php

- Downloads
- Documentation
- Get Involved
- Help
-

Search

Dutch PHP Conference 2021 - Call for Papers

File System Related Extensions
Human Language and Character Encoding Support
Image Processing and Generation
Mail Related Extensions
Mathematical Extensions
Non-Text MIME Output
Process Control Extensions
Other Basic Extensions
Other Services
Search Engine Extensions
Server Specific Extensions
Session Extensions
Text Processing
Variable and Type Related Extensions
Web Services
Windows Only Extensions
XML Manipulation
GUI Extensions

Keyboard Shortcuts

?	This help
j	Next menu item
k	Previous menu item
g p	Previous man page
g n	Next man page
G	Scroll to bottom
g g	Scroll to top
g h	Goto homepage
g s	Goto search (current page)
/	Focus search box

Change language: English

Released!
PHP 8.0 is a major update of the PHP language.
It contains many new features and optimizations including named arguments, union types, attributes, constructor property promotion, match expression, nullsafe operator, JIT, and improvements in the type system, error handling, and consistency.
Go update to PHP 8!

Named arguments RFC ¶

PHP 7
```
htmlspecialchars($string, ENT_COMPAT | ENT_HTML401, 'UTF-8', false);
```
PHP 8
```
htmlspecialchars($string, double_encode: false);
```

- Specify only required parameters, skipping optional ones.
- Arguments are order-independent and self-documented.

Attributes RFC Doc ¶

PHP 7
```
class PostsController
{
    /**
     * @Route("/api/posts/{id}", methods={"GET"})
     */
    public function get($id) { /* ... */ }
}
```

PHP 8

```
class PostsController
{
    #[Route("/api/posts/{id}", methods: ["GET"])]
    public function get($id) { /* ... */ }
}
```

Instead of PHPDoc annotations, you can now use structured metadata with PHP's native syntax.

Constructor property promotion RFC Doc ¶

PHP 7

```
class Point {
  public float $x;
  public float $y;
  public float $z;

  public function __construct(
    float $x = 0.0,
    float $y = 0.0,
    float $z = 0.0
  ) {
    $this->x = $x;
    $this->y = $y;
    $this->z = $z;
  }
}
```

PHP 8

```
class Point {
  public function __construct(
    public float $x = 0.0,
    public float $y = 0.0,
    public float $z = 0.0,
  ) {}
}
```

Less boilerplate code to define and initialize properties.

Union types RFC Doc ¶

PHP 7

```
class Number {
  /** @var int|float */
  private $number;

  /**
   * @param float|int $number
   */
  public function __construct($number) {
    $this->number = $number;
  }
}

new Number('NaN'); // Ok
```

PHP 8

```
class Number {
  public function __construct(
    private int|float $number
  ) {}
}

new Number('NaN'); // TypeError
```

Instead of PHPDoc annotations for a combination of types, you can use native union type declarations that are validated at runtime.

Match expression RFC Doc ¶

PHP 7

```
switch (8.0) {
  case '8.0':
    $result = "Oh no!";
    break;
  case 8.0:
    $result = "This is what I expected";
    break;
}
echo $result;
//> Oh no!
```

PHP 8

```
echo match (8.0) {
  '8.0' => "Oh no!",
  8.0 => "This is what I expected",
};
//> This is what I expected
```

The new match is similar to switch and has the following features:

- Match is an expression, meaning its result can be stored in a variable or returned.
- Match branches only support single-line expressions and do not need a break; statement.
- Match does strict comparisons.

Nullsafe operator RFC ¶

PHP 7

```
$country =  null;

if ($session !== null) {
  $user = $session->user;

  if ($user !== null) {
    $address = $user->getAddress();

    if ($address !== null) {
      $country = $address->country;
    }
  }
}
```

PHP 8

```
$country = $session?->user?->getAddress()?->country;
```

Instead of null check conditions, you can now use a chain of calls with the new nullsafe operator. When the evaluation of one element in the chain fails, the execution of the entire chain aborts and the entire chain evaluates to null.

Saner string to number comparisons RFC ¶

PHP 7

```
0 == 'foobar' // true
```

PHP 8

```
0 == 'foobar' // false
```

When comparing to a numeric string, PHP 8 uses a number comparison. Otherwise, it converts the number to a string and uses a string comparison.

Consistent type errors for internal functions RFC ¶

PHP 7

```
strlen([]); // Warning: strlen() expects parameter 1 to be string, array given

array_chunk([], -1); // Warning: array_chunk(): Size parameter expected to be greater than 0
```

PHP 8

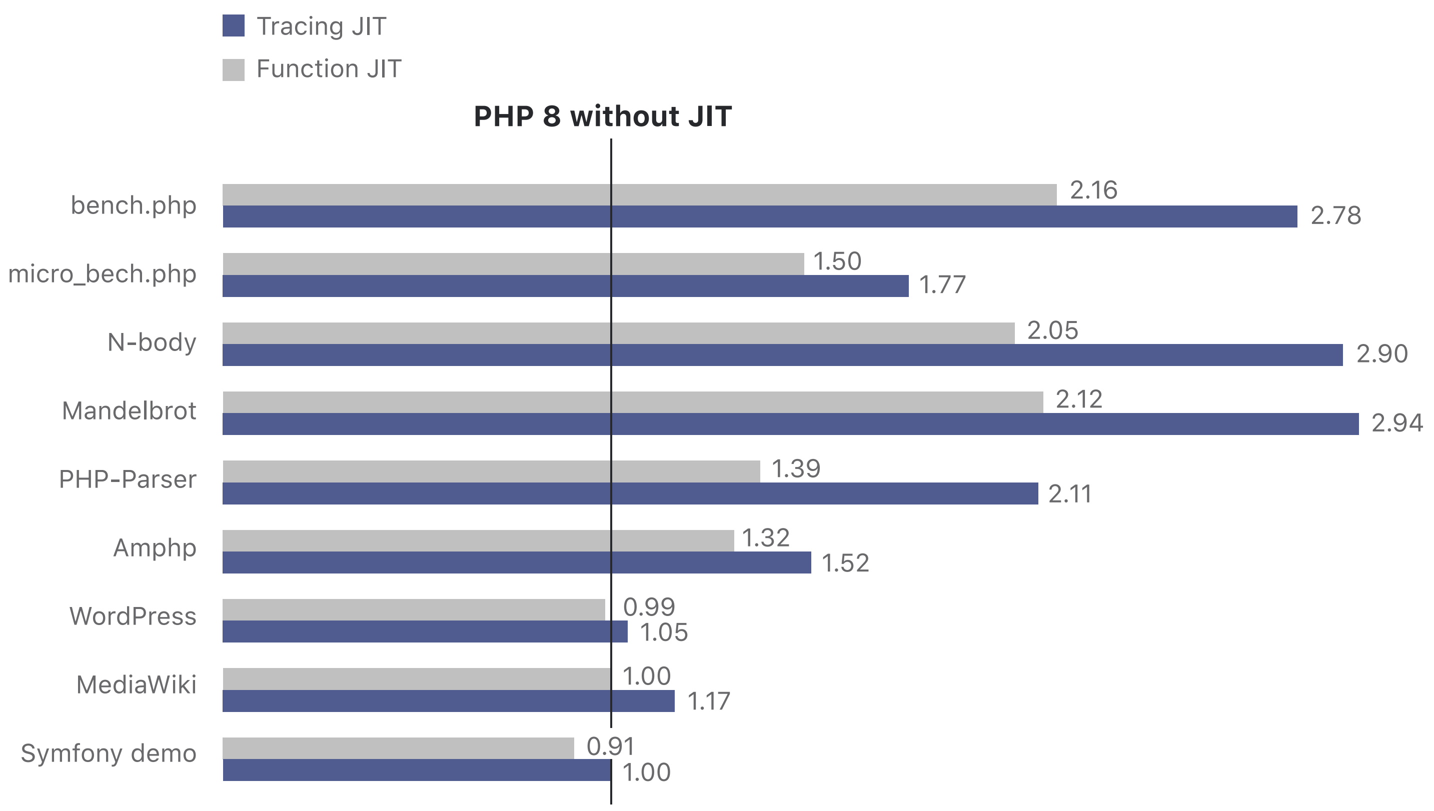
```
strlen([]); // TypeError: strlen(): Argument #1 ($str) must be of type string, array given

array_chunk([], -1); // ValueError: array_chunk(): Argument #2 ($length) must be greater than 0
```

Most of the internal functions now throw an Error exception if the validation of the parameters fails.

Just-In-Time compilation

PHP 8 introduces two JIT compilation engines. Tracing JIT, the most promising of the two, shows about 3 times better performance on synthetic benchmarks and 1.5–2 times improvement on some specific long-running applications. Typical application performance is on par with PHP 7.4.

Relative JIT contribution to PHP 8 performance



Type system and error handling improvements

- Stricter type checks for arithmetic/bitwise operators RFC
- Abstract trait method validation RFC
- Correct signatures of magic methods RFC
- Reclassified engine warnings RFC
- Fatal error for incompatible method signatures RFC
- The @ operator no longer silences fatal errors.
- Inheritance with private methods RFC
- Mixed type RFC
- Static return type RFC
- Types for internal functions Email thread
- Opaque objects instead of resources for Curl, Gd, Sockets, OpenSSL, XMLWriter, and XML extensions

Other syntax tweaks and improvements

- Allow a trailing comma in parameter lists RFC and closure use lists RFC
- Non-capturing catches RFC
- Variable Syntax Tweaks RFC
- Treat namespaced names as single token RFC
- Throw is now an expression RFC
- Allow ::class on objects RFC

New Classes, Interfaces, and Functions

- Weak Map class
- Stringable interface
- str_contains(), str_starts_with(), str_ends_with()
- fdiv()
- get_debug_type()
- get_resource_id()
- token_get_all() object implementation
- New DOM Traversal and Manipulation APIs

Better performance, better syntax, improved type safety.

Go update to PHP 8!

For source downloads of PHP 8 please visit the downloads page. Windows binaries can be found on the PHP for Windows site. The list of changes is recorded in the ChangeLog.

The migration guide is available in the PHP Manual. Please consult it for a detailed list of new features and backward-incompatible changes.