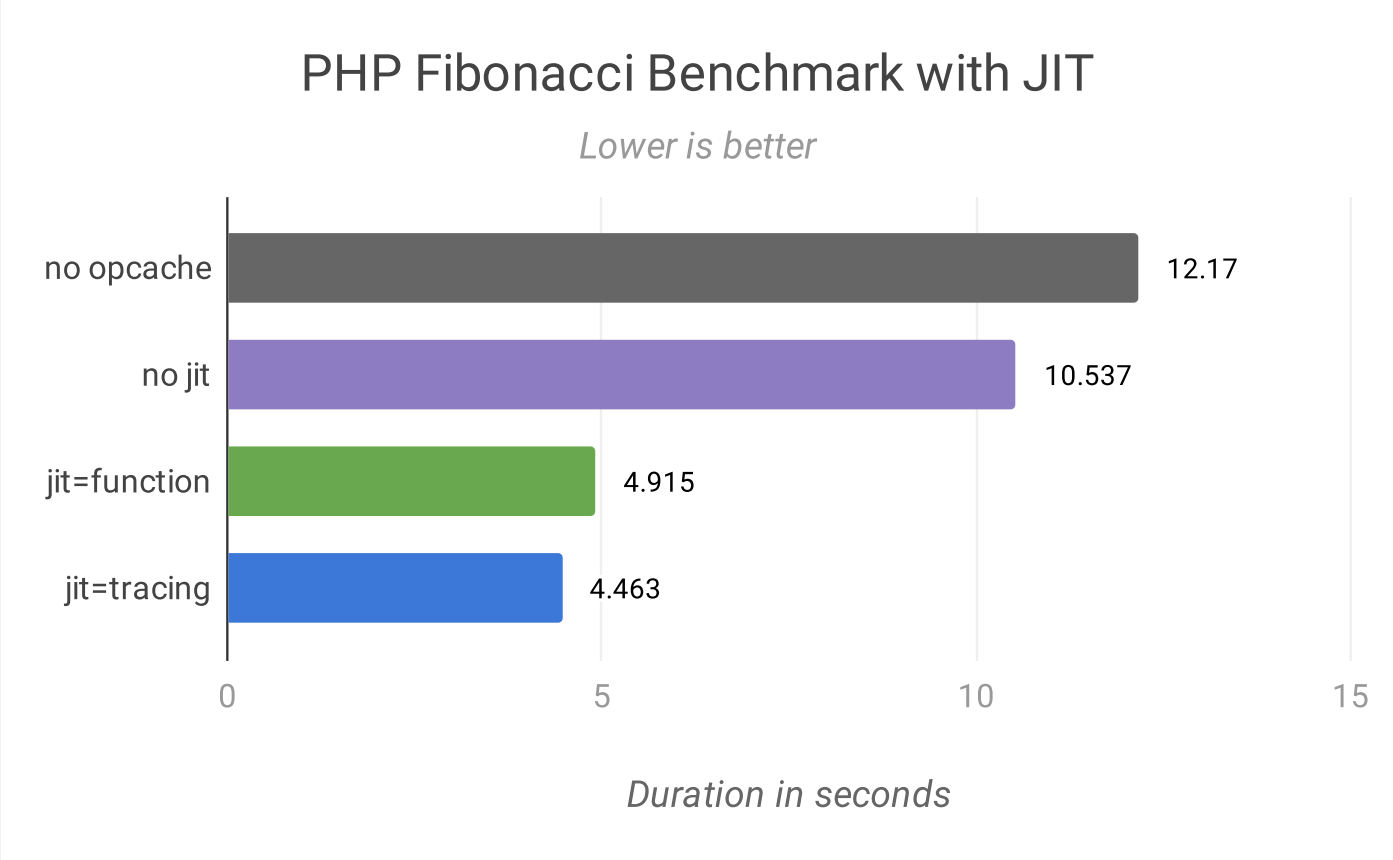
Just-In-Time (JIT) Compiler

Compile and cache CPU machine code, and skipping PHP virtual machine altogether.



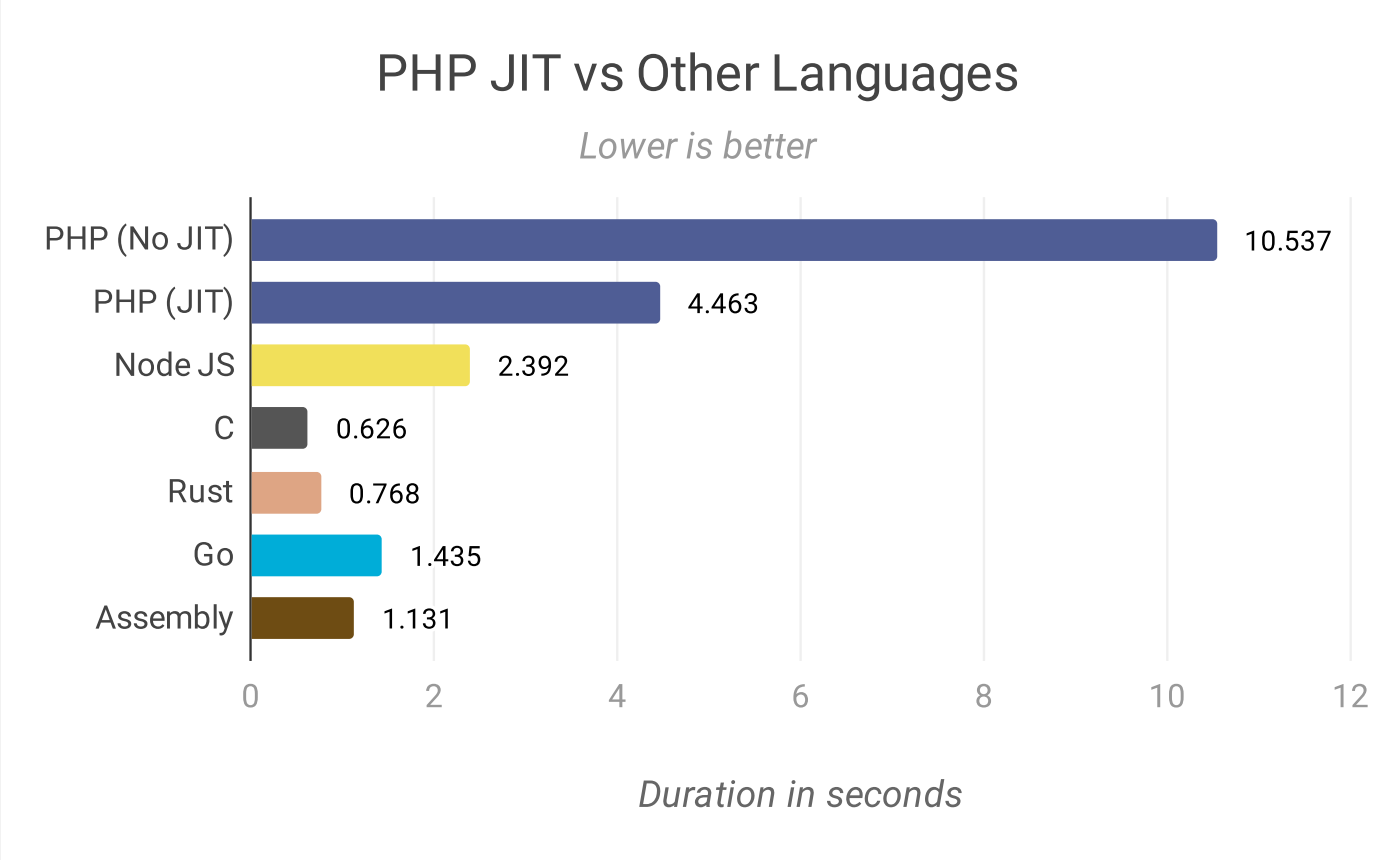
Just-In-Time (JIT) Compiler

Compile and cache CPU machine code, and skipping PHP virtual machine altogether.



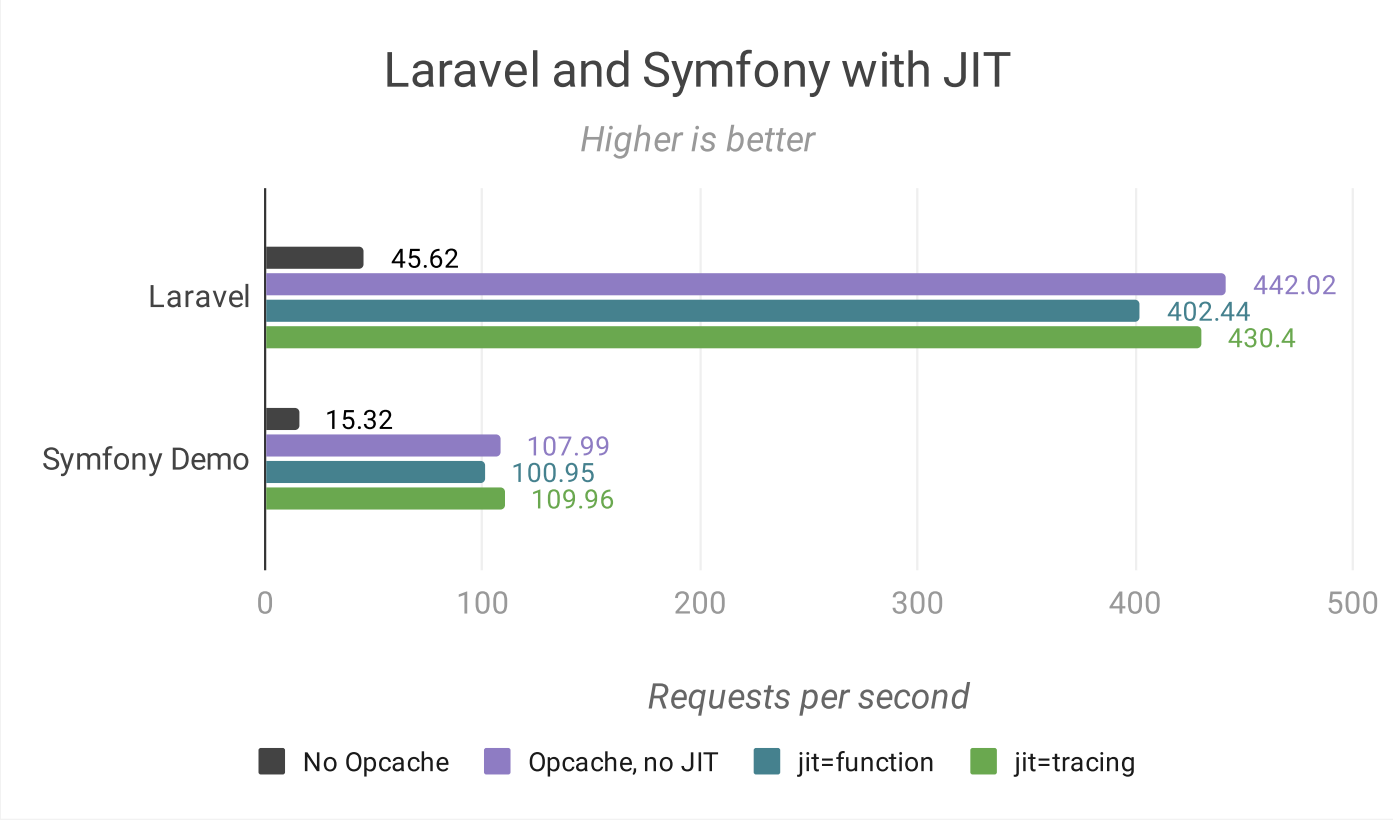
Just-In-Time (JIT) Compiler

Compile and cache CPU machine code, and skipping PHP virtual machine altogether.



Just-In-Time (JIT) Compiler

Compile and cache CPU machine code, and skipping PHP virtual machine altogether.



Just-In-Time (JIT) Compiler

Compile and cache CPU machine code, and skipping PHP virtual machine altogether.

https://php.watch/versions/8.0/JIT
https://php.watch/articles/jit-in-depth

```
[opcache]
opcache.enable=1
opcache.jit_buffer_size=100M
```

Constructor Properties

A new syntax to **declare** and **assign** class properties from the **constructor**

Constructor Properties

A new syntax to **declare** and **assign** class properties from the **constructor**

```
class User {
        public int $uid;
        public string $username;

        public function __construct(int $uid, string $username) {
                $this->$uid = $uid;
                $this->username = $username;
        }
}
```

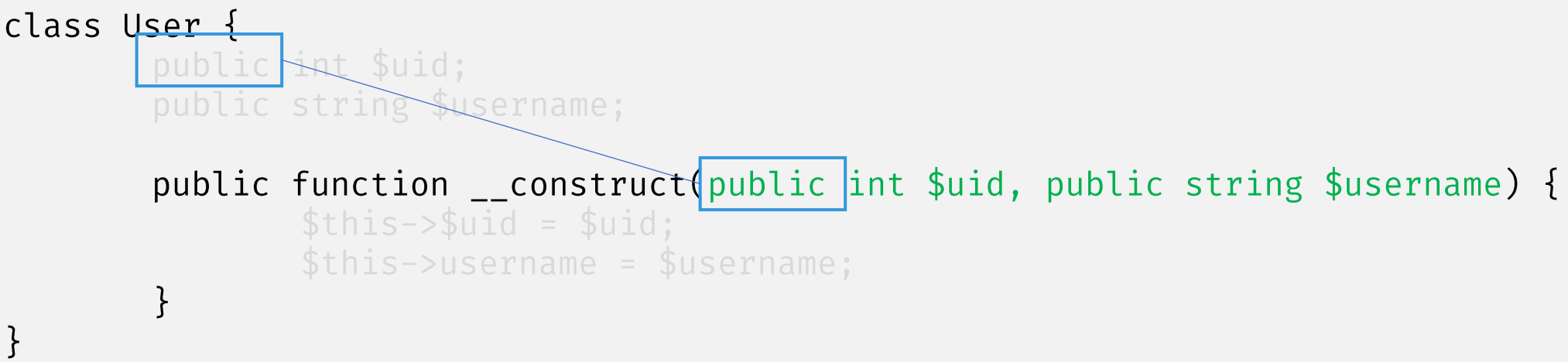
Constructor Properties

A new syntax to **declare** and **assign** class properties from the **constructor**

```
class User {
        public int $uid;
        public string $username;

        public function __construct(int $uid, string $username) {
                $this->$uid = $uid;
                $this->username = $username;
        }
}
```

Constructor Properties

A new syntax to **declare** and **assign** class properties from the **constructor**



Constructor Properties

A new syntax to **declare** and **assign** class properties from the **constructor**

```
class User {
        public function __construct(public int $uid, public string $username) {}
}
```

Constructor Properties

A new syntax to **declare** and **assign** class properties from the **constructor**

https://php.watch/versions/8.0/constructor-property-promotion

```
class User {
        public function __construct(public int $uid, public string $username) {}
}
```

Union Types

Dynamically declare a combination of types as a single type

Union Types

Dynamically declare a combination of types as a single type

```
function node_load($nid) {

}
```

Union Types

Dynamically declare a combination of types as a single type

```
/**
 * @param int|array
 * @return stdClass|bool
 */
function node_load($nid) {

}
```

Union Types

Dynamically declare a combination of types as a single type

```php
/**
 * @param int|array
 * @return stdClass|bool
 */
function node_load(int|array $nid): stdClass|bool {

}
```

Union Types

Dynamically declare a combination of types as a single type

```php
/**
 * @param int|array
 * @return stdClass|bool
 */
function node_load(int|array $nid): stdClass|false {

}
```

Union Types

Dynamically declare a combination of types as a single type

```
function node_load(int|array $nid): stdClass|false {

}
```

Union Types

Dynamically declare a combination of types as a single type

https://php.watch/versions/8.0/union-types

```
function node_load(int|array $nid): stdClass|false {

}
```

Null-Safe Operator

Optional chaining property access operator

Null-Safe Operator

Optional chaining property access operator

```
$customer->getAddress()->getCountry();
```

```
$address = $customer ->getAddress();
if ($address) {
    $country = $address ->getCountry();
}
else {
    $country = null;
}
```

Null-Safe Operator

Optional chaining property access operator

```
$customer->getAddress()->getCountry();
```

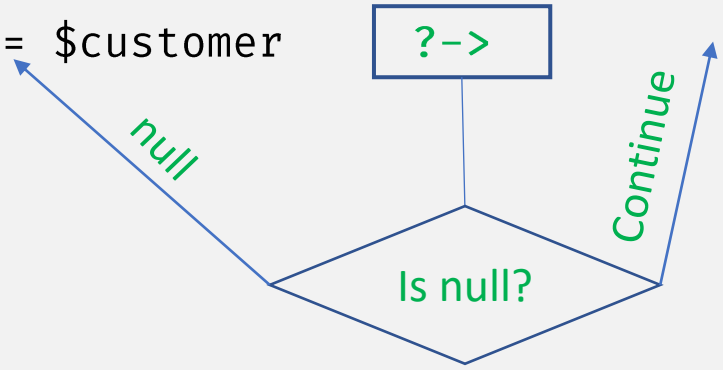
```
$country = $customer?->getAddress()?->getCountry();
```

Null-Safe Operator

Optional chaining property access operator

```
$customer->getAddress()->getCountry();
```

```
$country = $customer ?-> getAddress()?->getCountry();
```



Null-Safe Operator

Optional chaining property access operator

https://php.watch/versions/8.0/null-safe-operator

```
$country = $customer?->getAddress()?->getCountry();
```

`match` Expressions

`switch` block syntax with strict comparison and return value support

`match` Expressions

`switch` block syntax with strict comparison and return value support

```
switch ($request_method) {
        case 'post':
                $status = $this->handlePost();
                break;
        case 'get':
        case 'head':
                $status = $this->handleGet();
                break;
        default:
                throw new \Exception('Unsupported');
}
```

`match` Expressions

`switch` block syntax with strict comparison and return value support

```
switch ($request_method) {
        case 'post':
                $status = $this->handlePost();
                break;
        case 'get':
        case 'head':
                $status = $this->handleGet();
                break;
        default:
                throw new \Exception('Unsupported');
}
```

`match` Expressions

`switch` block syntax with strict comparison and return value support

```
switch ($request_method) {
        case 'post':
                $status = $this->handlePost();
                break;
        case 'get':
        case 'head':
                $status = $this->handleGet();
                break;
        default:
                throw new \Exception('Unsupported');
}
```

`match` Expressions

`switch` block syntax with strict comparison and return value support

```
switch ($request_method) {
        case 'post':
                $status = $this->handlePost();
                break;
        case 'get':
        case 'head':
                $status = $this->handleGet();
                break;
        default:
                throw new \Exception('Unsupported');
}
```

`match` Expressions

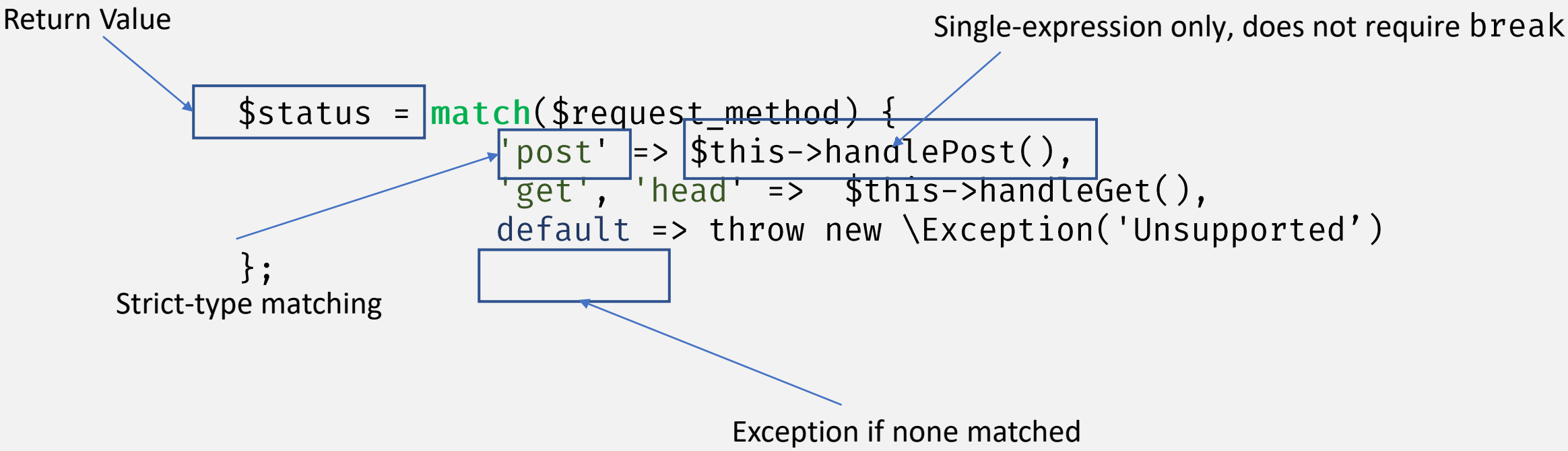
`switch` block syntax with strict comparison and return value support

```
switch ($request_method) {
        case 'post':
                $status = $this->handlePost();
                break;
        case 'get':
        case 'head':
                $status = $this->handleGet();
                break;
        default:
                throw new \Exception('Unsupported');
}
```

```
$status = match($request_method) {
        'post' => $this->handlePost(),
        'get', 'head' =>  $this->handleGet(),
        default => throw new \Exception('Unsupported')
};
```

`match` Expressions

`switch` block syntax with strict comparison and return value support



`match` Expressions

`switch` block syntax with strict comparison and return value support

https://php.watch/versions/8.0/match-expression

```
$status = match($request_method) {
            'post' => $this->handlePost(),
            'get', 'head' =>  $this->handleGet(),
            default => throw new \Exception('Unsupported')
};
```

WeakMaps

Store data for objects without blocking the garbage collector

WeakMaps

Store data for objects without blocking the garbage collector

```
$cache = new WeakMap();
$post = Entity::getPost(42);
$cache[$post] = Entity::getComments($post);
```

WeakMaps

Store data for objects without blocking the garbage collector

https://php.watch/versions/8.0/weakmap
https://php.watch/articles/practical-weakmap

```
$cache = new WeakMap();
$post = Entity::getPost(42);
$cache[$post] = Entity::getComments($post);
```

php8 Major New Features

Named Params	**Named** parameters in function/method calls in addition to traditional positional parameters.
Attributes	Declare **meta-data** for functions, classes, properties, parameters, and constants
Just-In-Time (JIT) Compiler	Compile and cache CPU machine code, and skipping PHP virtual machine altogether.
Constructor Properties	A new syntax to **declare** and **assign** class properties from the **constructor**
Union Types	Dynamically declare a combination of types as a single type
Null-Safe Operator	Optional chaining property access operator
`match` Expressions	`switch` block syntax with strict comparison and return value support
WeakMaps	Store data for objects without blocking the garbage collector

php8 New Functions and Classes

PHP 8 New Functions and Classes

str_contains

Determine if a string **contains** a given substring
https://php.watch/versions/8.0/str_contains

```
str_contains('Foobar', 'Foo’);
```

str_starts_with

Checks if a string **starts** with a given substring
https://php.watch/versions/8.0/str_starts_with-str_ends_with

```
str_starts_with('PHP 8.0', 'PHP)
```

str_ends_with

Checks if a string **ends** with a given substring
https://php.watch/versions/8.0/str_starts_with-str_ends_with

```
str_ends_with('PHP 8.0', '8.0)
```

php8 New Functions and Classes

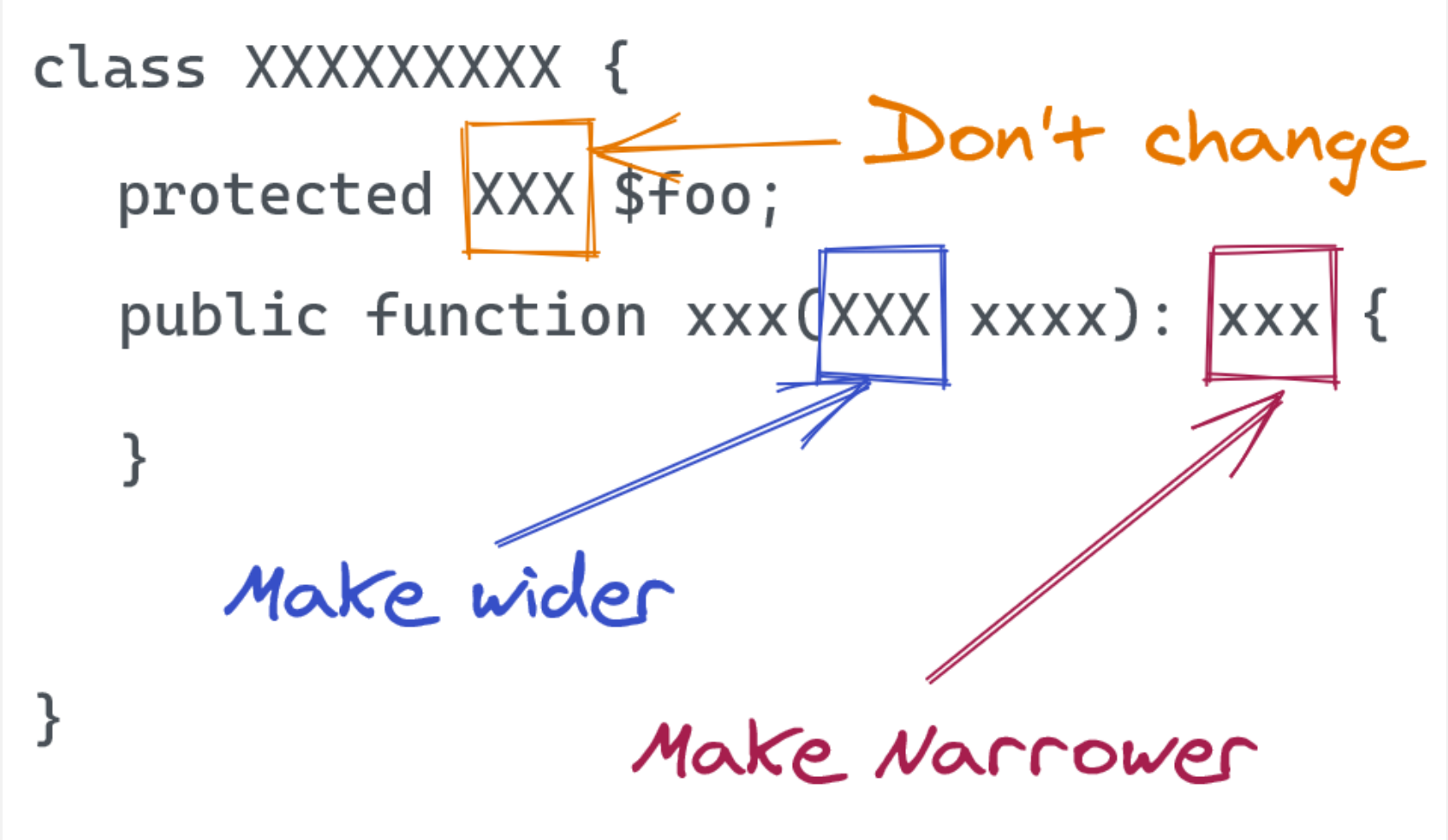
`fdiv`	Float division supporting IEEE-754 standard on Floating-Point Arithmetic.
`get_resource_id`	Returns the internal ID for a given PHP `resource`
`get_debug_type`	Returns the internal type of a passed variable
`preg_last_error_msg`	Returns a human-friendly error message for the last preg operation.

`Stringable`	The new Stringable `interface` is automatically added to all classes that implement __toString method, and those explicitly declare that they implements Stringable.
`PhpToken`	The new `PhpToken` class provides a more fluent Object-Oriented interface as an alternative to the legacy array-based `to token_get_all` function.

PHP 8 OOP Improvements

Liskov Substitution Principle

Superclass shall be replaceable with objects of its subclasses without breaking the application



Liskov Substitution Principle Is Enforced Strictly

```
class Foo {
    public function process(stdClass $item): array{}
}

class SubFoo extends Foo{
    public function process(array $items): array{}
}
```

Liskov Substitution Principle Is Enforced Strictly

```
class Foo {
    public function process(stdClass $item): array{}
}

class SubFoo extends Foo{
    public function process(array $items): array{}
}
```

```
Fatal error: Declaration of SubFoo ::process(array $items): array must be
compatible with Foo::process(stdClass $item): array in ... on line ...
```

php8 OOP Improvements

Liskov Substitution Principle Is Enforced Strictly

All extending classes and interface implementations must follow LSP; Parameters can get wider, return types can get narrower, and property types should not change.
https://php.watch/versions/8.0/lsp-errors , https://php.watch/articles/php-lsp

Magic Methods must implement correct signature

If magic methods declare types, they must be implement the expected signature.
https://php.watch/versions/8.0/magic-method-signatures

`private` methods/properties are exempt from LSP

Previously, changing static flag was not allowed on private methods and properties. It is not enforced in PHP 8.
https://php.watch/versions/8.0/final-private-function

PHP 8 Type System Improvements

PHP 8 Type System Improvements

Union Types

Dynamically declare a combination of types as a single type.

https://php.watch/versions/8.0/union-types

```
function node_load(int|array $nid): stdClass|false {
}
```

`mixed` type

Indicates a union type of all PHP types, excluding `void`.

https://php.watch/versions/8.0/mixed-type

```
Function cache_get(string $key): mixed {
}
```

`static` type

declares an object of the called class will be returned. Only supported as a return type.

https://php.watch/versions/8.0/static-return-type

```
class Foo {
    public static function getInstance(): static {
        return new static();
    }
}
```

php8 Error Handling Improvements

Internal Functions Error Handling

Internal functions throw `TypeError` and `ValueError` exceptions, some of them upgraded from warnings

Internal Functions Error Handling

Internal functions throw `TypeError` and `ValueError` exceptions, some of them upgraded from warnings
https://php.watch/versions/8.0/internal-function-exceptions

```
substr('foo', []);
```

PHP < 8.0

```
Warning: substr() expects parameter 2 to be int, array given in ... on line ...
```

php8

```
Fatal error: Uncaught TypeError: substr(): Argument #2 ($start) must be of type
int, array given in ...:...
```

Internal Functions Error Handling

Internal functions throw `TypeError` and `ValueError` exceptions, some of them upgraded from warnings
https://php.watch/versions/8.0/internal-function-exceptions

```
json_decode('"foo"', true, -1);
```

PHP < 8.0

```
Warning: json_decode(): Depth must be greater than zero in ... on line ...
```

php8

```
Fatal error: Uncaught ValueError: json_decode(): Argument #3 ($depth) must be
greater than 0 in ...:...
```

Internal Functions Error Handling

Internal functions throw `TypeError` and `ValueError` exceptions, some of them upgraded from warnings
https://php.watch/versions/8.0/internal-function-exceptions

```
method_exists([], 'getName');
```

PHP < 8.0

```
```

php8

```
Fatal error: Uncaught TypeError: method_exists(): Argument #1 ($object_or_class)
must be of type object|string, array given in ...:...
```

Default Error Reporting Changes

Error reporting set to E_ALL, startup errors are shown, the @ suppressor changes, and Assertion changes

Default Error Reporting Changes

Error reporting set to E_ALL, startup errors are shown, the @ suppressor changes, and Assertion changes

Default error reporting set to E_ALL

All errors are reported by default, from the previous option to not report deprecation notices.
https://php.watch/versions/8.0/error-display-E_ALL

Startup errors are shown by default

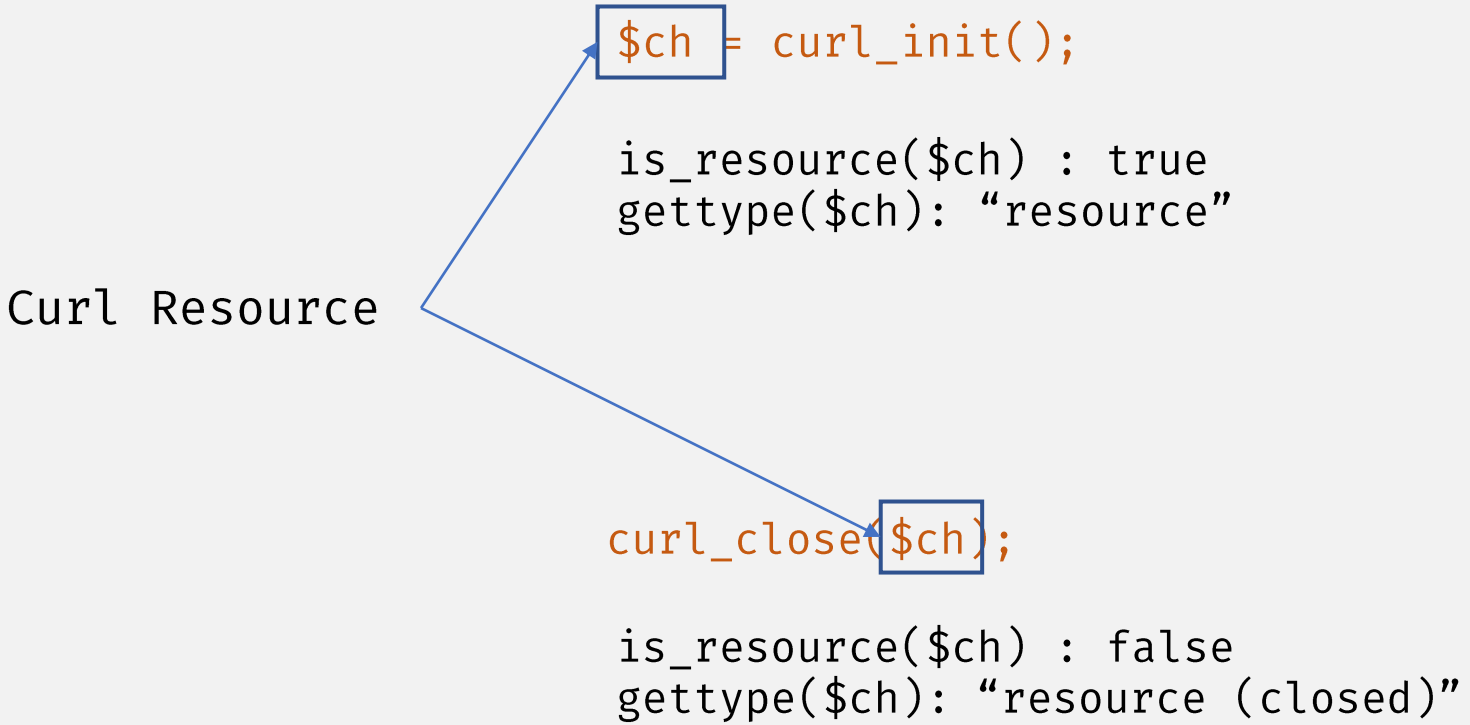
Warnings related to missing extension files, problematic INI configurations, etc. are shown by default.
https://php.watch/versions/8.0/startup-errors-enabled

@ suppression operator does not silent fatal errors

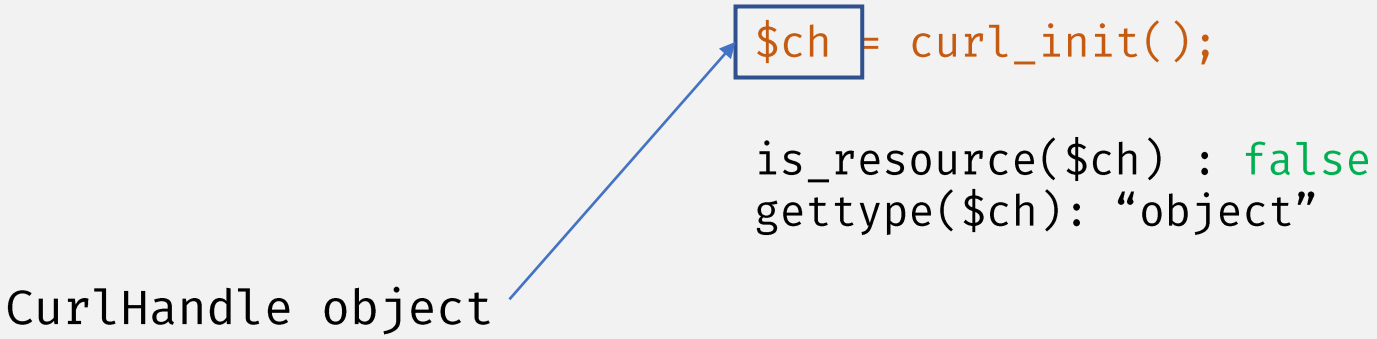
`@test()` function calls will display the error message if a fatal error is thrown.
https://php.watch/versions/8.0/fatal-error-suppression

php8 Resource to Object Migration

Resource to Object Migration



Resource to Object Migration



Resource to Object Migration

Extension	Resource Name (PHP < 8.0)	Object name (PHP >= 8.0)
Curl	Curl	CurlHandle
Curl	curl_multi	CurlMultiHandle
Curl	curl_share	CurlShareHandle
GD	gd	GdImage
Sockets	Socket	Socket
Sockets	AddressInfo	AddressInfo
OpenSSL	OpenSSL key	OpenSSLAsymmetricKey
OpenSSL	OpenSSL X.509	OpenSSLCertificate
OpenSSL	OpenSSL X.509 CSR	OpenSSLCertificateSigningRequest
XMLWriter	xmlwriter	XMLWriter
XML	xml	XMLParser

php8 String Related Changes

An empty string in every string

PHP considers every string contains empty strings between two characters

```
strpos('Foo', '');
```

PHP < 8.0

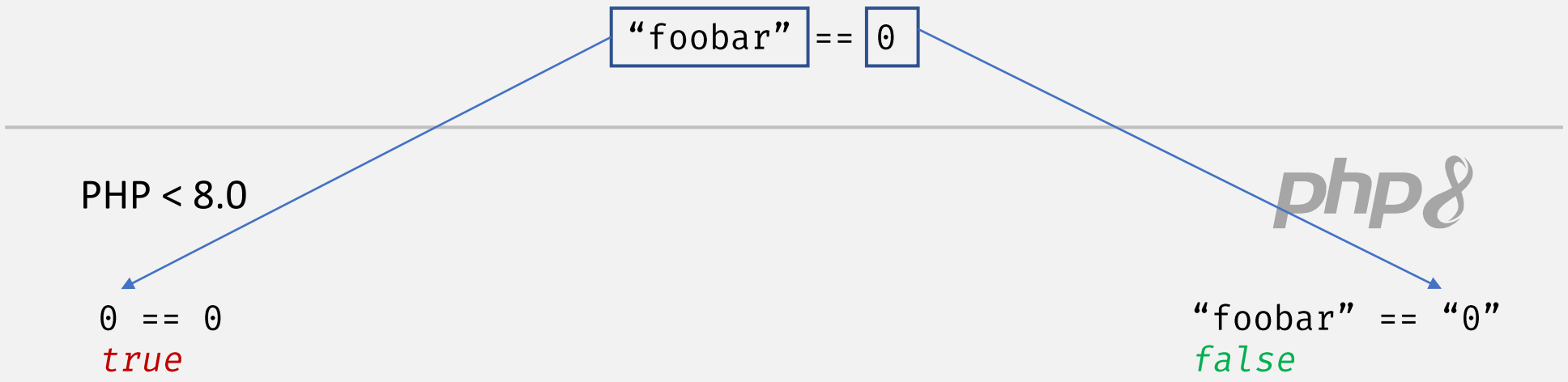
```
Warning: strpos(): Empty needle in /in/s39og on line 3
false
```

php8

```
true
```

Numeric string changes

Numeric string comparisons does not coerce non-numeric strings to numbers when comparing



php8 Removed Functionality

php8 Removed Functionality

- create_function() function
- each() function
- __autoload() autoloader support
- Curly brace array-access syntax
- XMLRPC extension (moved to PECL)
- string.strip_tags filter
- money_format() function
- get_magic_quotes_gpc() function
- get_magic_quotes_runtime()

Further Resources

- https://www.php.net/releases/8.0/en.php
- https://php.watch/versions/8.0
- https://www.drupal.org/project/issues/search?issue_tags=PHP%208.0
- https://www.drupal.org/project/drupal/issues/3109885
- https://wiki.php.net/rfc
- https://www.drupal.org/project/drupal/issues/3145797
- https://github.com/php/php-src/milestone/1
- https://github.com/php/php-src/milestone/3

Questions?

No question is too small.

@Ayeshlive ayesh@php.watch

https://ayesh.me/talk/php8-dceu

DrupalCon
EUROPE2020
DECEMBER 8-11

Join us for contribution opportunities

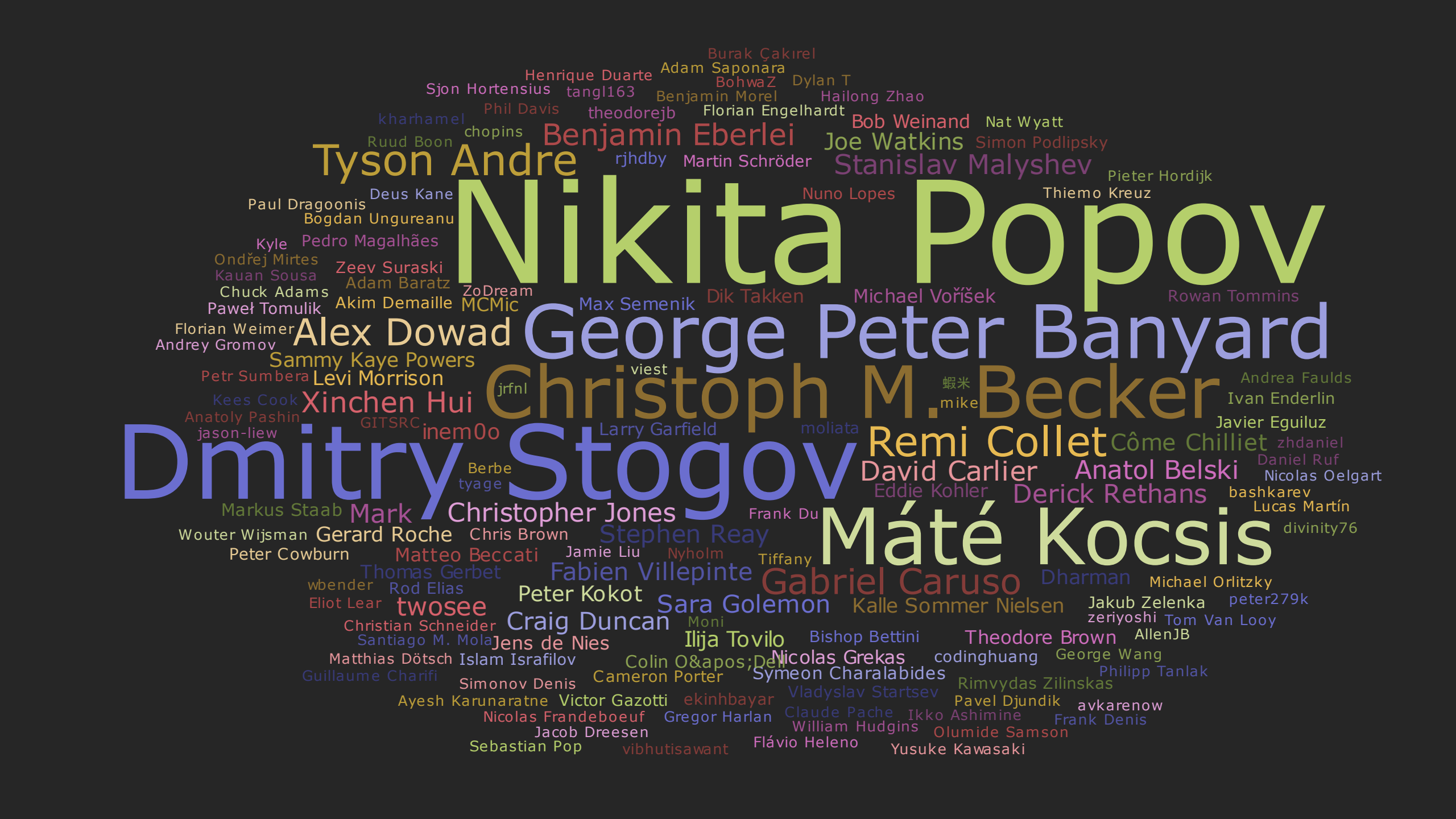
Friday, December 11, 2020

Mentored Contribution
9h00-18h00 CET

First Time Contributor Workshop
10h00-12h00 CET

General Contribution
9h00-18h00 CET

#DrupalContributions


Nikita Popov
George Peter Banyard
Christoph M. Becker
Dmitry Stogov
Máté Kocsis
Tyson Andre
Benjamin Eberlei
Joe Watkins
Stanislav Malyshev
Alex Dowad
Remi Collet
David Carlier
Anatol Belski
Derick Rethans
Christopher Jones
Mark
Gabriel Caruso
Fabien Villepinte
Stephen Reay
Sara Golemon
Craig Duncan
twosee
Peter Kokot
Jens de Nies
Ilija Tovilo
Nicolas Grekas
Symeon Charalabides
Theodore Brown
Xinchen Hui
inem0o
Levi Morrison
Sammy Kaye Powers
Gerard Roche
Matteo Beccati
Thomas Gerbet
Rod Elias
Kalle Sommer Nielsen
Dharman
Michael Orlitzky
Jakub Zelenka
Tom Van Looy
AllenJB
Bishop Bettini
George Wang
Philipp Tanlak
Rimvydas Zilinskas
Pavel Djundik
avkarenow
Frank Denis
Olumide Samson
Yusuke Kawasaki
Flávio Heleno
Vladyslav Startsev
Victor Gazotti
Ayesh Karunaratne
Cameron Porter
Colin O'Dell
Matthias Dötsch
Islam Israfilov
Christian Schneider
Santiago M. Mola
Guillaume Charifi
Simonov Denis
Nicolas Frandeboeuf
Gregor Harlan
Jacob Dreesen
Sebastian Pop
vibhutisawant
William Hudgins
Claude Pache
Ikko Ashimine
ekinhbayar
Moni
Eddie Kohler
Frank Du
Tiffany
Jamie Liu
Nyholm
Chris Brown
Peter Cowburn
Wouter Wijsman
Markus Staab
Berbe
tyage
Anatoly Pashin
jason-liew
GITSRC
Kees Cook
Petr Sumbera
Florian Weimer
Andrey Gromov
Paweł Tomulik
Chuck Adams
Kauan Sousa
Ondřej Mirtes
Kyle
Paul Dragoonis
Deus Kane
Bogdan Ungureanu
Pedro Magalhães
Zeev Suraski
Adam Baratz
Akim Demaille
ZoDream
MCMic
Max Semenik
Dik Takken
Michael Vořišek
Rowan Tommins
Nuno Lopes
Thiemo Kreuz
Pieter Hordijk
Andrea Faulds
Ivan Enderlin
Javier Eguiluz
Côme Chilliet
zhdaniel
Daniel Ruf
Nicolas Oelgart
bashkarev
Lucas Martín
divinity76
peter279k
zeriyoshi
Larry Garfield
moliata
mike
蝦米
jrfnl
viest
Ruud Boon
chopins
kharhamel
Phil Davis
theodorejb
Florian Engelhardt
Bob Weinand
Nat Wyatt
Simon Podlipsky
rjhdby
Martin Schröder
Henrique Duarte
Adam Saponara
Burak Çakırel
Sjon Hortensius
tangl163
Benjamin Morel
BohwaZ
Dylan T
Hailong Zhao


THANK YOU
dziękuję
Ďakujem
tak
arigatô
paldies
mahalo
kop khun
děkuji
diolch
dankie
köszönöm
хвала
cảm ơn bạn
shukran
tänan
a dank
gràcies
ngiyabonga
Баярлалаа
dhanyavād
Дякую
Благодарам
ευχαριστώ
спасибо
благодаря
tack
grazie
Ìh’gōi
Dank u
Благодаря ти
gracias
ස්තුතියි
mulţumesc
ačiū
תודה.
takk
nandri
danke
faleminderit
Xièxiè
choukrane
teşekkür ederim
obrigado
kiitos
Շնորհակալություն
terima kasih
hvala
grazzi

PHP 8

What's New and Changed

DrupalCon EUROPE 2020 DECEMBER 8-11

Ayesh Karunaratne | https://ayesh.me/talk/php8-dceu