**Match or switch?**

PHP 8 introduces the new match expression. A powerful feature that will often be the better choice to using switch. So what exactly are the differences?

Let's start by comparing the two. Here's a classic switch example:

switch ($statusCode) {

case 200:

case 300:

$message = null;

break;

case 400:

$message = 'not found';

break;

case 500:

$message = 'server error';

break;

default:

$message = 'unknown status code';

break;

}

Here's its match equivalent:

$message = match ($statusCode) {

200, 300 => null,

400 => 'not found',

500 => 'server error',

default => 'unknown status code',

};

First of all, the match expression is significantly shorter:

* it doesn't require a break statement
* it can combine different arms into one using a comma
* it returns a value, so you only have to assign value once

That's already quite a lot, but there's even more to it!

[#](https://stitcher.io/blog/php-8-match-or-switch#no-type-coercion)**No type coercion**

match will do strict type checks instead of loose ones. It's like using === instead of ==.

$statusCode = '200';

$message = match ($statusCode) {

200 => null,

default => 'unknown status code',

};

*// $message = 'unknown status code'*

Bottom of Form

[#](https://stitcher.io/blog/php-8-match-or-switch#unknown-values-cause-errors)**Unknown values cause errors**

If you forget to check for a value, and when there's no default arm specified, PHP will throw an UnhandledMatchError exception. Again more strictness, but it will prevent subtle bugs from going unnoticed.

$statusCode = 400;

$message = match ($statusCode) {

200 => 'perfect',

};

*// UnhandledMatchError*

[#](https://stitcher.io/blog/php-8-match-or-switch#only-single-line-expressions,-for-now)**Only single-line expressions, for now**

Just like [short closures](https://stitcher.io/blog/short-closures-in-php), you can only write one expression. Expression blocks will probably get added at one point, but it's still not clear when exactly.

[#](https://stitcher.io/blog/php-8-match-or-switch#combining-conditions)**Combining conditions**

You already noticed the lack of break? This also means match doesn't allow for fallthrough conditions, like the two combined case lines in the first switch example. On the other hand though, you can combine conditions on the same line, separated by commas.

So you have the same functionality as switch in this regards, but with less writing, and less ways to screw up. Win-win!

$message = match ($statusCode) {

200, 300, 301, 302 => 'combined expressions',

};

[#](https://stitcher.io/blog/php-8-match-or-switch#complex-conditions-and-performance)**Complex conditions and performance**

During the RFC discussion, some people suggested the following pattern as an argument against adding the match expression:

$message = [

$this->matchesRegex($line) => 'match A',

$this->matchesOtherRegex($line) => 'match B',

][$line] ?? 'no match';

There's one big caveat though: this technique will execute all regex functions first, decreasing performance. A good argument for match.

[#](https://stitcher.io/blog/php-8-match-or-switch#throwing-exceptions)**Throwing exceptions**

Finally, because of [throw expressions in PHP 8](https://stitcher.io/blog/new-in-php-8#throw-expression-rfc), it's also possible to directly throw from an arm, if you'd like to.

$message = match ($statusCode) {

200 => null,

500 => throw new ServerError(),

default => 'unknown status code',

};

[#](https://stitcher.io/blog/php-8-match-or-switch#pattern-matching)**Pattern matching**

Ok, there's one more thing: pattern matching. It's a technique used in other programming languages, to allow complexer matching than simple values. Think of it as regex, but for variables instead of text.

Pattern matching isn't supported right now, because it's quite a complex feature, but Ilija Tovilo, the [RFC author](https://wiki.php.net/rfc/match_expression_v2) did mention it as a possible future feature. Something to look out for!

[#](https://stitcher.io/blog/php-8-match-or-switch#so,-switch-or-match?)**So, switch or match?**

If I'd need to summarize the match expression in one sentence, I'd say it's the stricter and more modern version of it's little switch brother.

There are some cases — *see what I did there?* — where switch will offer more flexibility, especially with multiline code blocks. However, the strictness of the match operator is appealing, and the perspective of pattern matching would be a game-changer for PHP.

I admit I never wrote a switch statement in the past years because of its many quirks; quirks that match actually solve. So while it's not perfect yet, there are use cases that I can think of, where match would be a good… match.