

DPI

De Programmatica *Ipsium*

DE PROGRAMMATICA IPSUM

Issue 083: PHP

August 4th, 2025

Table of Contents

Issue 083: PHP	5
The Toyota Corolla Of Programming	9
Kévin Dunglas	17
Josh Lockhart & Phil Sturgeon	23

Issue 083: PHP



August 4th, 2025

Welcome to the 83rd issue of *De Programmatica Ipsum*, about *PHP*.

In this edition:

- We solemnly declare that PHP deserves a second chance¹.
- In the Library section², we review “PHP: The Right Way” by Josh Lockhart & Phil Sturgeon³.
- In our Vidéothèque section⁴, we watch Kévin Dunglas⁵ tell us everything we need to know about FrankenPHP.

We hereby announce the discontinuation of our email newsletter, as the service we have been using until now is unable to provide a solution against the proliferation

of spam accounts. We apologize for the inconvenience, but this is why we cannot have nice things.

Download this issue in DRM-free PDF⁶ or EPUB⁷ format, and read it on your preferred device. You can also subscribe to our RSS feed⁸, featuring the full content of our articles.

We would like to thank our patrons who generously contribute every month (or have contributed in the past) to our work and help us run this magazine. Thank you so much! In alphabetical order: Adam Guest, Adrian Tineo Cabello, Benjamin Sheldon, Christopher Nascone, Colin Powell, Franz Lucien Moersdorf, Guillermo Ramos Álvarez, Jean-Paul de Vooght, Dr. Juande Santander-Vela, Patryk Matuszewski, Paul Hudson, Quico Moya, Roger Turner, Szymon Licau, and countless more leaving anonymous tips every month.

Enjoy this issue! Please share the articles on social media, or contribute⁹ if you would like to support our work with a donation via Liberapay¹⁰.

Cover photo by Ben Griffiths¹¹ on Unsplash¹².

REFERENCES

- ¹ <https://deprogrammaticaipsum.com/the-toyota-corolla-of-programming/>
- ² <https://deprogrammaticaipsum.com/category/library/>
- ³ <https://deprogrammaticaipsum.com/josh-lockhart-phil-sturgeon/>
- ⁴ <https://deprogrammaticaipsum.com/category/videotheque/>
- ⁵ <https://deprogrammaticaipsum.com/kévin-dunglas/>
- ⁶ <https://deprogrammaticaipsum.com/pdf/issue-083-php.pdf>
- ⁷ <https://deprogrammaticaipsum.com/epub/issue-083-php.epub>
- ⁸ <https://deprogrammaticaipsum.com/index.xml>
- ⁹ <https://deprogrammaticaipsum.com/contribute/>
- ¹⁰ <https://liberapay.com/akosma/donate>
- ¹¹ https://unsplash.com/@benofthenorth?utm_content=creditCopyText&utm_medium=referral&utm_source=unsplash
- ¹² https://unsplash.com/photos/blue-elephant-plush-toy-on-black-laptop-computer-2Rd-hwT2xQ0?utm_content=creditCopyText&utm_medium=referral&utm_source=unsplash

The Toyota Corolla Of Programming



By Adrian Kosmaczewski

In 1995, an otherwise unknown software developer released the first version of a new scripting language whose explicit aim was to make applications for this new platform called “The World Wide Web”. After starting as a small project, and thanks to the crazy dot-com years, it grew dramatically to become one of the most widely used programming languages of all time. After some stumbling first steps, it eventually got some sort of standardization in 1997, even reluctantly including some OOP features to please community and pundits alike.

However, no matter how hard it tried, this language and its users were mocked for decades by so-called “serious” programmers, who derided its “WTF”-level syntax, the quirks of its runtime model, its ever-increasing amount of security issues, or the gazillion frameworks that sprang around it. Despite the backlash, this language and its community prevailed, eventually getting a huge second act, including some explicit support from Big Tech themselves. Nowadays, as the language reaches the glorious age of 30, a new project drives its future evolution thanks to the strengths of the Go programming language.

Interestingly enough, the description spread across the two paragraphs above fits not just one but two programming languages: on one side, PHP, heavily inspired by Perl and released¹ by Rasmus Lerdorf in June 1995 with the name “Personal Home Page Tools”; and on the other hand JavaScript, designed by Brendan Eich and released² December of that same year by Netscape.

But the parallels between both languages do not stop there. In 1997, they got both some level of standardization, thanks to PHP/FI 2³ and ECMA-262⁴. During the 2010s, each language got major support from Facebook on one side, and Google and Microsoft on the other side. And in 2025, they both got a major revamp effort based on the Go programming language⁵: a new PHP runtime called FrankenPHP⁶, and the recently announced TypeScript compiler⁷.

PHP and JavaScript represent two faces of the same coin: web programming, both server-side and client-side. The growth of the World Wide Web⁸ transformed them into major players, despite their (let us be honest) quite jarring initial design flaws, their slow committee-based evolution, and the seemingly endless series of security flaws that plagued their respective ecosystems.

An oft-quoted smirk⁹ by Bjarne Stroustrup, the creator of C++, states that

There are only two kinds of languages: the ones people complain about and the ones nobody uses.

Two years ago we published an issue about BASIC¹⁰, the programming language that all developers love to hate but which single-handedly defined a whole era in our industry. It is only fair that we dedicate some words to PHP, the one everyone

also complains about, the one everyone laughs¹¹ about, yet apparently¹² powers between 70 and 80 percent of the world's websites; needless to say, an impressive number, no matter how hard you look at it, and no matter how much “serious” programmers laugh about it.

Throughout history, programming languages have received interesting, if revealing, epithets. C is said to be “portable assembly”. Java¹³ is good for “write once, debug anywhere”. Python¹⁴ is usually referred to as “executable pseudocode”. JavaScript¹⁵ “was created in 10 days, and it shows”. Perl is the “duct tape of the Internet”¹⁶.

And, well, PHP is either a “fractal of bad design” (seriously, people?) or an acronym meaning “Pretty Horrific Programming”. Ouch.

Here is how I see things: PHP is the *lingua franca* of affordable cloud web hosting options; or, in other terms, the Toyota Corolla¹⁷ of programming languages: boring, solid, easy, and affordable. You can find, almost anywhere in the world, an affordable web hosting option offering the saint quadrinity of LAMP: Linux, Apache, MySQL, and PHP; an operating system, a web server, a database server, and a scripting language, in an inexpensive package, enabling the masses to go further. Paraphrasing George Clooney, what else?

These days, PHP features many traits of a modern programming language; let us enumerate some, beginning with its fully open-source¹⁸ nature. It has advanced OOP¹⁹ features like traits²⁰, property hooks²¹, namespaces²², attributes²³, and enums²⁴. It includes functional programming constructs, like closures²⁵ with capture lists and arrow functions²⁶, and a “pipe” operator coming next November²⁷ (rejoice, OCaml and F# developers!). PHP has associative arrays, and a rudimentary yet fast, useful, and growing type checking system (remember: `declare(strict_types=1);` is your friend) including never²⁸ and nullable types²⁹. It bundles a full library of algorithms ready to use, following the “batteries included” mantra, and if all else fails, there is a powerful and open-source package manager³⁰ with enough packages³¹ to make npm blush. There is excellent support for unit testing³². As scripting programming languages go, it is quite fast to compile and execute. It has its own ecosystem of conferences, a mascot³³, and even a powerful IDE³⁴ made by none other than JetBrains.

But, alas, it does feature a goto operator³⁵. Oh, là, là! What would Dijkstra say! Not to mention those pesky variables prefixed with a stupid dollar sign, and a cringeworthy foreach statement. The ignominy!

Most so-called “serious” programmers would be wise to step down from their ivory tower and take a look at PHP in 2025 before an LLM kicks them out of their job. The language has seen major, steady, and consistent revamps for the past decade, including a stable release rhythm once per year, every November, plus a dedicated team that has been consistently³⁶ removing security vulnerabilities, and removing obsolete APIs with newer and safer ones.

I know they should take another look at PHP, because I myself had to step down from my own ivory tower and do that. It is only fair to state my *mea culpa*: back in 2009 I participated in a (admittedly useless) community effort called “I Hate PHP” (of which the Internet Archive has duly kept a copy³⁷) where my name appeared prominently on the front page. I plead guilty, your honor.

In my defense, I will argue that those were the somewhat darker ages of PHP. Those were the times of Bruce Tate’s “Beyond Java”³⁸ and its long diatribes against PHP spread in the pages therein.

Remember the long-gone PHP Security Consortium³⁹? Are the days of Little Bobby Tables⁴⁰ over? Let us be honest; not really. You can still release code with SQL injection vulnerabilities if you want (hint: avoid the . operator as much as you can. For all things that are no database queries, string interpolation⁴¹ is your friend).

The Rt. Rvd. Pastor Manul Laphroaig⁴², in his sermon⁴³ to the Beloved Congregation of the First United Church of the Weird Machines, claimed that there is divinity in every programming language... including PHP:

If a language like PHP introduces so many people to pwnage, then that is its divinity. It provides a first step for children to learn how program execution goes astray, with control and data so easy to mangle.

Amen.

Nowadays, what I see now is a healthy, thriving community⁴⁴ around PHP. One that, with the exception from JetBrains, is not encumbered by the whims of a FAANG or any other “Big Tech” firm. And the PHP Foundation⁴⁵ is driving the evolution of its standards⁴⁶ towards the future, hopefully navigating that unstable space between money and people.

Take these numbers with a grain of salt, but if we look at language ratings, there is a lot of terrain to reclaim back: on TIOBE, even if PHP was named language of the year 2004⁴⁷, it presently appears at the 15th position⁴⁸ and going down: it used to be 3rd in 2010. In the IEEE ranking, it appears in the 13th position⁴⁹, and on the 7th position⁵⁰ at PYPL, which is quite a drop in the past 20 years. But hey! It appears 4th⁵¹ at the RedMonk ranking, and the graph⁵² shows that the fall from 2013 to now has not been *that* strong. Not all is lost!

Jokes aside, the most important developments in the history of PHP had to do with the engines used to power it. For decades, the Zend engine⁵³ served as the reference point for the evolution of the language; designed for the needs of the World Wide Web of 1999 (a world of LAMP stacks) and as good as it was for its time, it could not evolve gracefully into a world of DevOps, containers, and Kubernetes⁵⁴ orchestrators.

(Raise your hand if you have ever tried to put together a `Dockerfile` with Nginx, FPM⁵⁵, and Supervisor⁵⁶, to run some forgotten PHP 7.1 application. I feel your pain, my friend.)

Thankfully, we can move away from Zend now. Thanks to the unpredictable power of its community, there is this thing called FrankenPHP⁵⁷. This project has been recently adopted⁵⁸ by the PHP Foundation as an official runtime, and it ticks all the boxes that could propel PHP into another orbit.

FrankenPHP not only dramatically simplifies the creation of containers, but it also provides new execution models for PHP code, all while offering 100% compatibility with the massive existing codebase. We talk more about FrankenPHP in the Vidéotheque section⁵⁹ this month.

As nice as FrankenPHP is, I do not count on the end of the backlash against PHP anytime soon. The language will continue to suffer the stigma of its humble

beginnings. Rasmus Lerdorf⁶⁰ did not get an interview in “Masterminds of Programming”⁶¹, because PHP is the quintessential language built in a bazaar, as the fruit of an accidental ~~affair~~ design. Nor will Rasmus be invited to the next HOPL conference⁶², despite the unreasonable popularity of the language, nor will PHP be used for PhD dissertations (other than those related to security, that is). To add insult to injury, PHP is not even mentioned in the Tao of Programming⁶³ (OK, OK, it was published in 1987, I give you that).

But thankfully IEEE’s “Computer Magazine” did pay attention to PHP, and followed suit with an interview of Rasmus in both written⁶⁴ and video⁶⁵ formats, published in 2012.

At least that. In that interview, Rasmus states the core philosophy behind the bazaar:

I learned a bit along the way that, for this to grow, I had to give up control of PHP—I had to let other people have some control. (...) It’s not just them contributing to my project—it becomes our project, and that really changed the nature of PHP.

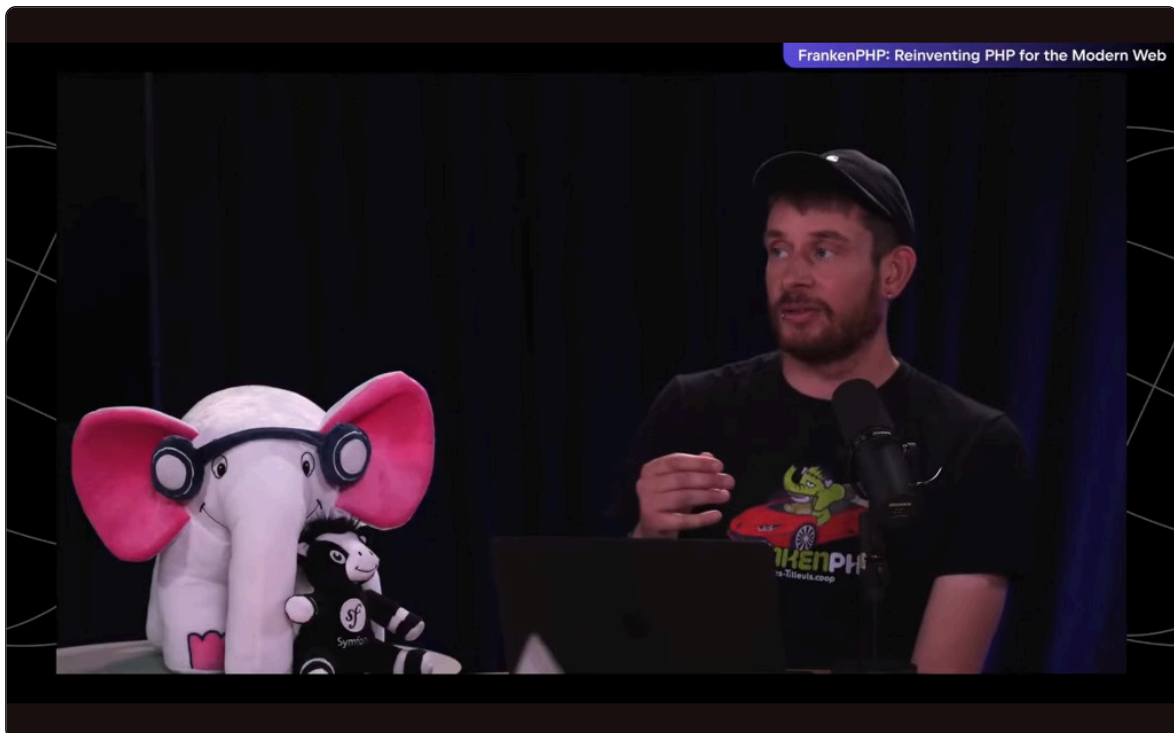
Cover photo by KOBU Agency⁶⁶ on Unsplash⁶⁷.

REFERENCES

- ¹ <https://groups.google.com/g/comp.infosystems.www.authoring.cgi/c/PyJ25gZ6z7A/m/M9FkTUVDfcwJ?lidx=2&wpid=363176&pli=1>
- ² <https://web.archive.org/web/20060111090514/http://wp.netscape.com/newsref/pr/newsrelease67.html>
- ³ <https://www.php.net/manual/phpfi2.php>
- ⁴ https://www.ecma-international.org/wp-content/uploads/ECMA-262_1st_edition_june_1997.pdf
- ⁵ <https://deprogrammaticaipsum.com/the-age-of-concurrency/>
- ⁶ <https://frankenphp.dev/>
- ⁷ <https://devblogs.microsoft.com/typescript/typescript-native-port/>
- ⁸ <https://deprogrammaticaipsum.com/from-hypertext-to-spas-to-hypertext/>
- ⁹ <https://www.stroustrup.com/quotes.html>
- ¹⁰ <https://deprogrammaticaipsum.com/programming-the-liberal-arts/>
- ¹¹ <https://php-ceo.medium.com/>
- ¹² <https://kinsta.com/php-market-share/>
- ¹³ <https://deprogrammaticaipsum.com/write-anywhere-run-once/>
- ¹⁴ <https://deprogrammaticaipsum.com/the-state-of-python-in-2021/>
- ¹⁵ <https://deprogrammaticaipsum.com/innovationscript/>
- ¹⁶ <https://www.wired.com/story/programmers-arent-humble-anymore-nobody-codes-in-perl/>
- ¹⁷ https://en.wikipedia.org/wiki/Toyota_Corolla
- ¹⁸ <https://github.com/php/php-src>
- ¹⁹ <https://deprogrammaticaipsum.com/the-hype-cycle-of-oop/>
- ²⁰ <https://www.php.net/manual/en/language.oop5.traits.php>
- ²¹ <https://www.php.net/manual/en/language.oop5.property-hooks.php>
- ²² <https://www.php.net/manual/en/language.namespaces.php>
- ²³ <https://www.php.net/manual/en/language.attributes.overview.php>
- ²⁴ <https://www.php.net/manual/en/language.types.enumerations.php>
- ²⁵ <https://www.php.net/manual/en/functions.anonymous.php>
- ²⁶ <https://www.php.net/manual/en/functions.arrow.php>
- ²⁷ <https://thephp.foundation/blog/2025/07/11/php-85-adds-pipe-operator/>
- ²⁸ <https://www.php.net/manual/en/language.types.never.php>
- ²⁹ <https://www.php.net/manual/en/language.types.declarations.php>
- ³⁰ <https://getcomposer.org/>
- ³¹ <https://packagist.org/>
- ³² <https://phpunit.de/index.html>
- ³³ <https://afieldguidetoelephpants.net/>
- ³⁴ <https://www.jetbrains.com/phpstorm/>
- ³⁵ <https://www.php.net/manual/en/control-structures.goto.php>
- ³⁶ <https://www.cvedetails.com/vendor/74/PHP.html>

- ³⁷ <https://web.archive.org/web/20071213013127/https://www.ihatephp.net/>
- ³⁸ <https://www.oreilly.com/library/view/beyond-java/0596100949/>
- ³⁹ <https://web.archive.org/web/20050210035857/http://phpsec.org/>
- ⁴⁰ <https://bobby-tables.com/>
- ⁴¹ <https://www.php.net/manual/en/language.types.string.php#language.types.string.parsing>
- ⁴² <https://deprogrammaticaipsum.com/pastor-manul-laphroaig/>
- ⁴³ <https://archive.org/details/Pocorgtfo01/page/n13/mode/2up>
- ⁴⁴ <https://deprogrammaticaipsum.com/the-tragedy-of-the-common-enemy/>
- ⁴⁵ <https://thephp.foundation/>
- ⁴⁶ <https://github.com/php-fig/fig-standards/>
- ⁴⁷ <https://web.archive.org/web/20050113041221/http://www.tiobe.com/tpci.htm>
- ⁴⁸ <https://www.tiobe.com/tiobe-index/php/>
- ⁴⁹ <https://spectrum.ieee.org/top-programming-languages-2024>
- ⁵⁰ <https://pypl.github.io/PYPL.html>
- ⁵¹ <https://redmonk.com/sogrady/2025/06/18/language-rankings-1-25/>
- ⁵² <https://redmonk.com/rstephens/files/2025/06/redmonk-language-rankings-jan-2025-1.png>
- ⁵³ <https://www.zend.com/>
- ⁵⁴ <https://deprogrammaticaipsum.com/antonomasia/>
- ⁵⁵ <https://www.php.net/manual/en/install.fpm.php>
- ⁵⁶ <https://supervisord.org/>
- ⁵⁷ <https://frankenphp.dev/>
- ⁵⁸ <https://thephp.foundation/blog/2025/06/08/php-30/>
- ⁵⁹ <https://deprogrammaticaipsum.com/kévin-dunglas/>
- ⁶⁰ https://en.wikipedia.org/wiki/Rasmus_Lerdorf
- ⁶¹ <https://www.oreilly.com/library/view/masterminds-of-programming/9780596801670/>
- ⁶² <https://deprogrammaticaipsum.com/jean-sammet/>
- ⁶³ <https://deprogrammaticaipsum.com/geoffrey-james/>
- ⁶⁴ <https://www.computer.org/csdl/magazine/co/2012/11/mco2012110006/13rUILLkze>
- ⁶⁵ <https://www.youtube.com/watch?v=YIGRXEzjE6c>
- ⁶⁶ https://unsplash.com/@kobuagency?utm_content=creditCopyText&utm_medium=referral&utm_source=unsplash
- ⁶⁷ https://unsplash.com/photos/hello-world-text-67L18R4tW_w?utm_content=creditCopyText&utm_medium=referral&utm_source=unsplash

Kévin Dunglas



By Adrian Kosmaczewski

PHP is the effective *lingua franca* of cheap web hosting. You know, that thing costing 5 bucks a month, where your uncle hosts a WordPress blog about fishing and his personal email. You know, that crappy service that comes with some gigabytes on a shared server, a nice LAMP stack on top, a shitty cPanel¹ and WHM combo to manage things... and SFTP access to upload pictures from Jamie and Larry's wedding in Chattanooga. Woop woop!

This is the kingdom of PHP. This is the world where PHP has been evolving into for the past 30 years. But there is a man called Kévin Dunglas (yes, with an accent mark on the *e*) who thinks that it is time for a change, and that PHP can effectively be more than that.

(Talk about having faith.)

Because, let us be honest here: there is a world in which PHP has never truly felt at home; and that is, containers and Kubernetes. Yes, good old Kubernetes, that thing that during the past 11 years² has effectively become the “state of the art” for companies to host applications in “the Cloud”³ (with a capital C, please).

And here is the thing: of all contemporary programming languages, PHP is among the few (if not the only one) that has not really shown a strong presence in that space. Of course, lots of very “professional” software developers are probably happy about that. But not Kévin.

This man is one of the founders of a small French software cooperative called Les Tilleuls⁴ with offices in Lille, Paris, Nantes, and Lyon. A cooperative, of all things. How could we not celebrate the fact that software engineers have decided to *finally* start organizing themselves and their work using the most democratic and progressive ways available in common law?

His team has decided to tackle the problem of “PHP in 2025”, which can be largely be understood as “PHP in Kubernetes”, and they succeeded in such a spectacular way, that even the PHP Foundation noticed... and ended up adopting⁵ their solution as the new standard for PHP.

This month’s Vidéotheque movie⁶ is, precisely, Kévin explaining the ins and outs of their solution for the future of PHP, called FrankenPHP⁷, at the last PHPverse 2025⁸ online conference, celebrating the 30 years of PHP, organized⁹ by JetBrains, the makers of the PHPStorm IDE.

We do not often dedicate a whole article to a single piece of technology in this magazine, as our aim is to give larger views about our craft. But we do believe that PHP deserves a better runtime environment today; not tomorrow, not in 10 years, but today (actually, it deserved it a decade ago, but better late than never).

FrankenPHP not only makes it easier for PHP to run in containers and Kubernetes (not that it was impossible, but it was certainly not straightforward), but it also enables much more than that. In essence, FrankenPHP is a reimplementation of the PHP language runtime and library using the Go¹⁰ programming language

(sounds familiar?¹¹) and unlocking, at the same time, new possibilities for PHP applications.

Not only that, but FrankenPHP is 100% compatible with other famous runtimes for PHP, such as the Zend engine, which means that applications are guaranteed to work with it without any kind of rewriting or adaptation.

In all justice, history teach us that this is not the first time someone has tried to come up with an alternative PHP runtime, or a way to interoperate / transpile / generate PHP code somehow; let us review some powerful ancestors, some of which you might not be aware of.

- The official Zend engine¹², in active development since 1999, has been the default runtime for PHP for over a quarter of a century now.
- There used to be PHP support in the Parrot Virtual Machine¹³.
- There was a short-lived thing called “Hippy VM”¹⁴ around 2012.
- How can we forget Facebook’s own “Hack”¹⁵ transpiler, announced¹⁶ in 2014?
- Somebody tried to run PHP in Java giving rise to Quercus¹⁷.
- And of course, somebody else tried to run PHP in .NET¹⁸ and then .NET from PHP¹⁹, because why not. There was even a transpiler to convert PHP into .NET CIL called Phalanger²⁰.
- Speaking about transpilers, one of the best known was HipHop²¹, to translate PHP code into C++, because why not.
- The Russian team behind the ВКонтакте social network developed KPHP²².
- For those stuck with old hosting providers, or legacy operating systems without access to the latest versions of PHP, Phabel²³ transpiled PHP 8.0 code into PHP code compatible with versions 5 or 7, security be damned.
- More transpilers: Phel²⁴ and Pharen²⁵ compile Lisp into PHP.
- Haxe²⁶ is a cross-platform language that also compiles into PHP.
- Oh, and let us not forget about IBM²⁷ and their support for PHP, past²⁸ and present²⁹.
- Honorable mentions: Kira³⁰, Tagua VM³¹, Zephir³², Phug³³, PHP7³⁴, and more!

So what does make FrankenPHP different? The fact that it single-handedly solved the most important problems that PHP developers faced, providing a faster and smaller runtime, and projecting the language into the present... and the future.

Of course, FrankenPHP is not the only one with such objectives: projects such as RoadRunner³⁶ and Open Swoole³⁷ are also throwing punches in this arena, and they all bring a promising future for PHP.

I am very enthusiastic about FrankenPHP, and even more so by the community spirit that drives it forward. You can feel the winds of change in the air of PHP. Laravel Octane³⁸ optimizes app startup when running under FrankenPHP; some talk about a renaissance³⁹ of the language; Maximiliano Firtman⁴⁰ doubles down on server-side rendering⁴¹ with PHP in his own training classes; and Fireship⁴² is happy to explain PHP in 100 seconds⁴³, or to build the same app with 10 different languages⁴⁴, including, you guessed, PHP. Yes, Prime, PHP 8.4 is good⁴⁵ and 8.5 will be even better.

PHP is the underdog, the Toyota Corolla, the poor cousin of programming languages; in this magazine we want to see it thrive and grow, and we cannot but cheer and support FrankenPHP, and hope that it will change the perception of the language in the near future. And we are happy to see that a cooperative, of all organizations, is the one driving behind the wheel.

Watch this month's Vidéothèque movie, "FrankenPHP: Reinventing PHP for the Modern Web" on YouTube⁴⁶, and then continue browsing videos from JetBrains' "PHP Annotated"⁴⁷, their YouTube channel dedicated to PHP.

Cover snapshot chosen by the author.

REFERENCES

- ¹ <https://www.cpanel.net/>
- ² <https://kubernetes.io/blog/2024/06/06/10-years-of-kubernetes/>
- ³ <https://deprogrammaticaipsum.com/five-computers/>
- ⁴ <https://les-tilleuls.coop/en>
- ⁵ <https://thephp.foundation/blog/2025/06/08/php-30/>
- ⁶ <https://www.youtube.com/watch?v=k-UwH91XnAo>
- ⁷ <https://frankenphp.dev/>
- ⁸ <https://www.youtube.com/watch?v=3b0ty1iZ8QM>
- ⁹ <https://lp.jetbrains.com/phpverse-2025/>
- ¹⁰ <https://deprogrammaticaipsum.com/rob-pike/>
- ¹¹ <https://devblogs.microsoft.com/typescript/typescript-native-port/>
- ¹² <https://www.zend.com/>
- ¹³ https://en.wikipedia.org/wiki/Parrot_virtual_machine
- ¹⁴ <https://morepypy.blogspot.com/2012/07/hello-everyone.html>
- ¹⁵ <https://hacklang.org/>
- ¹⁶ <https://engineering.fb.com/2014/11/11/developer-tools/announcing-the-hack-transpiler/>
- ¹⁷ <https://www.caucho.com/resin-3.1/doc/quercus.xtp>
- ¹⁸ <https://www.peachpie.io/>
- ¹⁹ <https://www.php.net/manual/en/class.dotnet.php>
- ²⁰ [https://en.wikipedia.org/wiki/Phalanger_\(compiler\)](https://en.wikipedia.org/wiki/Phalanger_(compiler))
- ²¹ https://en.wikipedia.org/wiki/HipHop_for_PHP
- ²² <https://vkcom.github.io/kphp/>
- ²³ <https://github.com/phabelio/phabel>
- ²⁴ <https://dev.to/chemaclass/phel-the-lisp-that-compiles-to-php-963>
- ²⁵ <https://web.archive.org/web/20101112031737/https://scriptor.github.com/pharen/>
- ²⁶ <https://haxe.org/>
- ²⁷ <https://deprogrammaticaipsum.com/think/>
- ²⁸ <https://www.redbooks.ibm.com/redbooks/pdfs/sg247218.pdf>
- ²⁹ <https://www.ibm.com/support/pages/php-ibm-i>
- ³⁰ <https://web.archive.org/web/20080701084723/https://www.mathgladiator.com/projects/kira/>
- ³¹ <https://github.com/tagua-vm/tagua-vm>
- ³² <https://zephir-lang.com/en>
- ³³ <https://phug-lang.com/>
- ³⁴ <https://ph7.symisc.net/>
- ³⁵ <https://web.archive.org/web/20240525084339/https://symfony.fi/entry/exotic-php-implementations-hippyvm-jphp-tagua-vm-peachpie.html>
- ³⁶ <https://roadrunner.dev/>
- ³⁷ <https://openswoole.com/>

³⁸ <https://laravel.com/docs/12.x/octane>

³⁹ <https://www.youtube.com/shorts/4O6yC65BgD8>

⁴⁰ <https://firt.dev/>

⁴¹ <https://www.youtube.com/shorts/Z0GsiVQRJJo>

⁴² <https://deprogrammaticaipsum.com/fireship/>

⁴³ https://www.youtube.com/watch?v=a7_WFUIFS94

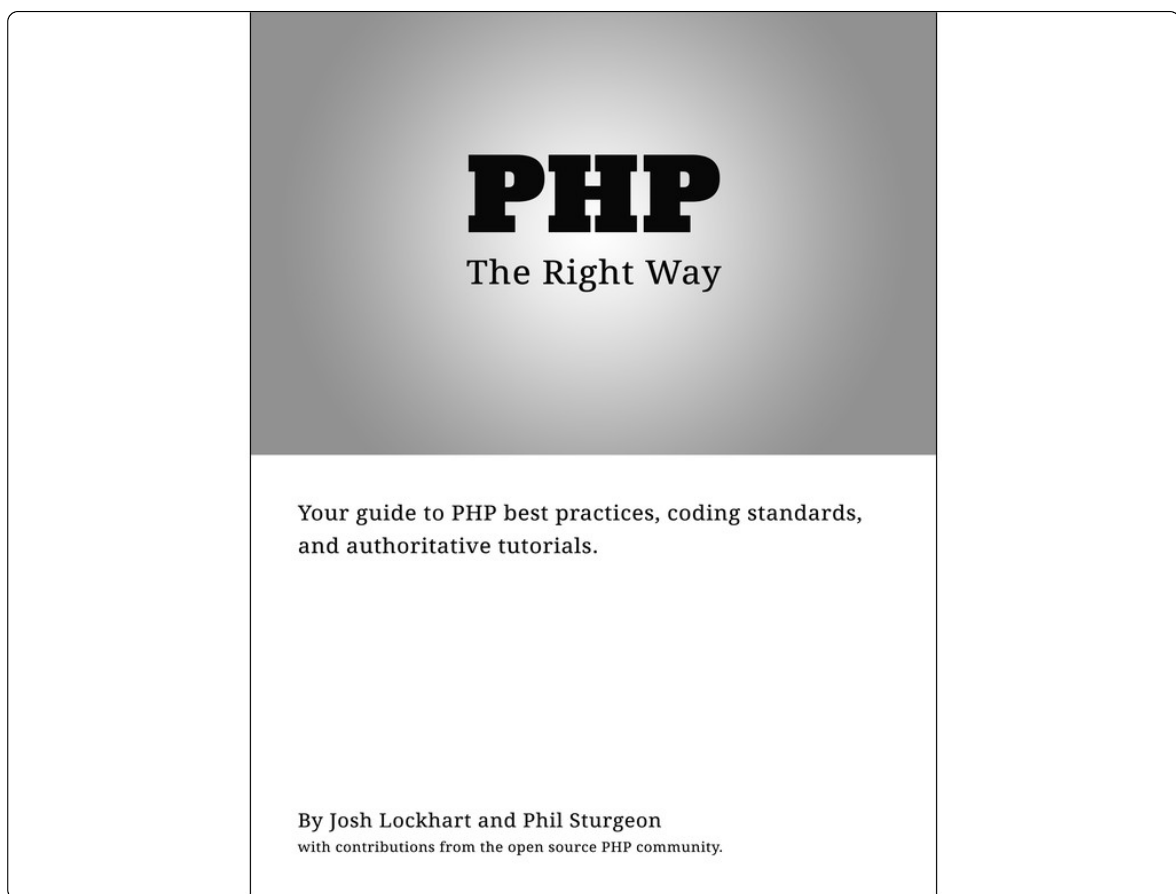
⁴⁴ <https://www.youtube.com/watch?v=FQPIEnKav48>

⁴⁵ <https://www.youtube.com/watch?v=f5nGmE92zNA>

⁴⁶ <https://www.youtube.com/watch?v=k-UwH91XnAo>

⁴⁷ <https://www.youtube.com/@phpannotated>

Josh Lockhart & Phil Sturgeon



By [Adrian Kosmaczewski](#)

The infinite flexibility of software is not without some major disadvantages. That is the main reason why we, software practitioners, crave so much any kind of information about “the best” or “the right” way to build, test, deploy, and maintain our systems. Yes, our craft is already complicated enough, and we are not even talking about the human complexities like office layouts, employment shenanigans, dress

codes, and whatnot. In this occasion we are going to talk about a resource that fights, with facts and examples, the battle of excellency in the world of PHP.

In the pages of this “Library” section we have often covered major books aiming to provide such guidance; for example Michael Feathers’ “Working Effectively with Legacy Code”¹; Scott Meyers’ “Effective C++”²; Amy Brown & Greg Wilson’s “Architecture of Open Source Applications”³; Jenifer Tidwell’s “Designing Interfaces”⁴; David Kadavy’s “Design for Hackers”⁵; Steve McConnell’s “Code Complete”⁶; Douglas Crockford’s “JavaScript: The Good Parts”⁷; and others. So many, actually, that we have added them all to a new category in this blog, called “Best Practices”⁸, which you can access at any time through the main menu.

Of course, this month’s entry, Josh Lockhart & Phil Sturgeon’s excellent “PHP: The Right Way” falls into this category. Solely mentioning the names of these two authors, however, I am acutely aware of the implicit, tacit, long list of contributors that have provided corrections, translations, reviews, and more to this work. Said list even includes some university professor like Kris Jordan⁹. But please understand that not enumerating them all here does not, in any way, diminish the relevance of their contributions. Let us just say that Josh and Phil are the instigators of this book, and not just the initial authors.

(And knowledgeable initial instigators they are, oh yes: Josh is the author of “Modern PHP”¹⁰, a book published by O’Reilly in 2015, and the creator of the Slim Framework¹¹, a favorite choice of mine in the world of PHP frameworks. Phil has worked as a consultant for various tech companies, but most importantly, he is behind the Protect Earth¹² project, a commendable reforestation effort we can only salute and support.)

As it stands, “PHP: The Right Way”, currently available on Leanpub¹³ in PDF or EPUB formats, and hosted on its own website¹⁴ for instant access, is the fruit of an open-source project hosted on GitHub¹⁵ with (at the time of this publication) more than 300 collaborators.

Now that is what I call a community.

Why read this? Well, to put it bluntly, this book aims to mark a milestone, leaving some classic books behind, like Rasmus Lerdorf’s own “PHP Pocket Reference”¹⁶

published by O'Reilly in 2000 (you can still read an excerpt from it on the Internet Archive¹⁷, by the way) or Paul Hudson's "PHP in a Nutshell"¹⁸ from 2005 (is it the same Paul Hudson that nowadays hacks with Swift¹⁹?) That "old PHP" is the default impression most developers have, particularly those who might have worked with the language and ecosystem decades ago, not realizing 23 years later that both had (thankfully) evolved for the better. Time for a wake-up call, if you will.

This book aims to point you towards what is considered the "State of the Art" and "Crème de la Crème" of tooling, practices, and patterns for your PHP code in 2025. From testing, to caching, to dependency management, to OOP, to internationalization and localization, and even to containerization, you will find enough information to help you along the way. This is a live document, updated every so often since its first version in 2013, translated to 20 languages other than English, and continuously evolving. Which, let us be honest, also explains why there is no printed version available—and rightfully so.

(The value you could derive from "PHP: The Right Way" assumes, of course, that you have not fallen prey of the latest fads around "vibe coding"²⁰, in which case, well, good luck with that. The whole premise of this magazine is very simple: that you care about your craft enough as to understand what is going on behind the scenes of the code that you put into production. That is all.)

PHP in 2025 is a fabulous beast with a thriving ecosystem, powering most of what you see on the Internet. This month's Library book, "PHP: The Right Way" is precisely the most appropriate choice to either discover *or* to get re-acquainted with PHP after a long hiatus.

We recommend coupling the lecture of this book with more resources. Books such as the 2023 "PHP Cookbook"²¹ by Eric A. Mann. Online resources, like "Learn modern PHP"²² by the prolific Daniel Opitz²³, or "Clean Code concepts adapted for PHP"²⁴ by Piotr Plenik. And modern best practices documents and manifestos, such as "The Twelve-Factor App"²⁵, "The Reactive Manifesto"²⁶, "Rootless Containers"²⁷, "Keep a Changelog"²⁸, "Oh Shit, Git!?"²⁹, "Reproducible Builds"³⁰, and the always useful "Bobby Tables: A guide to preventing SQL injection"³¹.

Oh, and if you excuse the shameless plug, and as mentioned previously, many of the books in the “Best Practices”³² category of this magazine could also be a welcome lecture in your road to betterment.

Cover photo adapted from the EPUB file.

REFERENCES

- ¹ <https://deprogrammaticaipsum.com/michael-feathers/>
- ² <https://deprogrammaticaipsum.com/scott-meyers/>
- ³ <https://deprogrammaticaipsum.com/amy-brown-greg-wilson/>
- ⁴ <https://deprogrammaticaipsum.com/jenifer-tidwell/>
- ⁵ <https://deprogrammaticaipsum.com/david-kadavy/>
- ⁶ <https://deprogrammaticaipsum.com/steve-mcconnell/>
- ⁷ <https://deprogrammaticaipsum.com/douglas-crockford/>
- ⁸ <https://deprogrammaticaipsum.com/category/best-practices/>
- ⁹ <https://krisjordan.com/>
- ¹⁰ <https://www.oreilly.com/library/view/modern-php/9781491905173/>
- ¹¹ <https://www.slimframework.com/>
- ¹² <https://www.protect.earth/>
- ¹³ <https://leanpub.com/phptherightway>
- ¹⁴ <https://phptherightway.com>
- ¹⁵ <https://github.com/codeguy/php-the-right-way>
- ¹⁶ <https://app.oreilly.com/pub/pr/879>
- ¹⁷ https://web.archive.org/web/20000815221550/http://www.oreilly.com/catalog/phppr/chapter/php_pkt.html
- ¹⁸ https://www.goodreads.com/book/show/43241.PHP_in_a_Nutshell
- ¹⁹ <https://www.hackingwithswift.com/>
- ²⁰ <https://deprogrammaticaipsum.com/the-allure-of-vibe-coding/>
- ²¹ <https://www.oreilly.com/library/view/php-cookbook/9781098121310/>
- ²² <https://odan.github.io/learn-php/>
- ²³ <https://odan.github.io/about.html>
- ²⁴ <https://github.com/piotrplenik/clean-code-php>
- ²⁵ <https://12factor.net/>
- ²⁶ <https://www.reactivemanifesto.org/>
- ²⁷ <https://rootlesscontaine.rs/>
- ²⁸ <https://keepachangelog.com/en/1.1.0/>
- ²⁹ <https://ohshitgit.com/>
- ³⁰ <https://reproducible-builds.org/>
- ³¹ <https://bobby-tables.com/>
- ³² <https://deprogrammaticaipsum.com/category/best-practices/>