https://stitcher.io/blog/new-in-php-73

Scout APM helps PHP developers pinpoint N+1 queries, memory leaks & more so you can troubleshoot fast & get back to coding faster. [Start your free 14-day trial today](https://ter.li/yvaix5).

« [back](https://stitcher.io/) — written by [Brent](https://twitter.com/brendt_gd" \t "_blank) on December 06, 2018

[#](https://stitcher.io/blog/new-in-php-73#new-in-php-7.3)**New in PHP 7.3**

* Released on December 6, 2018
* [Trailing commas](https://stitcher.io/blog/new-in-php-73#trailing-commas-in-function-calls-rfc) are now allowed in function calls
* The [is\_countable](https://stitcher.io/blog/new-in-php-73" \l "is_countable-rfc) function
* Improvements to the [Heredoc syntax](https://stitcher.io/blog/new-in-php-73" \l "flexible-heredoc-syntax-rfc) makes it more flexible to use
* [array\_key\_first and array\_key\_last](https://stitcher.io/blog/new-in-php-73#array_key_first-and-array_key_last-rfc) are two new array helper functions

[#](https://stitcher.io/blog/new-in-php-73#is_countable-rfc)**is\_countable [rfc](https://wiki.php.net/rfc/is-countable" \t "_blank)**

PHP 7.2 added a warning when counting uncountable objects. The is\_countable function can help prevent this warning.

$count = is\_countable($variable) ? count($variable) : null;

[#](https://stitcher.io/blog/new-in-php-73#array_key_first-and-array_key_last-rfc)**array\_key\_first and array\_key\_last [rfc](https://wiki.php.net/rfc/array_key_first_last" \t "_blank)**

These two functions basically do what the name says.

$array = [

'a' => '…',

'b' => '…',

'c' => '…',

];

array\_key\_first($array); *// 'a'*

array\_key\_last($array); *// 'c'*

The original RFC also proposed array\_value\_first and array\_value\_last, but these were voted against by the majority of people.

Another idea for array\_first and array\_last was proposed which would return a tuple [$key => $value], but opinions were mixed. For now we only have two functions to get the first and last key of an array.

[#](https://stitcher.io/blog/new-in-php-73#flexible-heredoc-syntax-rfc)**Flexible Heredoc syntax [rfc](https://wiki.php.net/rfc/flexible_heredoc_nowdoc_syntaxes" \t "_blank)**

Heredoc can be a useful tool for larger strings, though they had an indentation quirk in the past.

*// Instead of this:*

$query = <<<SQL

SELECT \*

FROM `table`

WHERE `column` = true;

SQL;

*// You can do this:*

$query = <<<SQL

SELECT \*

FROM `table`

WHERE `column` = true;

SQL;

This is especially useful when you're using Heredoc in an already nested context.

The whitespaces in front of the closing marker will be ignored on all lines.

An important note: because of this change, some existing Heredocs might break, when they are using the same closing marker in their body.

$str = <<<FOO

abcdefg

FOO

FOO;

*// Parse error: Invalid body indentation level in PHP 7.3*

[#](https://stitcher.io/blog/new-in-php-73#trailing-commas-in-function-calls-rfc)**Trailing commas in function calls [rfc](https://wiki.php.net/rfc/trailing-comma-function-calls" \t "_blank)**

What was already possible with arrays, can now also be done with function calls. Note that it's not possible in function definitions!

$compacted = compact(

'posts',

'units',

);

[#](https://stitcher.io/blog/new-in-php-73#better-type-error-reporting)**Better type error reporting**

TypeErrors for integers and booleans used to print out their full name, it has been changed to int and bool, to match the type hints in the code.

Argument 1 passed to foo() must be of the type int/bool

In comparison to PHP 7.2:

Argument 1 passed to foo() must be of the type

integer/boolean

[#](https://stitcher.io/blog/new-in-php-73#json-errors-can-be-thrown-rfc)**JSON errors can be thrown [rfc](https://wiki.php.net/rfc/json_throw_on_error" \t "_blank)**

Previously, JSON parse errors were a hassle to debug. The JSON functions now accept an extra option to make them throw an exception on parsing errors. This change obviously adds a new exception: JsonException.

json\_encode($data, JSON\_THROW\_ON\_ERROR);

json\_decode("invalid json", null, 512, JSON\_THROW\_ON\_ERROR);

*// Throws JsonException*

While this feature is only available with the newly added option, there's a chance it'll be the default behaviour in a future version.

[#](https://stitcher.io/blog/new-in-php-73#list-reference-assignment-rfc)**list reference assignment [rfc](https://wiki.php.net/rfc/list_reference_assignment" \t "_blank)**

The list() and its shorthand [] syntax now support references.

$array = [1, 2];

list($a, &$b) = $array;

$b = 3;

*// $array = [1, 3];*

[#](https://stitcher.io/blog/new-in-php-73#undefined-variables-in-compact-rfc)**Undefined variables in compact [rfc](https://wiki.php.net/rfc/compact" \t "_blank)**

Undefined variables passed to compact will be reported with a notice, they were previously ignored.

$a = 'foo';

compact('a', 'b');

*// Notice: compact(): Undefined variable: b*

[#](https://stitcher.io/blog/new-in-php-73#case-insensitive-constants-rfc)**Case-insensitive constants [rfc](https://wiki.php.net/rfc/case_insensitive_constant_deprecation" \t "_blank)**

There were a few edge cases were case-insensitive constants were allowed. These have been deprecated.

[#](https://stitcher.io/blog/new-in-php-73#same-site-cookie-rfc)**Same site cookie [rfc](https://wiki.php.net/rfc/same-site-cookie" \t "_blank)**

This change not only adds a new parameter, it also changes the way the setcookie, setrawcookie and session\_set\_cookie\_params functions work in a non-breaking manner.

Instead of one more parameters added to already huge functions, they now support an array of options, whilst still being backwards compatible. An example:

bool setcookie(

string $name

[, string $value = ""

[, int $expire = 0

[, string $path = ""

[, string $domain = ""

[, bool $secure = false

[, bool $httponly = false ]]]]]]

)

bool setcookie (

string $name

[, string $value = ""

[, int $expire = 0

[, array $options ]]]

)

// Both ways work.

[#](https://stitcher.io/blog/new-in-php-73#pcre2-migration-rfc)**PCRE2 migration [rfc](https://wiki.php.net/rfc/pcre2-migration" \t "_blank)**

PCRE — short for "Perl Compatible Regular Expressions" — has been updated to v2.

The migration had a focus on maximum backwards compatibility, though there are a few breaking changes. Be sure to read the [RFC](https://wiki.php.net/rfc/pcre2-migration) to know about them.

[#](https://stitcher.io/blog/new-in-php-73#string-search-functions-readme)**String search functions [readme](https://github.com/php/php-src/blob/43329e85e682bed4919bb37c15acb8fb3e63175f/UPGRADING" \l "L327-L339" \t "_blank)**

You can no longer pass a non-string needle to string search functions. These are the affected functions:

strpos()

strrpos()

stripos()

strripos()

strstr()

strchr()

strrchr()

stristr()

[#](https://stitcher.io/blog/new-in-php-73#mbstring-updates-readme)**MBString updates [readme](https://github.com/php/php-src/blob/php-7.3.0RC6/UPGRADING" \l "L186-L232" \t "_blank)**

MBString is PHP's way of [handling complex strings](http://php.net/manual/en/intro.mbstring.php" \t "_blank). This module has received some updates in this version of PHP. You can read about it [here](https://github.com/php/php-src/blob/php-7.3.0RC6/UPGRADING#L186-L232).

[#](https://stitcher.io/blog/new-in-php-73#several-deprecations-rfc)**Several deprecations [rfc](https://wiki.php.net/rfc/deprecations_php_7_3" \t "_blank)**

Several small things have been deprecated, there's a possibility errors can show up in your code because of this.

* Undocumented mbstring function aliases
* String search functions with integer needle
* fgetss() function and string.strip\_tags filter
* Defining a free-standing assert() function
* FILTER\_FLAG\_SCHEME\_REQUIRED and FILTER\_FLAG\_HOST\_REQUIRED flags
* pdo\_odbc.db2\_instance\_name php.ini directive

Please refer to the [RFC](https://wiki.php.net/rfc/deprecations_php_7_3) for a full explanation of each deprecation.

👍

**Footnotes**

[Upgrade to PHP 7.3 on a Mac](https://stitcher.io/blog/php-73-upgrade-mac)

[What's new in PHP 7.4](https://stitcher.io/blog/new-in-php-74)

[What's new in PHP 8](https://stitcher.io/blog/new-in-php-8)

[PHP in 2020](https://stitcher.io/blog/php-in-2020)

[**Next up:  What's new in PHP 7.4**](https://stitcher.io/blog/new-in-php-74)

Follow me: [Twitter](https://twitter.com/brendt_gd" \t "_blank) — [RSS](https://stitcher.io/rss) — [Newsletter](https://stitcher.io/newsletter/subscribe" \t "_blank) — [Patreon](https://www.patreon.com/brendt" \t "_blank) — [GitHub](https://github.com/brendt" \t "_blank)

© 2021 stitcher.io — [Cookies & Privacy](https://stitcher.io/cookies" \t "_blank)