WIKIPEDIA

PHP

PHP is a general-purpose scripting language geared towards web development.[8] It was originally created by Danish-Canadian programmer Rasmus Lerdorf in 1993 and released in 1995.[9][10] The PHP reference implementation is now produced by the PHP Group.[11] PHP was originally an abbreviation of ***Personal Home Page***,[12][13] but it now stands for the recursive acronym ***PHP: Hypertext Preprocessor***.[14]

PHP code is usually processed on a web server by a PHP interpreter implemented as a module, a daemon or a Common Gateway Interface (CGI) executable. On a web server, the result of the interpreted and executed PHP code—which may be any type of data, such as generated HTML or binary image data—would form the whole or part of an HTTP response. Various web template systems, web content management systems, and web frameworks exist that can be employed to orchestrate or facilitate the generation of that response. Additionally, PHP can be used for many programming tasks outside the web context, such as standalone graphical applications[15] and drone control.[16] PHP code can also be directly executed from the command line.

The standard PHP interpreter, powered by the Zend Engine, is free software released under the PHP License. PHP has been widely ported and can be deployed on most web servers on a variety of operating systems and platforms.[17]

The PHP language has evolved without a written formal specification or standard, with the original implementation acting as the *de facto* standard that other implementations aimed to follow.

PHP

Paradigm	Multi-paradigm: imperative, functional, object-oriented, procedural, reflective
Designed by	Rasmus Lerdorf
Developer	The PHP Development Team (https://php.net/credits/), Zend Technologies, PHP Foundation (https://thephp.foundation/)
First appeared	8 June 1995[1][2]
Stable release	8.4.3 / 17 January 2025[3]
Typing discipline	Dynamic, weak, gradual[4]
Implementation language	C (primarily; some components C++)
OS	Unix-like, Windows, macOS, IBM i, OpenVMS, IBM Z
License	PHP License (most of Zend engine under Zend Engine License) for PHP 4 and later versions (only; dual-licensed GNU General Public License version 2 or any later version and PHP License for PHP versions 3.0 or earlier.[5])
Filename extensions	`.php`,`.phar`,`.phtml`,`.pht`,`.phps`
Website	www.php.net (https://www.php.net)

Major implementations

Zend Engine, HHVM, PeachPie, Quercus, Parrot

Influenced by

Perl, C, C++, Java,[6] Tcl,[2] JavaScript[7]

Influenced

Hack, JSP, ASP, React JS

PHP Programming at Wikibooks

W3Techs reports that as of 27 October 2024 (about two years since PHP 7 was discontinued and 11 months after the PHP 8.3 release), PHP 7 is still used by 50.0% of PHP websites, which is outdated and known to be insecure.[18][19] In addition, the even more outdated (discontinued for 5+ years) and insecure PHP 5 is used by 13.2% and the no longer supported PHP 8.0 is also very popular, so the majority of PHP websites do not use supported versions.

History

Early history

Rasmus Lerdorf, creator of PHP; and Andi Gutmans and Zeev Suraski, creators of the Zend Engine

PHP development began in 1993[9] when Rasmus Lerdorf wrote several Common Gateway Interface (CGI) programs in C,[20][21] which he used to maintain his personal homepage. He extended them to work with web forms and to communicate with databases, and called this implementation "Personal Home Page/Forms Interpreter" or PHP/FI.

An example of the early PHP syntax:[22]

```
<!--include /text/header.html-->

<!--getenv HTTP_USER_AGENT-->
<!--if substr $exec_result Mozilla-->
  Hey, you are using Netscape!<p>
<!--endif-->

<!--sql database select * from table where user='$username'-->
<!--ifless $numentries 1-->
  Sorry, that record does not exist<p>
<!--endif exit-->
  Welcome <!--$user-->!<p>
  You have <!--$index:0--> credits left in your account.<p>

<!--include /text/footer.html-->
```

PHP/FI could be used to build simple, dynamic web applications. To accelerate bug reporting and improve the code, Lerdorf initially announced the release of PHP/FI as "Personal Home Page Tools (PHP Tools) version 1.0" on the Usenet discussion group *comp.infosystems.www.authoring.cgi* on 8 June 1995.[1][23] This release included basic functionality such as Perl-like variables, form handling, and the ability to embed HTML. By this point, the syntax had changed to resemble that of Perl, but was simpler, more limited, and less consistent.[12][11]

Early PHP was never intended to be a new programming language; rather, it grew organically, with Lerdorf noting in retrospect: "I don't know how to stop it [...] there was never any intent to write a programming language [...] I have absolutely no idea how to write a programming language [...] I

just kept adding the next logical step on the way."[24] A development team began to form and, after months of work and beta testing, officially released PHP/FI 2 in November 1997.[25]

The fact that PHP was not originally designed, but instead was developed organically has led to inconsistent naming of functions and inconsistent ordering of their parameters.[26] In some cases, the function names were chosen to match the lower-level libraries which PHP was "wrapping",[27] while in some very early versions of PHP the length of the function names was used internally as a hash function, so names were chosen to improve the distribution of hash values.[28]

PHP 3 and 4

Zeev Suraski and Andi Gutmans rewrote the parser in 1997 and formed the base of PHP 3, changing the language's name to the recursive acronym *PHP: Hypertext Preprocessor*.[11][29] Afterwards, public testing of PHP 3 began, and the official launch came in June 1998. Suraski and Gutmans then started a new rewrite of PHP's core, producing the Zend Engine in 1999.[30] They also founded Zend Technologies in Ramat Gan, Israel.[11]

This is an example of PHP code for the WordPress content management system.

On 22 May 2000, PHP 4, powered by the Zend Engine 1.0, was released.[11] By August 2008, this branch had reached version 4.4.9. PHP 4 is now no longer under development and nor are any security updates planned to be released.[31][32]

PHP 5

On 1 July 2004, PHP 5 was released, powered by the new Zend Engine II.[11] PHP 5 included new features such as improved support for object-oriented programming, the PHP Data Objects (PDO) extension (which defines a lightweight and consistent interface for accessing databases), and numerous performance enhancements.[33] In 2008, PHP 5 became the only stable version under development. Late static binding had been missing from previous versions of PHP, and was added in version 5.3.[34][35]

Many high-profile open-source projects ceased to support PHP 4 in new code from February 5, 2008, because of the GoPHP5 initiative,[36] provided by a consortium of PHP developers promoting the transition from PHP 4 to PHP 5.[37][38]

Over time, PHP interpreters became available on most existing 32-bit and 64-bit operating systems, either by building them from the PHP source code or by using pre-built binaries.[39] For PHP versions 5.3 and 5.4, the only available Microsoft Windows binary distributions were 32-bit IA-32 builds,[40][41] requiring Windows 32-bit compatibility mode while using Internet Information Services (IIS) on a 64-bit Windows platform. PHP version 5.5 made the 64-bit x86-64 builds available for Microsoft Windows.[42]

Official security support for PHP 5.6 ended on 31 December 2018.[43]

PHP 6 and Unicode

PHP received mixed reviews due to lacking native Unicode support at the core language level.[44][45] In 2005, a project headed by Andrei Zmievski was initiated to bring native Unicode support throughout PHP, by embedding the International Components for Unicode (ICU) library, and representing text strings as UTF-16 internally.[46] Since this would cause major changes both to the internals of the language and to user code, it was planned to release this as version 6.0 of the language, along with other major features then in development.[47]

However, a shortage of developers who understood the necessary changes, and performance problems arising from conversion to and from UTF-16, which is rarely used in a web context, led to delays in the project.[48] As a result, a PHP 5.3 release was created in 2009, with many non-Unicode features back-ported from PHP 6, notably namespaces. In March 2010, the project in its current form was officially abandoned, and a PHP 5.4 release was prepared to contain most remaining non-Unicode features from PHP 6, such as traits and closure re-binding.[49] Initial hopes were that a new plan would be formed for Unicode integration, but by 2014 none had been adopted.

PHP 7

During 2014 and 2015, a new major PHP version was developed, PHP 7. The numbering of this version involved some debate among internal developers.[50] While the PHP 6 Unicode experiments had never been released, several articles and book titles referenced the PHP 6 names, which might have caused confusion if a new release were to reuse the name.[51] After a vote, the name PHP 7 was chosen.[52]

The foundation of PHP 7 is a PHP branch that was originally dubbed *PHP next generation* (*phpng*). It was authored by Dmitry Stogov, Xinchen Hui and Nikita Popov,[53] and aimed to optimize PHP performance by refactoring the Zend Engine while retaining near-complete language compatibility.[54] By 14 July 2014, WordPress-based benchmarks, which served as the main benchmark suite for the phpng project, showed an almost 100% increase in performance. Changes from phpng make it easier to improve performance in future versions, as more compact data structures and other changes are seen as better suited for a successful migration to a just-in-time (JIT) compiler.[55] Because of the significant changes, the reworked Zend Engine was called *Zend Engine 3*, succeeding Zend Engine 2 used in PHP 5.[56]

Because of the major internal changes in phpng, it must receive a new major version number of PHP, rather than a minor PHP 5 release, according to PHP's release process.[57] Major versions of PHP are allowed to break backward-compatibility of code and therefore PHP 7 presented an opportunity for other improvements beyond phpng that require backward-compatibility breaks. In particular, it involved the following changes:

- Many fatal or recoverable-level legacy PHP error mechanisms were replaced with modern object-oriented exceptions.[58]
- The syntax for variable dereferencing was reworked to be internally more consistent and complete, allowing the use of the operators –>, `[]`, `()`,`{}`, and `::`, with arbitrary meaningful left-side expressions.[59]
- Support for legacy PHP 4-style constructor methods was deprecated.[60]
- The behavior of the `foreach` statement was changed to be more predictable.[61]
- Constructors for the few classes built-in to PHP which returned null upon failure were changed to throw an exception instead, for consistency.[62]

- Several unmaintained or deprecated server application programming interfaces (SAPIs) and extensions were removed from the PHP core, most notably the legacy `mysql` extension.[63]
- The behavior of the `list()` operator was changed to remove support for strings.[64]
- Support was removed for legacy ASP-style delimiters <% and %> and `<script language="php"> ... </script>`.[65]
- An oversight allowing a switch statement to have multiple `default` clauses was fixed.[66]
- Support for hexadecimal number support in some implicit conversions from strings to number types was removed.[67]
- The left-shift and right-shift operators were changed to behave more consistently across platforms.[68]
- Conversions between floating-point numbers and integers were changed (e.g. infinity changed to convert to zero) and implemented more consistently across platforms.[68][69]

PHP 7 also included new language features. Most notably, it introduced return type declarations for functions[70] which complement the existing parameter type declarations, and support for the scalar types (integer, float, string, and boolean) in parameter and return type declarations.[71]

PHP 8

PHP 8 was released on 26 November 2020, and is currently the second-most used PHP major version. PHP 8 is a major version and has breaking changes from previous versions.[72][73] New features and notable changes include:

Just-in-time compilation

Just-in-time compilation is supported in PHP 8.[74]

PHP 8's JIT compiler can provide substantial performance improvements for some use cases,[75][76] while (then PHP) developer Nikita Popov stated that the performance improvements for most websites will be less substantial than the upgrade from PHP 5 to PHP 7.[77] Substantial improvements are expected more for mathematical-type operations than for common web-development use cases.[77] Additionally, the JIT compiler provides the future potential to move some code from C to PHP, due to the performance improvements for some use cases.[78]

Addition of the match expression

PHP 8 introduced the `match` expression.[79] The match expression is conceptually similar to a `switch` statement and is more compact for some use cases.[80] Because `match` is an expression, its result can be assigned to a variable or returned from a function.[81]

Type changes and additions

PHP 8 introduced union types, a new `static` return type, and a new `mixed` type.[72]

"Attributes", often referred to as "annotations" in other programming languages, were added in PHP 8, which allow metadata to be added to classes.[72]

`throw` was changed from being a statement to being an expression.[82] This allows exceptions to be thrown in places that were not previously possible.[72]

Syntax changes and additions

PHP 8 includes changes to allow alternate, more concise, or more consistent syntaxes in a number of scenarios. For example, the nullsafe operator is similar to the null coalescing operator `??`, but used when calling methods.[83] The following code snippet will not throw an error if `getBirthday()` returns null:

```
$human_readable_date = $user->getBirthday()?->diffForHumans();
```

Constructor property promotion has been added as "syntactic sugar," allowing class properties to be set automatically when parameters are passed into a class constructor.[72] This reduces the amount of boilerplate code that must be written.[84]

Other minor changes include support for use of `::class` on objects, which serves as an alternative for the use of `get_class()`;[72] non-capturing catches in try-catch blocks; variable syntax tweaks to resolve inconsistencies; support for named arguments; and support for trailing commas in parameter lists, which adds consistency with support for trailing commas in other contexts, such as in arrays.[73]

Standard library changes and additions

- Weak maps were added in PHP 8. A `WeakMap` holds references to objects, but these references do not prevent such objects from being garbage collected.[85] This can provide performance improvements in scenarios where data is being cached; this is of particular relevance for object–relational mappings (ORM).[72]
- Various adjustments to interfaces, such as adding support for creating `DateTime` objects from interfaces, and the addition of a `Stringable` interface that can be used for type hinting.[72]
- Various new functions including `str_contains()`, `str_starts_with()`, and `str_ends_with()`;[86] `fdiv()`; `get_debug_type()`; and `get_resource_id()`[72]
- Object implementation of `token_get_all()`[72]

Additional changes

- Type annotations were also added into PHP's C source code itself to allow internal functions and methods to have "complete type information in reflection."[87]
- Inheritance with private methods[72]
- Abstract methods in traits improvements[72]

PHP 8.1

PHP 8.1 was released on November 25, 2021.[88] It added support for enumerations (also called "enums"), declaring properties as `readonly` (which prevents modification of the property after initialization), and array unpacking with string keys. The new never type can be used to indicate that a function does not return.[89]

PHP 8.2

PHP 8.2 was released on December 8, 2022.[90] New in this release are `readonly` classes (whose instance properties are implicitly readonly), disjunctive normal form (DNF) types, and the `random` extension, which provides a pseudorandom number generator with an object-oriented API,[91] Sensitive Parameter value redaction, and a ton of other features.

PHP 8.3

PHP 8.3 was released on November 23, 2023. This release introduced readonly array properties, allowing arrays to be declared as immutable after initialization. It also added support for class aliases for built-in PHP classes, new methods for random float generation in the Random extension, and enhanced PHP INI settings with fallback value support. Additionally, the new `stream_context_set_options` function provides improved API for stream manipulation, among other updates and deprecations.

PHP 8.4

PHP 8.4 was released on November 21, 2024.

Release history

Version	Release date	Supported until[92]	Notes
1.0	8 June 1995		Officially called "Personal Home Page Tools (PHP Tools)". This is the first use of the name "PHP".[11]
2.0	1 November 1997		Officially called "PHP/FI 2.0". This is the first release that could actually be characterised as PHP, being a standalone language with many features that have endured to the present day.
3.0	6 June 1998	20 October 2000[92]	Development moves from one person to multiple developers. Zeev Suraski and Andi Gutmans rewritten the base for this version.[11]
4.0	22 May 2000[93]	23 June 2001[92]	Added more advanced two-stage parse/execute tag-parsing system called the Zend engine.[94]
4.1	10 December 2001[95]	12 March 2002[92]	Introduced "superglobals" (`$_GET`, `$_POST`, `$_SESSION`, etc.)[94]
4.2	22 April 2002[96]	6 September 2002[92]	Disabled `register_globals` by default. Data received over the network is not inserted directly into the global namespace anymore, closing possible security holes in applications.[94]
4.3	27 December 2002[97]	31 March 2005[92]	Introduced the command-line interface (CLI), to supplement the CGI.[94][98]
4.4	11 July 2005[99]	7 August 2008[92]	Fixed a memory corruption bug, which required breaking binary compatibility with extensions compiled against PHP version 4.3.x.[100]
5.0	13 July 2004[101]	5 September 2005[92]	Zend Engine II with a new object model.[102]
5.1	24 November 2005[103]	24 August 2006[92]	Performance improvements with the introduction of compiler variables in re-engineered PHP Engine.[102] Added PHP Data Objects (PDO) as a consistent interface for accessing databases.[104]
5.2	2 November 2006[105]	6 January 2011[92]	Enabled the filter extension by default. Native JSON support.[102]
5.3	30 June 2009[106]	14 August 2014[92]	Namespace support; late static bindings, jump label (limited goto), anonymous functions, closures, PHP archives (phar), garbage collection for circular references, improved Windows support, sqlite3, mysqlnd as a replacement for libmysql as the underlying library for the extensions that work with MySQL, fileinfo as a replacement for mime_magic for better MIME support, the Internationalization extension, and deprecation of ereg extension.
5.4	1 March 2012[107]	3 September 2015[92]	Trait support, short array syntax support. Removed items: `register_globals`, `safe_mode`, `allow_call_time_pass_reference`, `session_register()`, `session_unregister()` and `session_is_registered()`. Built-in web server.[108] Several improvements to existing features, performance and reduced memory requirements.

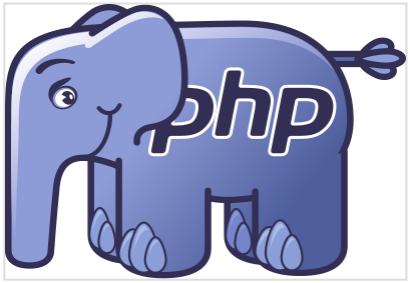
5.5	20 June 2013[109]	10 July 2016[110]	Support for generators, `finally` blocks for exceptions handling, OpCache (based on Zend Optimizer+) bundled in official distribution.[111]
5.6	28 August 2014[112]	31 December 2018[110]	Constant scalar expressions, variadic functions, argument unpacking, new exponentiation operator, extensions of the `use` statement for functions and constants, new `phpdbg` debugger as a SAPI module, and other smaller improvements.[113]
6.x	Not released	—	Abandoned version of PHP that planned to include native Unicode support.[114][115]
7.0	3 December 2015[116]	10 January 2019[57]	Zend Engine 3 (performance improvements[55] and 64-bit integer support on Windows[117]), uniform variable syntax,[59] AST-based compilation process,[118] added `Closure::call()`,[119] bitwise shift consistency across platforms,[120] `??` (null coalesce) operator,[121] Unicode code point escape syntax,[122] return type declarations,[70] scalar type (integer, float, string and boolean) declarations,[71] <=> "spaceship" three-way comparison operator,[123] generator delegation,[124] anonymous classes,[125] simpler and more consistently available CSPRNG API,[126] replacement of many remaining internal PHP "errors" with the more modern exceptions,[58] and shorthand syntax for importing multiple items from a namespace.[127]
7.1	1 December 2016	1 December 2019[110]	`iterable` type,[128] nullable types,[129] `void` return type,[130] class constant visibility modifiers,[131] short list syntax,[132] multi-catch[133]
7.2	30 November 2017	30 November 2020[110]	Object parameter and return type declaration,[134] libsodium extension,[135] abstract method overriding,[136] parameter type widening[137]
7.3	6 December 2018[138]	6 December 2021	Flexible Heredoc and Nowdoc syntax,[139] support for reference assignment and array deconstruction with `list()`,[140] PCRE2 support,[141] `hrtime` function[142]
7.4	28 November 2019[143]	28 November 2022	Typed properties 2.0,[144] preloading,[145] null-coalescing assignment operator,[146] improve `openssl_random_pseudo_bytes`,[147] weak references,[85] foreign function interface (FFI),[148] always available hash extension,[149] password hash registry,[150] multibyte string splitting,[151] reflection for references,[152] unbundle ext/wddx,[153] new custom object serialization mechanism[154]
8.0	26 November 2020[155]	26 November 2023	Just-In-Time (JIT) compilation,[74] arrays starting with a negative index,[156] stricter/saner language semantics (validation for abstract trait methods),[157] saner string to number comparisons,[158] saner numeric strings,[159] `TypeError` on invalid arithmetic/bitwise operators,[160] reclassification of various engine errors,[161] consistent type errors for internal functions,[162] fatal error for incompatible method signatures[163]), locale-independent float to string conversion,[164] variable syntax tweaks,[165] attributes,[166][167][168][169] named arguments,[170] match expression,[171] constructor property

			promotion,[172] union types,[173] `mixed` type,[174] static return type,[175] nullsafe operator,[83] non-capturing catches,[176] `throw` expression,[82] JSON extension is always available.[177]
8.1	25 November 2021[178]	31 December 2025	Explicit octal integer literal notation,[179] enumerations,[180] read-only properties,[181] first-class callable syntax,[182] `new` in initializers,[183] pure intersection types,[184] `never` return type,[185] `final` class constraints,[186] fibers[187]
8.2	8 December 2022[188]	31 December 2026	Readonly classes,[189] `null`, `false`, and `true` as stand-alone types,[190][191] locale-independent case conversion,[192] disjunctive normal form types,[193] constants in traits[194]
8.3	23 November 2023[195]	31 December 2027	Typed class constants,[196] dynamic class constant fetch,[197] `#[\Override]` attribute,[198] deep-cloning of read-only properties,[199] new `json_validate` function,[200] randomizer additions,[201] the command-line linter supports multiple files
8.4	21 November 2024[202]	31 December 2028	Property hooks, asymmetric visibility, an updated DOM API, performance improvements, bug fixes, and general cleanup.
Legend: Unsupported version	Old version, still maintained	**Latest version**	Future release		

Beginning on 28 June 2011, the PHP Development Team implemented a timeline for the release of new versions of PHP.[57] Under this system, at least one release should occur every month. Once per year, a minor release should occur which may include new features. Every minor release should at least be supported for two years with security and bug fixes, followed by at least one year of only security fixes, for a total of a three-year release process for every minor release. No new features, unless small and self-contained, are to be introduced into a minor release during the three-year release process.

Mascot

The mascot of the PHP project is the *elePHPant*, a blue elephant with the PHP logo on its side, designed by Vincent Pontier[203] in 1998.[204] "The (PHP) letters were forming the shape of an elephant if viewed in a sideways angle."[205] The elePHPant is sometimes differently coloured when in plush toy form.[206]



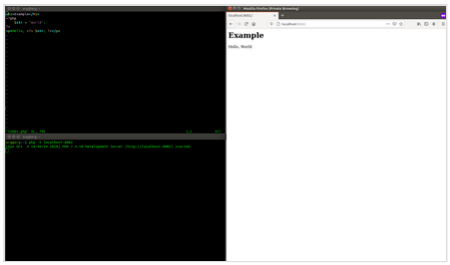
The elePHPant, PHP mascot

Many variations of this mascot have been made over the years. Only the elePHPants based on the original design by Vincent Pontier are considered official by the community.[207] These are collectable and some of them are extremely rare.[208]

Syntax

The following "Hello, World!" program is written in PHP code embedded in an HTML document:

```
<!DOCTYPE html>
<html>
    <head>
        <title>PHP "Hello, World!" program</title>
    </head>
    <body>
        <p><?= 'Hello, World!' ?></p>
    </body>
</html>
```



A "Hello, World" application in PHP 7.4 running on its built-in development server

However, as no requirement exists for PHP code to be embedded in HTML, the simplest version of *Hello, World!* may be written like this, with the closing tag `?>` omitted as preferred in files containing pure PHP code.[209]

```
<?php echo 'Hello, World!';
```

The PHP interpreter only executes PHP code within its delimiters. Anything outside of its delimiters is not processed by PHP, although the non-PHP text is still subject to control structures described in PHP code. The most common delimiters are `<?php` to open and `?>` to close PHP sections. The shortened form `<?` also exists. This short delimiter makes script files less portable since support for them can be disabled in the local PHP configuration and it is therefore discouraged.[210][211] Conversely, there is no recommendation against the echo short tag `<?=`.[212] Prior to PHP 5.4.0, this short syntax for `echo` only works with the `short_open_tag` configuration setting enabled, while for PHP 5.4.0 and later it is always available.[213][214][210] The purpose of all these delimiters is to separate PHP code from non-PHP content, such as JavaScript code or HTML markup.[215] So the shortest "Hello, World!" program written in PHP is:

```
<?='Hello, World!';
```

The first form of delimiters, `<?php` and `?>`, in XHTML and other XML documents, creates correctly formed XML processing instructions.[216] This means that the resulting mixture of PHP code and other markups in the server-side file is itself well-formed XML.

Variables are prefixed with a dollar symbol, and a type does not need to be specified in advance. PHP 5 introduced *type declarations* that allow functions to force their parameters to be objects of a specific class, arrays, interfaces or callback functions. However, before PHP 7, type declarations could not be used with scalar types such as integers or strings.[71]

Below is an example of how PHP variables are declared and initialized.

```
<?php
    $name = 'John';  // variable of string type being declared and initialized
    $age = 18;       // variable of integer type being declared and initialized
    $height = 5.3;   // variable of double type being declared and initialized
    echo $name . ' is ' . $height . "m tall\n"; // concatenating variables and strings
    echo "$name is $age years old."; // interpolating variables to string
?>
```

Unlike function and class names, variable names are case-sensitive. Both double-quoted ("") and heredoc strings provide the ability to interpolate a variable's value into the string.[217] PHP treats newlines as whitespace in the manner of a free-form language, and statements are terminated by a

semicolon.[218] PHP has three types of comment syntax: `/* */` marks block and inline comments; `//` or `#` are used for one-line comments.[219] The `echo` statement is one of several facilities PHP provides to output text.

In terms of keywords and language syntax, PHP is similar to C-style syntax. **`if`** conditions, **`for`** and **`while`** loops and function returns are similar in syntax to languages such as C, C++, C#, Java and Perl.

Data types

PHP is loosely typed. It stores integers in a platform-dependent range, either as a 32, 64 or 128-bit signed integer equivalent to the C-language long type. Unsigned integers are converted to signed values in certain situations, which is different behaviour to many other programming languages.[220] Integer variables can be assigned using decimal (positive and negative), octal, hexadecimal, and binary notations.

Floating-point numbers are also stored in a platform-specific range. They can be specified using floating-point notation, or two forms of scientific notation.[221] PHP has a native Boolean type that is similar to the native Boolean types in Java and C++. Using the Boolean type conversion rules, non-zero values are interpreted as true and zero as false, as in Perl and C++.[221]

The null data type represents a variable that has no value; `NULL` is the only allowed value for this data type.[221]

Variables of the "resource" type represent references to resources from external sources. These are typically created by functions from a particular extension, and can only be processed by functions from the same extension; examples include file, image, and database resources.[221]

Arrays can contain elements of any type that PHP can handle, including resources, objects, and even other arrays. Order is preserved in lists of values and in hashes with both keys and values, and the two can be intermingled.[221] PHP also supports strings, which can be used with single quotes, double quotes, nowdoc or heredoc syntax.[222]

The **Standard PHP Library** (SPL) attempts to solve standard problems and implements efficient data access interfaces and classes.[223]

Functions

PHP defines a large array of functions in the core language and many are also available in various extensions; these functions are well documented online PHP documentation (https://www.php.net/docs.php).[224] However, the built-in library has a wide variety of naming conventions and associated inconsistencies, as described under history above.

Custom functions may be defined by the developer:

```
function myAge(int $birthYear): string
{
    // calculate the age by subtracting the birth year from the current year.
    $yearsOld = date('Y') - $birthYear;

    // return the age in a descriptive string.
    return $yearsOld . ($yearsOld == 1 ? ' year' : ' years');
}
```

```
echo 'I am currently ' . myAge(1995) . ' old.';
```

As of 2025, the output of the above sample program is "I am currently 30 years old."

In lieu of function pointers, functions in PHP can be referenced by a string containing their name. In this manner, normal PHP functions can be used, for example, as callbacks or within function tables.[225] User-defined functions may be created at any time without being prototyped.[224][225] Functions may be defined inside code blocks, permitting a run-time decision as to whether or not a function should be defined. There is a `function_exists` function that determines whether a function with a given name has already been defined. Function calls must use parentheses, with the exception of zero-argument class constructor functions called with the PHP operator `new`, in which case parentheses are optional.

Since PHP 4.0.1 `create_function()`, a thin wrapper around `eval()`, allowed normal PHP functions to be created during program execution; it was deprecated in PHP 7.2 and removed in PHP 8.0[226] in favor of syntax for anonymous functions or "closures"[227] that can capture variables from the surrounding scope, which was added in PHP 5.3. Shorthand arrow syntax was added in PHP 7.4:[228]

```
function getAdder($x) {
    return fn($y) => $x + $y;
}

$adder = getAdder(8);
echo $adder(2);  // prints "10"
```

In the example above, `getAdder()` function creates a closure using passed argument `$x`, which takes an additional argument `$y`, and returns the created closure to the caller. Such a function is a first-class object, meaning that it can be stored in a variable, passed as a parameter to other functions, etc.[229]

Unusually for a dynamically typed language, PHP supports type declarations on function parameters, which are enforced at runtime. This has been supported for classes and interfaces since PHP 5.0, for arrays since PHP 5.1, for "callables" since PHP 5.4, and scalar (integer, float, string and boolean) types since PHP 7.0.[71] PHP 7.0 also has type declarations for function return types, expressed by placing the type name after the list of parameters, preceded by a colon.[70] For example, the `getAdder` function from the earlier example could be annotated with types like so in PHP 7:

```
function getAdder(int $x): Closure
{
    return fn(int $y): int => $x + $y;
}

$adder = getAdder(8);
echo $adder(2); // prints "10"
echo $adder(null); // throws an exception because an incorrect type was passed
$adder = getAdder([]); // would also throw an exception
```

By default, scalar type declarations follow weak typing principles. So, for example, if a parameter's type is `int`, PHP would allow not only integers, but also convertible numeric strings, floats or Booleans to be passed to that function, and would convert them.[71] However, PHP 7 has a "strict

typing" mode which, when used, disallows such conversions for function calls and returns within a file.[71]

PHP objects

Basic object-oriented programming functionality was added in PHP 3 and improved in PHP 4.[11] This allowed for PHP to gain further abstraction, making creative tasks easier for programmers using the language. Object handling was completely rewritten for PHP 5, expanding the feature set and enhancing performance.[230] In previous versions of PHP, objects were handled like value types.[230] The drawback of this method was that code had to make heavy use of PHP's "reference" variables if it wanted to modify an object it was passed rather than creating a copy of it. In the new approach, objects are referenced by handle, and not by value.

PHP 5 introduced private and protected member variables and methods, along with abstract classes, final classes, abstract methods, and final methods. It also introduced a standard way of declaring constructors and destructors, similar to that of other object-oriented languages such as C++, and a standard exception handling model. Furthermore, PHP 5 added interfaces and allowed for multiple interfaces to be implemented. There are special interfaces that allow objects to interact with the runtime system. Objects implementing ArrayAccess can be used with array syntax and objects implementing Iterator or IteratorAggregate can be used with the `foreach` language construct. There is no virtual table feature in the engine, so static variables are bound with a name instead of a reference at compile time.[231]

If the developer creates a copy of an object using the reserved word `clone`, the Zend engine will check whether a `__clone()` method has been defined. If not, it will call a default `__clone()` which will copy the object's properties. If a `__clone()` method is defined, then it will be responsible for setting the necessary properties in the created object. For convenience, the engine will supply a function that imports the properties of the source object, so the programmer can start with a by-value replica of the source object and only override properties that need to be changed.[232]

The visibility of PHP properties and methods is defined using the keywords `public`, `private`, and `protected`. The default is public, if only var is used; `var` is a synonym for `public`. Items declared `public` can be accessed everywhere. `protected` limits access to inherited classes (and to the class that defines the item). `private` limits visibility only to the class that defines the item.[233] Objects of the same type have access to each other's private and protected members even though they are not the same instance.

Example

The following is a basic example of object-oriented programming in PHP 8:

```php
<?php

abstract class User
{
    protected string $name;

    public function __construct(string $name)
    {
        // make first letter uppercase and the rest lowercase
        $this->name = ucfirst(strtolower($name));
    }

```

```php
    public function greet(): string
    {
        return "Hello, my name is " . $this->name;
    }

    abstract public function job(): string;
}

class Student extends User
{
    public function __construct(string $name, private string $course)
    {
        parent::__construct($name);
    }

    public function job(): string
    {
        return "I learn " . $this->course;
    }
}

class Teacher extends User
{
    public function __construct(string $name, private array $teachingCourses)
    {
        parent::__construct($name);
    }

    public function job(): string
    {
        return "I teach " . implode(", ", $this->teachingCourses);
    }
}

$students = [
    new Student("Alice", "Computer Science"),
    new Student("Bob", "Computer Science"),
    new Student("Charlie", "Business Studies"),
];

$teachers = [
    new Teacher("Dan", ["Computer Science", "Information Security"]),
    new Teacher("Erin", ["Computer Science", "3D Graphics Programming"]),
    new Teacher("Frankie", ["Online Marketing", "Business Studies", "E-commerce"]),
];

foreach ([$students, $teachers] as $users) {
    echo $users[0]::class . "s:\n";

    array_walk($users, function (User $user) {
        echo "{$user->greet()}, {$user->job()}\n";
    });
}
```

This program outputs the following:

```
  Students:
Hello, my name is Alice, I learn Computer Science
Hello, my name is Bob, I learn Computer Science
Hello, my name is Charlie, I learn Business Studies
Teachers:
Hello, my name is Dan, I teach Computer Science, Information Security
Hello,  my  name  is  Erin,  I  teach  Computer  Science,  3D  Graphics
```

Implementations

The only complete PHP implementation is the original, known simply as PHP. It is the most widely used and is powered by the Zend Engine. To disambiguate it from other implementations, it is sometimes unofficially called "Zend PHP". The Zend Engine compiles PHP source code on-the-fly into an internal format that it can execute, thus it works as an interpreter.[234][235] It is also the "reference implementation" of PHP, as PHP has no formal specification, and so the semantics of Zend PHP define the semantics of PHP. Due to the complex and nuanced semantics of PHP, defined by how Zend works, it is difficult for competing implementations to offer complete compatibility.[236]

PHP's single-request-per-script-execution model, and the fact that the Zend Engine is an interpreter, leads to inefficiency; as a result, various products have been developed to help improve PHP performance. In order to speed up execution time and not have to compile the PHP source code every time the web page is accessed, PHP scripts can also be deployed in the PHP engine's internal format by using an opcode cache, which works by caching the compiled form of a PHP script (opcodes) in shared memory to avoid the overhead of parsing and compiling the code every time the script runs. An opcode cache, Zend Opcache, is built into PHP since version 5.5.[237] Another example of a widely used opcode cache is the Alternative PHP Cache (APC), which is available as a PECL extension.[238]

While Zend PHP is still the most popular implementation, several other implementations have been developed. Some of these are compilers or support JIT compilation, and hence offer performance benefits over Zend PHP at the expense of lacking full PHP compatibility. Alternative implementations include the following:

- HHVM (HipHop Virtual Machine) – developed at Facebook and available as open source, it converts PHP code into a high-level bytecode (commonly known as an intermediate language), which is then translated into x86-64 machine code dynamically at runtime by a just-in-time (JIT) compiler, resulting in up to 6× performance improvements.[239] However, since version 7.2 Zend has outperformed HHVM,[240] and HHVM 3.24 is the last version to officially support PHP.[241]
    - HipHop – developed at Facebook and available as open source, it transforms the PHP scripts into C++ code and then compiles the resulting code, reducing the server load up to 50%. In early 2013, Facebook deprecated it in favour of HHVM due to multiple reasons, including deployment difficulties and lack of support for the whole PHP language, including the `create_function()` and `eval()` constructs.[242]
- Parrot – a virtual machine designed to run dynamic languages efficiently; the cross-translator Pipp transforms the PHP source code into the Parrot intermediate representation, which is then translated into the Parrot's bytecode and executed by the virtual machine.
- PeachPie – a second-generation compiler to .NET Common Intermediate Language (CIL) bytecode, built on the Roslyn platform; successor of Phalanger, sharing several architectural components
- Phalanger – compiles PHP into .Net Common Intermediate Language bytecode; predecessor of PeachPie

- Quercus – compiles PHP into Java bytecode

Licensing

PHP is free software released under the PHP License, which stipulates that:[243]

> Products derived from this software may not be called "PHP", nor may "PHP" appear in their name, without prior written permission from group@php.net. You may indicate that your software works in conjunction with PHP by saying "Foo for PHP" instead of calling it "PHP Foo" or "phpfoo".

This restriction on the use of "PHP" makes the PHP License incompatible with the GNU General Public License (GPL), while the Zend License is incompatible due to an advertising clause similar to that of the original BSD license.[244]

Development and community

PHP includes various free and open-source libraries in its source distribution or uses them in resulting PHP binary builds. PHP is fundamentally an Internet-aware system with built-in modules for accessing File Transfer Protocol (FTP) servers and many database servers, including PostgreSQL, MySQL, Microsoft SQL Server and SQLite (which is an embedded database), LDAP servers, and others. Numerous functions are familiar to C programmers, such as those in the stdio family, are available in standard PHP builds.[245]

PHP allows developers to write extensions in C to add functionality to the PHP language. PHP extensions can be compiled statically into PHP or loaded dynamically at runtime. Numerous extensions have been written to add support for the Windows API, process management on Unix-like operating systems, multibyte strings (Unicode), cURL, and several popular compression formats. Other PHP features made available through extensions include integration with Internet Relay Chat (IRC), dynamic generation of images and Adobe Flash content, *PHP Data Objects* (PDO) as an abstraction layer used for accessing databases,[246][247][248][249][250][251][252] and even speech synthesis. Some of the language's core functions, such as those dealing with strings and arrays, are also implemented as extensions.[253] The PHP Extension Community Library (PECL) project is a repository for extensions to the PHP language.[254]

Some other projects, such as *Zephir*, provide the ability for PHP extensions to be created in a high-level language and compiled into native PHP extensions. Such an approach, instead of writing PHP extensions directly in C, simplifies the development of extensions and reduces the time required for programming and testing.[255]

By December 2018 the PHP Group consisted of ten people: Thies C. Arntzen, Stig Bakken, Shane Caraveo, Andi Gutmans, Rasmus Lerdorf, Sam Ruby, Sascha Schumann, Zeev Suraski, Jim Winstead, and Andrei Zmievski.[256]

Zend Technologies provides a PHP Certification based on PHP 8[257] exam (and previously based on PHP 7 and 5.5) for programmers to become certified PHP developers.

The PHP Foundation

On 26 November 2021, the JetBrains blog announced the creation of The PHP Foundation, which will sponsor the design and development of PHP.[259]

Year	Commits	Reviews	RFCs
2022[260]	683	283	8
2023[261]	784	702	17

The foundation hires "Core Developers" to work on the PHP language's core repository. Roman Pronskiy, a member of the foundation's board, said that they aim to pay "market salaries" to developers.[262]

PHP

The PHP Foundation

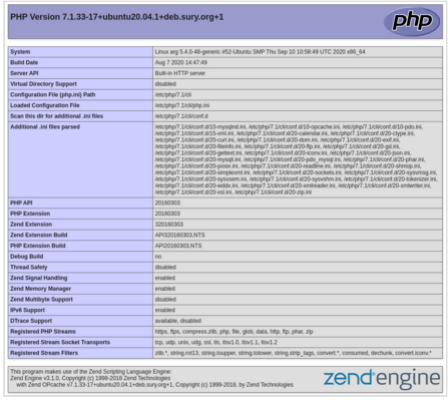
Formation	November 22, 2021
Founder	Automattic, Laravel, Acquia, Zend, Private Packagist, Symfony, Craft CMS, Tideways, PrestaShop, JetBrains[258]
Website	https://thephp.foundation

The response to the foundation has mostly been positive, with the foundation being praised for better supporting the language and helping to stop the decrease in the language's popularity.[263][264] However, it has also been criticised for adding breaking changes to minor versions of PHP, such as in PHP 8.2 where initialising members of a class out-with the original class scope would cause deprecation errors,[265] these changes impacted a number of open source projects including WordPress. [266]

Germany's Sovereign Tech Fund provided more than 200,000 Euros to support the PHP Foundation.[267]

Installation and configuration

There are two primary ways for adding support for PHP to a web server – as a native web server module, or as a CGI executable. PHP has a direct module interface called server application programming interface (SAPI), which is supported by many web servers including Apache HTTP Server, Microsoft IIS and iPlanet Web Server. Some other web servers, such as OmniHTTPd, support the Internet Server Application Programming Interface (ISAPI), which is Microsoft's web server module interface. If PHP has no module support for a web server, it can always be used as a Common Gateway Interface (CGI) or FastCGI processor; in that case, the web server is configured to use PHP's CGI executable to process all requests to PHP files.[268]



Example output of the phpinfo() function in PHP 7.1

PHP-FPM (FastCGI Process Manager) is an alternative FastCGI implementation for PHP, bundled with the official PHP distribution since version 5.3.3.[269] When compared to the older FastCGI implementation, it contains some additional features, mostly useful for heavily loaded web servers.[270]

When using PHP for command-line scripting, a PHP command-line interface (CLI) executable is needed. PHP supports a CLI server application programming interface (SAPI) since PHP 4.3.0.[271] The main focus of this SAPI is developing shell applications using PHP. There are quite a few differences between the CLI SAPI and other SAPIs, although they do share many of the same behaviours.[272]

PHP has a direct module interface called SAPI for different web servers;[273] in case of PHP 5 and Apache 2.0 on Windows, it is provided in form of a DLL file called `php5apache2.dll`,[274] which is a module that, among other functions, provides an interface between PHP and the web server, implemented in a form that the server understands. This form is what is known as a SAPI.

There are different kinds of SAPIs for various web server extensions. For example, in addition to those listed above, other SAPIs for the PHP language include the Common Gateway Interface and command-line interface.[273][275]

PHP can also be used for writing desktop graphical user interface (GUI) applications, by using the "PHP Desktop" (https://github.com/cztomczak/phpdesktop). *GitHub*. or discontinued PHP-GTK extension. PHP-GTK is not included in the official PHP distribution,[268] and as an extension, it can be used only with PHP versions 5.1.0 and newer. The most common way of installing PHP-GTK is by compiling it from the source code.[276]

When PHP is installed and used in cloud environments, software development kits (SDKs) are provided for using cloud-specific features. For example:

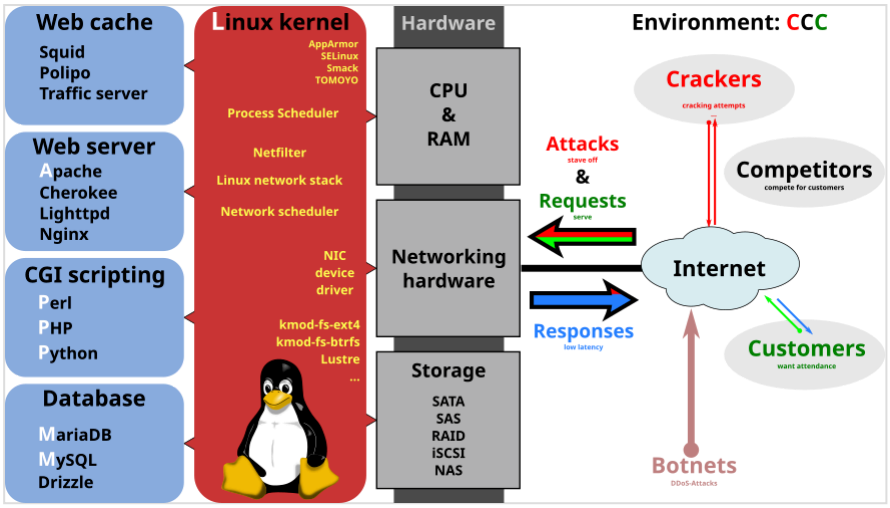
- Amazon Web Services provides the AWS SDK for PHP[277]
- Microsoft Azure can be used with the Windows Azure SDK for PHP.[278]

Numerous configuration options are supported, affecting both core PHP features and extensions.[279][280] Configuration file `php.ini` is searched for in different locations, depending on the way PHP is used.[281] The configuration file is split into various sections,[282] while some of the configuration options can be also set within the web server configuration.[283]

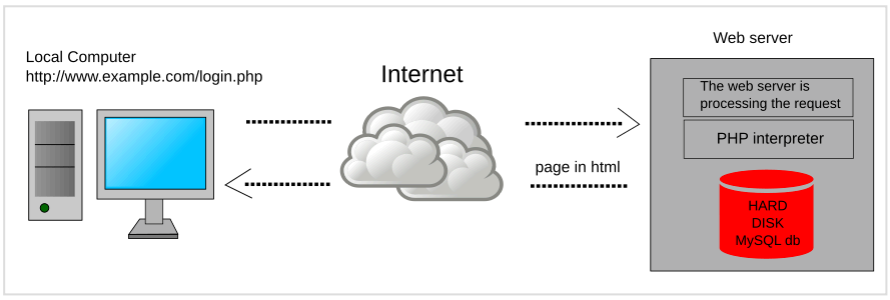
Use

PHP is a general-purpose scripting language that is especially suited to server-side web development, in which case PHP generally runs on a web server. Any PHP code in a requested file is executed by the PHP runtime, usually to create dynamic web page content or dynamic images used on websites or elsewhere.[284] It can also be used for command-line scripting and client-side graphical user interface (GUI) applications. PHP can be deployed on most web servers, many operating systems and platforms, and can be used with many relational database management systems (RDBMS). Most web hosting providers support PHP for use by their clients. It is available free of charge, and the PHP Group provides the complete source code for users to build, customize and extend for their own use.[17]

Originally designed to create dynamic web pages, PHP now focuses mainly on server-side scripting,[285] and it is similar to other server-side scripting languages that provide dynamic content from a web server to a client, such as Python, Microsoft's ASP.NET, Sun Microsystems' JavaServer Pages,[286] and `mod_perl`. PHP has also attracted the development of many software frameworks that provide building blocks and a design structure to promote rapid application development (RAD). Some of these include PRADO, CakePHP, Symfony, CodeIgniter, Laravel, Yii Framework, Phalcon and Laminas, offering features similar to other web frameworks.



A broad overview of the LAMP software bundle, displayed here together with Squid



Dynamic web page: example of server-side scripting (PHP and MySQL)

The LAMP architecture has become popular in the web industry as a way of deploying web applications.[287] PHP is commonly used as the *P* in this bundle alongside Linux, Apache and MySQL, although the *P* may also refer to Python, Perl, or some mix of the three. Similar packages, WAMP and MAMP, are also available for Windows and macOS, with the first letter standing for the respective operating system. Although both PHP and Apache are provided as part of the macOS base install, users of these packages seek a simpler installation mechanism that can be more easily kept up to date.

For specific and more advanced usage scenarios, PHP offers a well-defined and documented way for writing custom extensions in C or C++.[288][289][290][291][292][293][294] Besides extending the language itself in form of additional libraries, extensions are providing a way for improving execution speed where it is critical and there is room for improvements by using a true compiled language.[295][296] PHP also offers well-defined ways for embedding itself into other software projects. That way PHP can be easily used as an internal scripting language for another project, also providing tight interfacing with the project's specific internal data structures.[297]

PHP received mixed reviews due to lacking support for multithreading at the core language level,[298] though using threads is made possible by the "pthreads" PECL extension.[299][300]

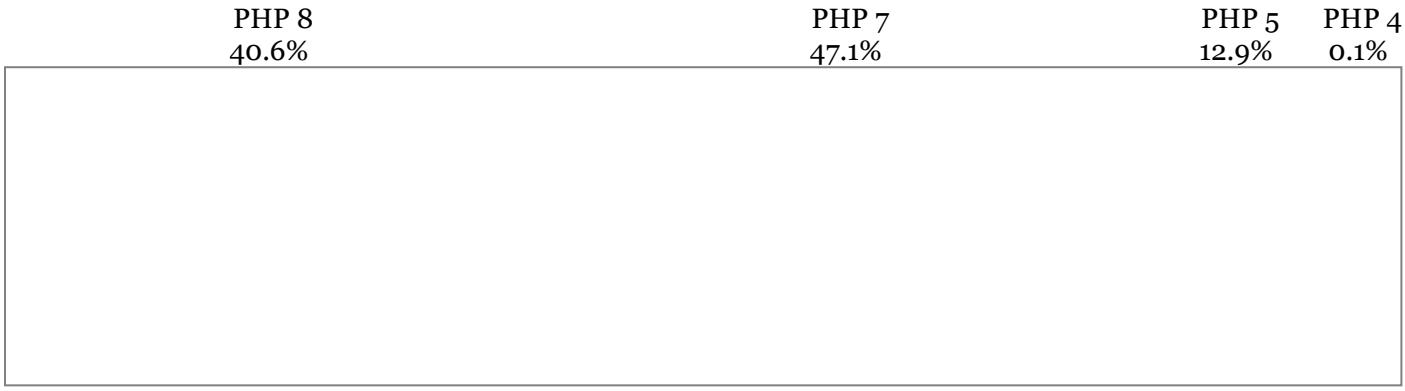
A command line interface, php-cli, and two ActiveX Windows Script Host scripting engines for PHP have been produced.

Popularity and usage statistics

PHP is used for Web content management systems including MediaWiki,[301] WordPress,[302] Joomla,[303] Drupal,[304] Moodle,[305] eZ Publish, eZ Platform, and SilverStripe.[306]

As of January 2013, PHP was used in more than 240 million websites (39% of those sampled) and was installed on 2.1 million web servers.[307]

As of 21 January 2025 (two months after PHP 8.4's release), PHP is used as the server-side programming language on 75.0% of websites where the language could be determined; PHP 7 is the most used version of the language with 47.1% of websites using PHP being on that version, while 40.6% use PHP 8, 12.2% use PHP 5 and 0.1% use PHP 4.[19]



Security

In 2019, 11% of all vulnerabilities listed by the National Vulnerability Database were linked to PHP;[312] historically, about 30% of all vulnerabilities listed since 1996 in this database are linked to PHP. Technical security flaws of the language itself or of its core libraries are not frequent (22 in 2009, about 1% of the total although PHP applies to about 20% of programs listed).[313] Recognizing that programmers make mistakes, some languages include taint checking to automatically detect the lack of input validation which induces many issues. Such a feature has been proposed for PHP in the past, but either been rejected or the proposal abandoned.[314][315][316]

Third-party projects such as Suhosin[317] and Snuffleupagus[318] aim to remove or change dangerous parts of the language.

Historically, old versions of PHP had some configuration parameters and default values for such runtime settings that made some PHP applications prone to security issues. Among these, `magic_quotes_gpc` and `register_globals`[319] configuration directives were the best known; the latter made any URL parameters become PHP variables, opening a path for serious security vulnerabilities by allowing an attacker to set the value of any uninitialized global variable and interfere with the execution of a PHP script. Support for "magic quotes" and "register globals" settings has been deprecated since PHP 5.3.0, and removed from PHP 5.4.0.[320]

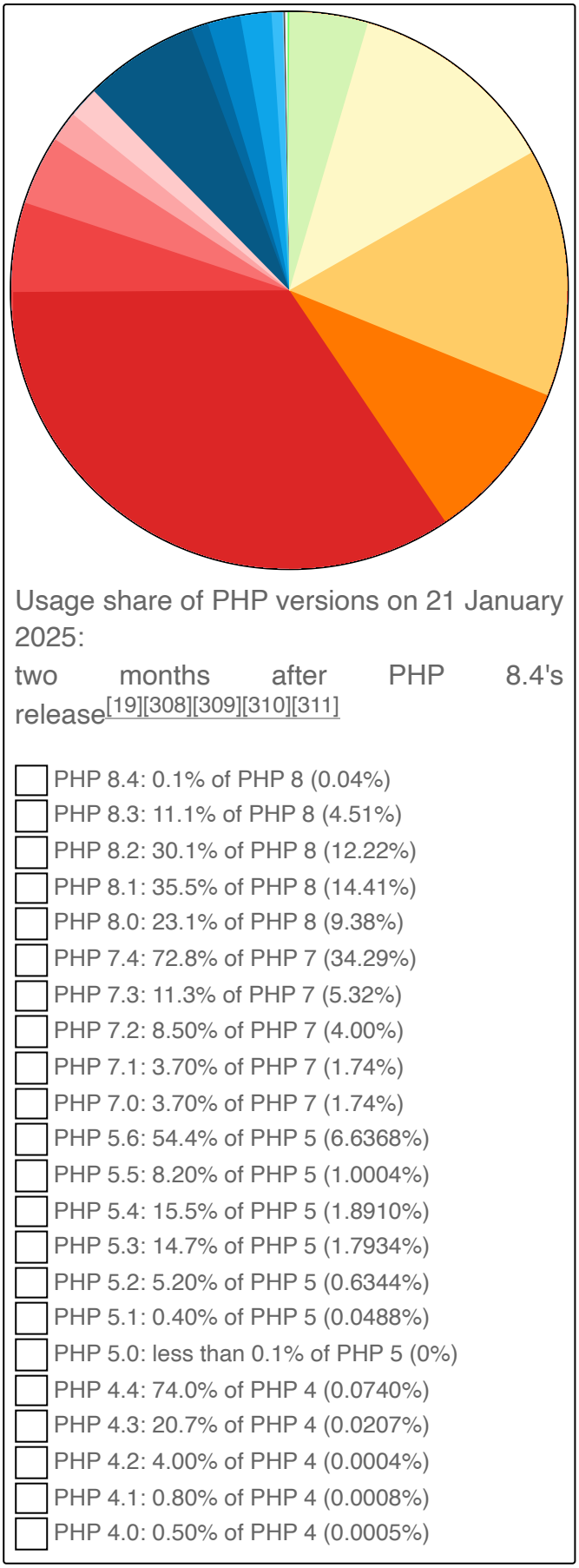
Another example for the potential runtime-settings vulnerability comes from failing to disable PHP execution (for example by using the `engine` configuration directive)[321] for the directory where uploaded files are stored; enabling it can result in the execution of malicious code embedded

within the uploaded files.[322][323][324] The best practice is to either locate the image directory outside of the document root available to the web server and serve it via an intermediary script or disable PHP execution for the directory which stores the uploaded files.

Also, enabling the dynamic loading of PHP extensions (via `enable_dl` configuration directive)[325] in a shared web hosting environment can lead to security issues.[326][327]

Implied type conversions that result in different values being treated as equal, sometimes against the programmer's intent, can lead to security issues. For example, the result of the comparison `'0e1234' == '0'` is `true`, because strings that are parsable as numbers are converted to numbers; in this case, the first compared value is treated as scientific notation having the value (0×10^{1234}), which is zero. Errors like this resulted in authentication vulnerabilities in Simple Machines Forum,[328] Typo3[329] and phpBB[330] when MD5 password hashes were compared. The recommended way is to use `hash_equals()` (`https://secure.php.net/hash_equals`) (for timing attack safety), `strcmp` or the identity operator (===), as `'0e1234' === '0'` results in `false`.

In a 2013 analysis of over 170,000 website defacements, published by Zone-H, the most frequently (53%) used technique was the exploitation of file inclusion vulnerability, mostly related to insecure usage of the PHP language constructs `include`, `require`, and `allow_url_fopen`.[331][332]



Cryptographic Security

PHP includes `rand()`[333] and `mt_rand()`[334] functions which use a pseudorandom number generator, and are not cryptographically secure. As of version 8.1, the `random_int()` function is included, which uses a cryptographically secure source of randomness provided by the system.[335]

There are two attacks that can be performed over PHP entropy sources: "seed attack" and "state recovery attack". As of 2012, a $250 GPU can perform up to 2^{30} MD5 calculations per second, while a $750 GPU can perform four times as many calculations at the same time.[336] In

combination with a "birthday attack" this can lead to serious security vulnerabilities.

Long-Term Support

The PHP development team provides official bug fixes for 2 years following release of each minor version, followed by another 2 years where only security fixes are released.[337] After this, the release is considered end of life and no longer officially supported.

Extended Long-Term Support beyond this is available from commercial providers, such as Zend and others[338][339]

See also

- Comparison of programming languages
- List of Apache–MySQL–PHP packages
- List of PHP accelerators
- List of PHP editors
- PEAR (PHP Extension and Application Repository)
- PHP accelerator
- Template processor
- XAMPP (free and open-source cross-platform web server solution stack package)
- Zend Server

References

1. Lerdorf, Rasmus (June 8, 1995). "Announce: Personal Home Page Tools (PHP Tools)" (https://groups.google.com/group/comp.infosystems.www.authoring.cgi/msg/cc7d43454d64d133?pli=1). Retrieved 7 June 2011.
2. Lerdorf, Rasmus (2007-04-26). "PHP on Hormones – history of PHP presentation by Rasmus Lerdorf given at the MySQL Conference in Santa Clara, California" (https://web.archive.org/web/20130729204354id_/http://itc.conversationsnetwork.org/shows/detail3298.html). The Conversations Network. Archived from the original (http://itc.conversationsnetwork.org/shows/detail3298.html) on 2013-07-29.
3. "PHP: Hypertext Preprocessor" (https://www.php.net/index.php#2025-01-17-1). *www.php.net*.
4. "PHP: Function arguments – Manual" (https://secure.php.net/manual/en/functions.arguments.php#functions.arguments.type-declaration.strict). *secure.php.net*.
5. "PHP: Release Archives (museum)" (https://museum.php.net/php3/). *museum.php.net*.
6. "PHP: Preface – Manual" (https://www.php.net/manual/en/preface.php).
7. Stogov, Dmitry [@dstogov] (2015-12-04). "It's not a secret that some #PHP7 optimization ideas came from HHVM, LuaJIT and V8. Thank you @HipHopVM @SaraMG. #php7thankyou" (https://x.com/dstogov/status/672864802474229760) (Tweet) – via Twitter.
8. "PHP: Hypertext Preprocessor" (https://www.php.net/). *www.php.net*. Retrieved 2020-02-12.

9. Krill, Paul (2013-11-18). "Believe the hype: PHP founder backs Facebook's HipHop technology" (https://www.infoworld.com/article/2609877/believe-the-hype--php-founder-backs-facebook-s-hiphop-technology.html). *InfoWorld*. Retrieved 2022-10-13.
10. "Announce: Personal Home Page Tools (PHP Tools)" (https://groups.google.com/g/comp.infosystems.www.authoring.cgi/c/PyJ25gZ6z7A/m/M9FkTUVDfcwJ?pli=1). *groups.google.com*. Retrieved 2022-11-03.
11. "History of PHP and related projects" (https://www.php.net/history). The PHP Group. Retrieved 2008-02-25.
12. "History of PHP" (https://php.net/manual/en/history.php.php). *php.net*.
13. Olsson, Mikael (2013-09-04). *PHP Quick Scripting Reference* (https://books.google.com/books?id=_ahBAAAAQBAJ). Apress. ISBN 978-1-4302-6284-8.
14. PHP Manual: Preface (https://php.net/manual/en/preface.php), www.php.net.
15. "Introduction: What can PHP do?" (https://php.net/manual/en/intro-whatcando.php). *PHP Manual*. Retrieved 2009-03-05.
16. *helicopter: Port of node-ar-drone which allows user to control a Parrot AR Drone over PHP: jolicode/php-ar-drone* (https://github.com/jolicode/php-ar-drone), JoliCode, 2019-01-11, retrieved 2019-02-23
17. "Embedding PHP in HTML" (https://web.archive.org/web/20080219180226/http://www.onlamp.com/pub/a/php/2001/05/03/php_foundations.html). O'Reilly. 2001-05-03. Archived from the original (http://www.onlamp.com/pub/a/php/2001/05/03/php_foundations.html) on 2008-02-19. Retrieved 2008-02-25.
18. "PHP: Unsupported Branches" (https://www.php.net/eol.php). *www.php.net*.
19. "Usage statistics of PHP for websites" (https://w3techs.com/technologies/details/pl-php). *W3Techs – World Wide Web Technology Surveys*. W3Techs. Retrieved 21 January 2025.
20. Lerdorf, Rasmus (2012-07-20). "I wonder why people keep writing that PHP was ever written in Perl. It never was. #php" (https://twitter.com/rasmus/status/226405807305138176). Twitter. Retrieved 2014-09-04.
21. Lerdorf, Rasmus (2007-04-26). "PHP on Hormones" (https://web.archive.org/web/20190106230504/http://web.archive.org/web/20130729204354id_/http://itc.conversationsnetwork.org/shows/detail3298.html). The Conversations Network. Archived from the original (http://itc.conversationsnetwork.org/shows/detail3298.html) (mp3) on 2019-01-06. Retrieved 2009-06-22.
22. Lerdorf, Rasmus (2007). "Slide 3" (http://talks.php.net/show/mysql07key/3). *slides for 'PHP on Hormones' talk*. The PHP Group. Retrieved 2009-06-22.
23. Lerdorf, Rasmus (1995-06-08). "Announce: Personal Home Page Tools (PHP Tools)" (https://groups.google.com/group/comp.infosystems.www.authoring.cgi/msg/cc7d43454d64d133). Newsgroup: comp.infosystems.www.authoring.cgi (news:comp.infosystems.www.authoring.cgi). Retrieved 2006-09-17.
24. "Rasmus Lerdorf, Senior Technical Yahoo: PHP, Behind the Mic" (https://web.archive.org/web/20130728125152/http://itc.conversationsnetwork.org/shows/detail58.html). 2003-11-19. Archived from the original (http://itc.conversationsnetwork.org/shows/detail58.html) on 2013-07-28.
25. Alshetwi, A.B.; Rahmat, R. A. A. O.; Borhan, M. N.; Ismael, S.; Ali, A.; Irtema, H. I. M.; Alfakhria, A. Y. (2018). "Web-Based Expert System for Optimizing of Traffic Road in Developing Countries" (https://www.researchgate.net/publication/326727672). Retrieved 13 Feb 2024.
26. "Problems with PHP" (http://toykeeper.net/soapbox/php_problems/). Retrieved 20 December 2010.
27. "php.internals: Re: Function name consistency" (http://news.php.net/php.internals/70950). *news.php.net*. 2013-12-28. Retrieved 2014-02-09.
28. Rasmus Lerdorf (Dec 16, 2013). "Re: Flexible function naming" (http://news.php.net/php.internals/70691). Newsgroup: php.internals (news:php.internals). Retrieved December 26, 2013.
29. "PHP – Acronym Meaning Vote" (https://web.archive.org/web/20000815063125/http://il.php.net/vote_listing.php3). *PHP.net*. Archived from the original (http://il.php.net/vote_listing.php3) on August 15, 2000.

30. "Zend Engine version 2.0: Feature Overview and Design" (https://web.archive.org/web/20060719204721/http://www.zend.com/zend/zend-engine-summary.php). Zend Technologies Ltd. Archived from the original (http://www.zend.com/zend/zend-engine-summary.php) on 2006-07-19. Retrieved 2006-09-17.
31. "php.net 2007 news archive" (https://www.php.net/archive/2007.php). The PHP Group. 2007-07-13. Retrieved 2008-02-22.
32. Kerner, Sean Michael (2008-02-01). "PHP 4 is Dead—Long Live PHP 5" (https://web.archive.org/web/20180806115411/http://www.internetnews.com/dev-news/article.php/3725291). InternetNews. Archived from the original (https://www.internetnews.com/developer/php-4-is-dead%ef%bf%bdlong-live-php-5/) on 2018-08-06. Retrieved 2018-12-16.
33. Trachtenberg, Adam (2004-07-15). "Why PHP 5 Rocks!" (https://web.archive.org/web/20160331232050/http://www.onlamp.com/pub/a/php/2004/07/15/UpgradePHP5.html). O'Reilly. Archived from the original (http://www.onlamp.com/pub/a/php/2004/07/15/UpgradePHP5.html) on 2016-03-31. Retrieved 2008-02-22.
34. "Late Static Binding in PHP" (http://www.digitalsandwich.com/archives/53-Late-Static-Binding-in-PHP.html). Digital Sandwich. 2006-02-23. Retrieved 2008-03-25.
35. "Static Keyword" (https://www.php.net/language.oop5.static). The PHP Group. Retrieved 2008-03-25.
36. "GoPHP5" (https://web.archive.org/web/20110717133313/http://gophp5.org/projects). Archived from the original (http://www.gophp5.org/projects) on 2011-07-17.
37. "PHP projects join forces to Go PHP 5" (https://web.archive.org/web/20190804012720/http://gophp5.org/sites/gophp5.org/files/press_release.pdf) (PDF). *GoPHP5 Press Release*. Archived from the original (http://gophp5.org/sites/gophp5.org/files/press_release.pdf) (PDF) on 2019-08-04. Retrieved 2008-02-23.
38. "GoPHP5" (https://web.archive.org/web/20110427101913/http://www.gophp5.org/). GoPHP5. Archived from the original (http://gophp5.org/) on 2011-04-27. Retrieved 2008-02-22.
39. "PHP Installation and Configuration" (https://www.php.net/manual/en/install.php). *php.net*. Retrieved 2013-10-29.
40. "PHP for Windows: Binaries and sources releases (5.3)" (https://windows.php.net/download/#php-5.3). *php.net*. Retrieved 2013-10-29.
41. "PHP for Windows: Binaries and sources releases (5.4)" (https://windows.php.net/download/#php-5.4). *php.net*. Retrieved 2013-10-29.
42. "PHP for Windows: Binaries and sources releases (5.5)" (https://windows.php.net/download/#php-5.5). *php.net*. Retrieved 2013-10-29.
43. "PHP: Supported Versions" (https://php.net/supported-versions.php).
44. "Types: Strings (PHP Manual)" (https://php.net/manual/en/language.types.string.php). *PHP.net*. Retrieved 2013-09-22.
45. "Details of the String Type (PHP Manual)" (https://www.php.net/manual/en/language.types.string.php#language.types.string.details). *PHP.net*. Retrieved 2021-09-22.
46. Andrei Zmievski (2005-08-10). "PHP Unicode support design document" (https://marc.info/?l=php-internals&m=112365908921757&w=1) (Mailing list). Retrieved 2014-02-09.
47. "PHP 5.5 or 6.0" (http://news.php.net/php.internals/17668). Retrieved 2014-02-09.
48. Andrei Zmievski (2011-04-22). "The Good, the Bad, and the Ugly: What Happened to Unicode and PHP 6" (https://www.slideshare.net/andreizm/the-good-the-bad-and-the-ugly-what-happened-to-unicode-and-php-6). Retrieved 2014-02-09.
49. Rasmus Lerdorf (2010-03-11). "PHP 6" (http://news.php.net/php.internals/47120) (Mailing list). Retrieved 2014-02-07.
50. "The Neverending Muppet Debate of PHP 6 v PHP 7" (https://web.archive.org/web/20151119132438/https://philsturgeon.uk/php/2014/07/23/neverending-muppet-debate-of-php-6-v-php-7/). Archived from the original (https://philsturgeon.uk/php/2014/07/23/neverending-muppet-debate-of-php-6-v-php-7/) on 2015-11-19. Retrieved 2015-11-19.
51. "RFC: Name of Next Release of PHP" (https://wiki.php.net/rfc/php6). *php.net*. 2014-07-07. Retrieved 2014-07-15.

52. "Re: [PHP-DEV] [VOTE] [RFC] Name of Next Release of PHP (again)" (https://www.mail-archive.com/internals@lists.php.net/msg68598.html). 2014-07-30. Retrieved 2014-07-30.
53. "phpng: Refactored PHP Engine with Big Performance Improvement" (http://news.php.net/php.internals/73888). *news.php.net*.
54. "PHP: rfc:phpng" (https://wiki.php.net/rfc/phpng). *php.net*. Retrieved 16 December 2014.
55. "PHP: phpng" (https://wiki.php.net/phpng). *php.net*. Retrieved 2014-07-15.
56. "Merge branch 'ZendEngine3' " (https://github.com/php/php-src/commit/150dc69d6eee35738f505e925ee664c02060196d). *github.com*. 2014-12-05. Retrieved 2014-12-05.
57. "PHP: Release Process" (https://wiki.php.net/rfc/releaseprocess). 2011-06-20. Retrieved 2013-10-06.
58. "PHP RFC: Exceptions in the engine (for PHP 7)" (https://wiki.php.net/rfc/engine_exceptions_for_php7). *php.net*. Retrieved 2015-05-21.
59. "PHP RFC: Uniform Variable Syntax" (https://wiki.php.net/rfc/uniform_variable_syntax). *php.net*. 2014-05-31. Retrieved 2014-07-30.
60. "Online PHP editor | output for udRhX" (https://3v4l.org/udRhX). *3v4l.org*.
61. "PHP RFC: Fix "foreach" behavior" (https://wiki.php.net/rfc/php7_foreach). *php.net*. Retrieved 2015-05-21.
62. "PHP RFC: Constructor behaviour of internal classes" (https://wiki.php.net/rfc/internal_constructor_behaviour). *php.net*. Retrieved 2015-05-21.
63. "PHP RFC: Removal of dead or not yet PHP7 ported SAPIs and extensions" (https://wiki.php.net/rfc/removal_of_dead_sapis_and_exts). *php.net*. Retrieved 2015-05-21.
64. "PHP RFC: Fix list() behavior inconsistency" (https://wiki.php.net/rfc/fix_list_behavior_inconsistency). *php.net*. Retrieved 2015-05-21.
65. "PHP RFC: Remove alternative PHP tags" (https://wiki.php.net/rfc/remove_alternative_php_tags). *php.net*. Retrieved 2015-05-21.
66. "PHP RFC: Make defining multiple default cases in a switch a syntax error" (https://wiki.php.net/rfc/switch.default.multiple). *php.net*. Retrieved 2015-05-21.
67. "PHP RFC: Remove hex support in numeric strings" (https://wiki.php.net/rfc/remove_hex_support_in_numeric_strings). *php.net*. Retrieved 2015-05-21.
68. "PHP RFC: Integer Semantics" (https://wiki.php.net/rfc/integer_semantics). *php.net*. Retrieved 2015-05-21. "Making NaN and Infinity always become zero when cast to integer means more cross-platform consistency, and is also less surprising than what is currently produces"
69. "PHP RFC: ZPP Failure on Overflow" (https://wiki.php.net/rfc/zpp_fail_on_overflow). *php.net*. Retrieved 2015-05-21.
70. "RFC: Return Types" (https://wiki.php.net/rfc/return_types). *php.net*. 2015-01-27. Retrieved 2015-01-28.
71. "RFC: Scalar Type Declarations" (https://wiki.php.net/rfc/scalar_type_hints_v5). *php.net*. 2015-03-16. Retrieved 2015-03-17.
72. Brent. "What's new in PHP 8" (https://stitcher.io/blog/new-in-php-8). *Stitcher*. Retrieved 22 September 2020.
73. "PHP 8 Released" (https://www.php.net/releases/8.0/en.php). *PHP*. Retrieved 27 November 2020.
74. "PHP: rfc:jit" (https://wiki.php.net/rfc/jit). *wiki.php.net*. Retrieved 2019-04-05.
75. Brent. "PHP 8: JIT performance in real-life web applications" (https://stitcher.io/blog/jit-in-real-life-web-applications). *Stitcher.io*. Retrieved 4 October 2020.
76. Rethams, Derick. "PHP 8: A Quick Look at JIT" (https://derickrethans.nl/a-quick-look-at-jit.html).
77. Popov, Nikita (13 July 2020). " "What's new in PHP 8.0?" Nikita Popov" (https://www.youtube.com/watch?v=NbBRXwu1Md8). PHP fwdays. Archived (https://ghostarchive.org/varchive/youtube/20211211/NbBRXwu1Md8) from the original on 2021-12-11. Retrieved 4 October 2020.
78. Daniele, Carlo (25 May 2020). "What's New in PHP 8 (Features, Improvements, and the JIT Compiler)" (https://kinsta.com/blog/php-8/). *Kinsta*. Retrieved 24 December 2020.

79. Redmond, Paul (15 July 2020). "Match Expression is Coming to PHP 8" (https://laravel-news.com/match-expression-php-8). *Laravel News*. Retrieved 4 October 2020.
80. "PHP 8.0: Match Expressions" (https://php.watch/versions/8.0/match-expression). *PHP Watch*. Retrieved 4 October 2020.
81. Barnes, Eric (27 November 2020). "PHP 8 is now Released!" (https://laravel-news.com/php-8). *Laravel News*. Retrieved 24 December 2020.
82. "PHP RFC: throw expression" (https://wiki.php.net/rfc/throw_expression). *wiki.php.net*. Retrieved 14 August 2020.
83. "PHP RFC: Nullsafe operator" (https://wiki.php.net/rfc/nullsafe_operator). *wiki.php.net*. Retrieved 14 August 2020.
84. Roose, Brent. "PHP 8: Constructor property promotion" (https://stitcher.io/blog/constructor-promotion-in-php-8). Retrieved 30 April 2024.
85. "PHP: rfc:weakrefs" (https://wiki.php.net/rfc/weakrefs). *wiki.php.net*. Retrieved 2019-04-05.
86. Merchant, Amit (13 June 2020). "These new string functions are coming in PHP 8" (https://www.amitmerchant.com/new-string-functions-php8/). *Amit Merchant*. Retrieved 4 October 2020.
87. Popov, Nikita. "Call for participation: Annotating internal function argument and return types" (https://externals.io/message/106522). *Externals*. Retrieved 19 November 2020.
88. "PHP 8 ChangeLog" (https://www.php.net/ChangeLog-8.php#8.1.0). *PHP.net*. Retrieved 2024-01-05.
89. "PHP: PHP 8.1.0 Release Announcement" (https://www.php.net/releases/8.1/en.php). *PHP.net*. Retrieved 2024-01-05.
90. "PHP 8 ChangeLog" (https://www.php.net/ChangeLog-8.php#8.2.0). *PHP.net*. Retrieved 2024-01-05.
91. "PHP: PHP 8.2.0 Release Announcement" (https://www.php.net/releases/8.2/en.php). *PHP.net*. Retrieved 2024-01-05.
92. "Unsupported Branches" (https://php.net/eol.php). *php.net*. Retrieved 2019-07-31.
93. "PHP 4.0.0 Released" (https://news-web.php.net/php.announce/22). Retrieved 25 October 2020.
94. "PHP: PHP 4 ChangeLog" (https://www.php.net/ChangeLog-4.php). The PHP Group. 2008-01-03. Retrieved 2008-02-22.
95. "PHP 4.1.0 Release Announcement" (https://www.php.net/releases/4_1_0.php). Retrieved 25 October 2020.
96. "PHP 4.2.0 Release Announcement" (https://www.php.net/releases/4_2_0.php). Retrieved 25 October 2020.
97. "PHP 4.3.0 Release Announcement" (https://www.php.net/releases/4_3_0.php). Retrieved 25 October 2020.
98. "Using PHP from the command line" (https://php.net/manual/en/features.commandline.php). *PHP Manual*. The PHP Group. Retrieved 2009-09-11.
99. "PHP 4.4.0 Release Announcement" (https://www.php.net/releases/4_4_0.php). Retrieved 25 October 2020.
100. "PHP 4.4.0 Release Announcement" (https://php.net/releases/4_4_0.php). *PHP Manual*. The PHP Group. Retrieved 2013-11-24.
101. "PHP 5.0.0 Released!" (https://news-web.php.net/php.announce/50). Retrieved 25 October 2020.
102. "PHP: PHP 5 ChangeLog" (https://www.php.net/ChangeLog-5.php). The PHP Group. 2007-11-08. Retrieved 2008-02-22.
103. "PHP 5.1.0 Release Announcement" (https://www.php.net/releases/5_1_0.php). Retrieved 25 October 2020.
104. "PHP manual: PDO" (https://php.net/manual/en/intro.pdo.php). The PHP Group. 2011-11-15. Retrieved 2011-11-15.
105. "PHP 5.2.0 Release Announcement" (https://www.php.net/releases/5_2_0.php). Retrieved 25 October 2020.

106. "PHP 5.3.0 Release Announcement" (https://www.php.net/releases/5_3_0.php). Retrieved 25 October 2020.
107. "PHP 5.4.0 Release Announcement" (https://www.php.net/releases/5_4_0.php). Retrieved 25 October 2020.
108. "Built-in web server" (https://php.net/manual/en/features.commandline.webserver.php). Retrieved March 26, 2012.
109. "PHP 5.5.0 Release Announcement" (https://www.php.net/releases/5_5_0.php). Retrieved 25 October 2020.
110. "Supported Versions" (https://php.net/supported-versions.php). *php.net*. Retrieved 2017-12-13.
111. "PHP 5.5.0 changes" (https://php.net/manual/en/migration55.new-features.php). *php.net*. Retrieved 2015-03-03.
112. "PHP 5.6.0 Release Announcement" (https://www.php.net/releases/5_6_0.php). Retrieved 25 October 2020.
113. "Migrating from PHP 5.5.x to PHP 5.6.x" (https://www.php.net/manual/en/migration56.new-features.php). *php.net*. Retrieved 2014-03-24.
114. "Resetting PHP 6" (https://lwn.net/Articles/379909/). "There have been books on the shelves purporting to cover PHP 6 since at least 2008. But, in March 2010, the PHP 6 release is not out – in fact, it is not even close to out. Recent events suggest that PHP 6 will not be released before 2011 – if, indeed, it is released at all."
115. Krill, Paul (2014-10-31). "PHP 7 moves full speed ahead" (http://www.infoworld.com/article/2841561/php/php-7-moves-full-speed-ahead.html). *InfoWorld*. "Recent versions of PHP have been part of the 5.x release series, but there will be no PHP 6. "We're going to skip [version] 6, because years ago, we had plans for a 6, but those plans were very different from what we're doing now," Gutmans said. Going right to version 7 avoids confusion."
116. "News Archive – 2018: PHP 7.2.9 Released" (https://php.net/archive/2018.php#id2018-07-19-2). *php.net*. 2018-08-16. Retrieved 2018-08-16.
117. "PHP: rfc:size_t_and_int64_next" (https://wiki.php.net/rfc/size_t_and_int64_next). *php.net*. Retrieved 16 December 2014.
118. "PHP: rfc:abstract_syntax_tree" (https://wiki.php.net/rfc/abstract_syntax_tree). *php.net*. Retrieved 16 December 2014.
119. "PHP: rfc:closure_apply" (https://wiki.php.net/rfc/closure_apply). *php.net*. Retrieved 16 December 2014.
120. "PHP: rfc:integer_semantics" (https://wiki.php.net/rfc/integer_semantics). *php.net*. Retrieved 16 December 2014.
121. "PHP: rfc:isset_ternary" (https://wiki.php.net/rfc/isset_ternary). *php.net*. Retrieved 16 December 2014.
122. "RFC: Unicode Codepoint Escape Syntax" (https://wiki.php.net/rfc/unicode_escape). 2014-11-24. Retrieved 2014-12-19.
123. "Combined Comparison (Spaceship) Operator" (https://wiki.php.net/rfc/combined-comparison-operator). *php.net*. Retrieved 2015-05-21.
124. "PHP RFC: Generator Delegation" (https://wiki.php.net/rfc/generator-delegation). *php.net*. Retrieved 2015-05-21.
125. "PHP RFC: Anonymous Classes" (https://wiki.php.net/rfc/anonymous_classes). *php.net*. Retrieved 2015-05-21.
126. "PHP RFC: Easy User-land CSPRNG" (https://wiki.php.net/rfc/easy_userland_csprng). *php.net*. Retrieved 2015-05-21.
127. "PHP RFC: Group Use Declarations" (https://wiki.php.net/rfc/group_use_declarations). *php.net*. Retrieved 2015-05-21.
128. "PHP: rfc:iterable" (https://wiki.php.net/rfc/iterable). *php.net*. 2016-06-10. Retrieved 2023-06-30.
129. "PHP: rfc:nullable_types" (https://wiki.php.net/rfc/nullable_types). *php.net*. 2014-04-10. Retrieved 2023-06-30.

130. "PHP: rfc:void_return_type" (http://wiki.php.net/rfc/void_return_type). *php.net*. 2015-11-09. Retrieved 2015-11-14.
131. "PHP: rfc:class_constant_visibility" (https://wiki.php.net/rfc/class_const_visibility). *php.net*. 2015-10-27. Retrieved 2015-12-08.
132. "PHP: rfc:short_list_syntax" (https://wiki.php.net/rfc/short_list_syntax). *php.net*. 2016-04-07. Retrieved 2023-06-30.
133. "PHP: rfc:multiple-catch" (https://wiki.php.net/rfc/multiple-catch). *php.net*. 2016-03-06. Retrieved 2023-06-30.
134. "PHP: rfc:object-typehint" (https://wiki.php.net/rfc/object-typehint). *wiki.php.net*.
135. "PHP: rfc:libsodium" (https://wiki.php.net/rfc/libsodium). *wiki.php.net*.
136. "PHP: rfc:allow-abstract-function-override" (https://wiki.php.net/rfc/allow-abstract-function-override). *wiki.php.net*.
137. "PHP: rfc:parameter-no-type-variance" (https://wiki.php.net/rfc/parameter-no-type-variance). *wiki.php.net*.
138. "PHP: todo:php73" (https://wiki.php.net/todo/php73). *wiki.php.net*.
139. "PHP: rfc:flexible_heredoc_nowdoc_syntaxes" (https://wiki.php.net/rfc/flexible_heredoc_nowdoc_syntaxes). *wiki.php.net*.
140. "PHP: rfc:list_reference_assignment" (https://wiki.php.net/rfc/list_reference_assignment). *wiki.php.net*.
141. "PHP: rfc:pcre2-migration" (https://wiki.php.net/rfc/pcre2-migration). *wiki.php.net*.
142. "PHP: hrtime – Manual" (https://php.net/manual/en/function.hrtime.php). *php.net*.
143. "PHP 7.4.0 Released!" (https://www.php.net/archive/2019.php#2019-11-28-1). *php.net*. Retrieved 2019-11-28.
144. "PHP: rfc:typed_properties_v2" (https://wiki.php.net/rfc/typed_properties_v2). *wiki.php.net*. Retrieved 2019-04-04.
145. "PHP: rfc:preload" (https://wiki.php.net/rfc/preload). *wiki.php.net*. Retrieved 2019-04-04.
146. "PHP: rfc:null_coalesce_equal_operator" (https://wiki.php.net/rfc/null_coalesce_equal_operator). *wiki.php.net*. Retrieved 2019-04-04.
147. "PHP: rfc:improve-openssl-random-pseudo-bytes" (https://wiki.php.net/rfc/improve-openssl-random-pseudo-bytes). *wiki.php.net*. Retrieved 2019-04-04.
148. "PHP: rfc:ffi" (https://wiki.php.net/rfc/ffi). *wiki.php.net*. Retrieved 2019-04-05.
149. "PHP: rfc:permanent_hash_ext" (https://wiki.php.net/rfc/permanent_hash_ext). *wiki.php.net*. Retrieved 2019-04-05.
150. "PHP: rfc:password_registry" (https://wiki.php.net/rfc/password_registry). *wiki.php.net*. Retrieved 2019-04-05.
151. "PHP: rfc:mb_str_split" (https://wiki.php.net/rfc/mb_str_split). *wiki.php.net*. Retrieved 2019-04-05.
152. "PHP: rfc:reference_reflection" (https://wiki.php.net/rfc/reference_reflection). *wiki.php.net*. Retrieved 2019-04-05.
153. "PHP: rfc:deprecate-and-remove-ext-wddx" (https://wiki.php.net/rfc/deprecate-and-remove-ext-wddx). *wiki.php.net*. Retrieved 2019-04-05.
154. "PHP: rfc:custom_object_serialization" (https://wiki.php.net/rfc/custom_object_serialization). *wiki.php.net*. Retrieved 2019-04-05.
155. "PHP: Supported Versions" (https://www.php.net/supported-versions.php). *php.net*. Retrieved 2023-11-26.
156. "PHP: rfc:negative_array_index" (https://wiki.php.net/rfc/negative_array_index). *wiki.php.net*. Retrieved 2019-04-05.
157. "PHP RFC: Validation for abstract trait methods" (https://wiki.php.net/rfc/abstract_trait_method_validation). *wiki.php.net*. Retrieved 14 August 2020.
158. "PHP RFC: Saner string to number comparisons" (https://wiki.php.net/rfc/string_to_number_comparison). *wiki.php.net*. Retrieved 14 August 2020.

159. "PHP RFC: Saner numeric strings" (https://wiki.php.net/rfc/saner-numeric-strings). *wiki.php.net*. Retrieved 14 August 2020.
160. "PHP RFC: Stricter type checks for arithmetic/bitwise operators" (https://wiki.php.net/rfc/arithmetic_operator_type_checks). *wiki.php.net*. Retrieved 14 August 2020.
161. "PHP RFC: Reclassifying engine warnings" (https://wiki.php.net/rfc/engine_warnings). *wiki.php.net*. Retrieved 14 August 2020.
162. "PHP: rfc:consistent_type_errors" (https://wiki.php.net/rfc/consistent_type_errors). *wiki.php.net*. Retrieved 2019-04-05.
163. "PHP: rfc:lsp_errors" (https://wiki.php.net/rfc/lsp_errors). *wiki.php.net*. Retrieved 2019-05-26.
164. "PHP RFC: Locale-independent float to string cast" (https://wiki.php.net/rfc/locale_independent_float_to_string). *wiki.php.net*. Retrieved 14 August 2020.
165. "PHP RFC: Variable Syntax Tweaks" (https://wiki.php.net/rfc/variable_syntax_tweaks). *wiki.php.net*. Retrieved 14 August 2020.
166. "PHP RFC: Attributes V2" (https://wiki.php.net/rfc/attributes_v2). *wiki.php.net*. Retrieved 14 August 2020.
167. "PHP RFC: Attribute Amendments" (https://wiki.php.net/rfc/attribute_amendments). *wiki.php.net*. Retrieved 14 August 2020.
168. "PHP RFC: Shorter Attribute Syntax" (https://wiki.php.net/rfc/shorter_attribute_syntax). *wiki.php.net*. Retrieved 2020-06-20.
169. "PHP RFC: Shorter Attribute Syntax Change" (https://wiki.php.net/rfc/shorter_attribute_syntax_change). *wiki.php.net*. Retrieved 14 August 2020.
170. "PHP RFC: Named Arguments" (https://wiki.php.net/rfc/named_params). *wiki.php.net*. Retrieved 14 August 2020.
171. "PHP RFC: Match expression v2" (https://wiki.php.net/rfc/match_expression_v2). *wiki.php.net*. Retrieved 14 August 2020.
172. "PHP RFC: Constructor Property Promotion" (https://wiki.php.net/rfc/constructor_promotion). *wiki.php.net*. Retrieved 14 August 2020.
173. "PHP RFC: Union Types 2.0" (https://wiki.php.net/rfc/union_types_v2). *wiki.php.net*. Retrieved 14 August 2020.
174. "PHP RFC: Mixed Type v2" (https://wiki.php.net/rfc/mixed_type_v2). *wiki.php.net*. Retrieved 14 August 2020.
175. "PHP RFC: Static return type" (https://wiki.php.net/rfc/static_return_type). *wiki.php.net*. Retrieved 14 August 2020.
176. "PHP RFC: non-capturing catches" (https://wiki.php.net/rfc/non-capturing_catches). *wiki.php.net*. Retrieved 14 August 2020.
177. Andre, Tyson. "PHP RFC: Always available JSON extension" (https://wiki.php.net/rfc/always_enable_json). *PHP*. Retrieved 25 October 2020.
178. "PHP: todo:php81" (https://wiki.php.net/todo/php81). *wiki.php.net*. Retrieved 2022-06-16.
179. "PHP RFC: Explicit octal integer literal notation" (https://wiki.php.net/rfc/explicit_octal_notation). *wiki.php.net*. Retrieved 2020-11-25.
180. "PHP RFC: Enumerations" (https://wiki.php.net/rfc/enumerations). *wiki.php.net*. Retrieved 2021-03-25.
181. "PHP: rfc:readonly_properties_v2" (https://wiki.php.net/rfc/readonly_properties_v2). *wiki.php.net*. Retrieved 2021-11-26.
182. "PHP: rfc:first_class_callable_syntax" (https://wiki.php.net/rfc/first_class_callable_syntax). *wiki.php.net*. Retrieved 2021-11-26.
183. "PHP: rfc:new_in_initializers" (https://wiki.php.net/rfc/new_in_initializers). *wiki.php.net*. Retrieved 2021-11-26.
184. "PHP: rfc:pure-intersection-types" (https://wiki.php.net/rfc/pure-intersection-types). *wiki.php.net*. Retrieved 2021-11-26.
185. "PHP: rfc:noreturn_type" (https://wiki.php.net/rfc/noreturn_type). *wiki.php.net*. Retrieved 2021-11-26.

186. "PHP: rfc:final_class_const" (https://wiki.php.net/rfc/final_class_const). *wiki.php.net*. Retrieved 2021-11-26.
187. "PHP: rfc:fibers" (https://wiki.php.net/rfc/fibers). *wiki.php.net*. Retrieved 2021-11-26.
188. "PHP: todo:php82" (https://wiki.php.net/todo/php82). *wiki.php.net*. Retrieved 2022-06-16.
189. "PHP: rfc:readonly_classes" (https://wiki.php.net/rfc/readonly_classes). *wiki.php.net*. Retrieved 2022-06-16.
190. "PHP: rfc:null-false-standalone-types" (https://wiki.php.net/rfc/null-false-standalone-types). *wiki.php.net*. Retrieved 2022-06-16.
191. "PHP: rfc:true-type" (https://wiki.php.net/rfc/true-type). *wiki.php.net*. Retrieved 2022-06-16.
192. "PHP: rfc:strtolower-ascii" (https://wiki.php.net/rfc/strtolower-ascii). *wiki.php.net*. Retrieved 2022-06-16.
193. "PHP: rfc:dnf_types" (https://wiki.php.net/rfc/dnf_types). *wiki.php.net*. Retrieved 2023-02-07.
194. "PHP: rfc:constants_in_traits" (https://wiki.php.net/rfc/constants_in_traits). *wiki.php.net*. Retrieved 2023-02-07.
195. "PHP 8.3.0 Released!" (https://www.php.net/archive/2023.php#2023-11-23-2). *php.net*. 23 November 2023. Retrieved 24 November 2023.
196. "PHP: rfc:typed_class_constants" (https://wiki.php.net/rfc/typed_class_constants). *wiki.php.net*. Retrieved 2023-12-17.
197. "PHP: rfc:dynamic_class_constant_fetch" (https://wiki.php.net/rfc/dynamic_class_constant_fetch). *wiki.php.net*. Retrieved 2023-12-17.
198. "PHP: rfc:marking_overriden_methods" (https://wiki.php.net/rfc/marking_overriden_methods). *wiki.php.net*. Retrieved 2023-12-17.
199. "PHP: rfc:readonly_amendments" (https://wiki.php.net/rfc/readonly_amendments). *wiki.php.net*. Retrieved 2023-12-17.
200. "PHP: rfc:json_validate" (https://wiki.php.net/rfc/json_validate). *wiki.php.net*. Retrieved 2023-12-17.
201. "PHP: rfc:randomizer_additions" (https://wiki.php.net/rfc/randomizer_additions). *wiki.php.net*. Retrieved 2023-12-17.
202. "PHP: todo: php84" (https://wiki.php.net/todo/php84). *php.net*. 26 March 2024. Retrieved 26 March 2024.
203. "PHP: ElePHPant" (https://php.net/elephpant.php). 4 Oct 2014. Retrieved 4 Oct 2014.
204. "Redirecting…" (https://wwphp-fb.github.io/faq/community/elephpant/). *wwphp-fb.github.io*.
205. "The PHP Mascot's Birth – Creator Of The elePHPant Vincent Pontier Reveals The True Story!" (https://7php.com/elephpant/). *7php.com*. 2014-01-06.
206. "ElePHPant" (https://docs.php.earth/php/community/elephpant/). *PHP.earth*. Retrieved 2024-02-13.
207. "PHP: ElePHPant" (https://www.php.net/elephpant.php). *www.php.net*.
208. "A Field Guide to Elephpants" (https://afieldguidetoelephpants.net/). *afieldguidetoelephpants.net*.
209. "tags – Manual" (https://www.php.net/manual/en/language.basic-syntax.phptags.php). *php.net*. Retrieved 2014-02-17.
210. "PHP: rfc:shortags" (http://wiki.php.net/rfc/shortags). *php.net*. 2008-04-03. Retrieved 2014-05-08.
211. "PHP: Basic syntax" (https://www.php.net/manual/en/language.basic-syntax.php). The PHP Group. Retrieved 2008-02-22.
212. "Basic Coding Standard" (https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-1-basic-coding-standard.md). PHP Framework Interoperability Group. Retrieved 2016-01-03.
213. "echo – Manual" (https://www.php.net/echo). *php.net*. Retrieved 2014-02-17.
214. "Description of core php.ini directives – Manual" (https://www.php.net/manual/en/ini.core.php#ini.short-open-tag). *php.net*. 2002-03-17. Retrieved 2014-02-17.

215. "Your first PHP-enabled page" (https://www.php.net/manual/en/tutorial.firstpage.php). The PHP Group. Retrieved 2008-02-25.
216. Bray, Tim; et al. (26 November 2008). "Processing Instructions" (http://www.w3.org/TR/2008/REC-xml-20081126/#sec-pi). *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. W3C. Retrieved 2009-06-18.
217. "Variables" (https://www.php.net/manual/en/language.variables.php). The PHP Group. Retrieved 2008-03-16.
218. "Instruction separation" (https://www.php.net/basic-syntax.instruction-separation). The PHP Group. Retrieved 2008-03-16.
219. "Comments" (https://www.php.net/manual/en/language.basic-syntax.comments.php). The PHP Group. Retrieved 2008-03-16.
220. "Integers in PHP, running with scissors, and portability" (http://www.mysqlperformanceblog.com/2007/03/27/integers-in-php-running-with-scissors-and-portability/). MySQL Performance Blog. March 27, 2007. Retrieved 2007-03-28.
221. "Types" (https://www.php.net/manual/en/language.types.php). The PHP Group. Retrieved 2008-03-16.
222. "Strings" (https://www.php.net/manual/en/language.types.string.php). The PHP Group. Retrieved 2008-03-21.
223. "SPL – StandardPHPLibrary" (https://www.php.net/spl). *PHP.net*. March 16, 2009. Retrieved 2009-03-16.
224. "User-defined functions (PHP manual)" (https://www.php.net/manual/en/functions.user-defined.php). *php.net*. 2014-07-04. Retrieved 2014-07-07.
225. "Variable functions (PHP manual)" (https://www.php.net/manual/en/functions.variable-functions.php). *php.net*. 2014-07-04. Retrieved 2014-07-07.
226. "create_function() (PHP manual)" (https://www.php.net/manual/en/function.create-function.php). *php.net*. 2022-04-06. Retrieved 2022-05-04.
227. "Anonymous functions (PHP manual)" (https://www.php.net/manual/en/functions.anonymous.php). *php.net*. 2014-07-04. Retrieved 2014-07-07.
228. "Arrow Functions (PHP manual)" (https://www.php.net/manual/en/functions.arrow.php). *php.net*. Retrieved 2021-01-25.
229. Christian Seiler; Dmitry Stogov (2008-07-01). "Request for Comments: Lambda functions and closures" (http://wiki.php.net/rfc/closures). *php.net*. Retrieved 2014-07-07.
230. "PHP 5 Object References" (http://mjtsai.com/blog/2004/07/15/php-5-object-references/). *mjtsai.com*. Retrieved 2008-03-16.
231. "Classes and Objects (PHP 5)" (http://www.php.net/zend-engine-2.php). The PHP Group. Retrieved 2008-03-16.
232. "Object cloning" (https://www.php.net/language.oop5.cloning). The PHP Group. Retrieved 2008-03-16.
233. "Visibility (PHP Manual)" (https://web.archive.org/web/20100924033414/http://theserverpages.com/php/manual/en/language.oop5.visibility.php). *theserverpages.com*. 2005-05-19. Archived from the original (http://theserverpages.com/php/manual/en/language.oop5.visibility.php) on 2010-09-24. Retrieved 2010-08-26.
234. "How do computer languages work?" (https://web.archive.org/web/20110716214917/http://www.linux-tutorial.info/modules.php?name=Howto&pagename=Unix-and-Internet-Fundamentals-HOWTO/languages.html). Archived from the original (http://www.linux-tutorial.info/modules.php?name=Howto&pagename=Unix-and-Internet-Fundamentals-HOWTO/languages.html) on 2011-07-16. Retrieved 2009-11-04.
235. Gilmore, W. Jason (2006-01-23). *Beginning PHP and MySQL 5: From Novice to Professional* (https://archive.org/details/beginningphpmysq0000gilm/page/43). Apress. p. 43 (https://archive.org/details/beginningphpmysq0000gilm/page/43). ISBN 1-59059-552-1.
236. Julien Pauli; Nikita Popov; Anthony Ferrara. "PHP Internals Book" (https://www.phpinternalsbook.com). *PHP Internals Book*. Archived (https://web.archive.org/web/20250121000000/https://www.phpinternalsbook.com) from the original on 2025-01-21. Retrieved 2025-01-21.

237. "[VOTE] Integrating Zend Optimizer+ into the PHP distribution" (http://news.php.net/php.internals/66531). *news.php.net*. Retrieved 2013-03-08.
238. "Alternative PHP Cache" (https://web.archive.org/web/20131115071936/http://php.net/manual/en/book.apc.php). *PHP.net*. Archived from the original (http://www.php.net/manual/en/book.apc.php) on 2013-11-15. Retrieved 2013-09-21.
239. "We are the 98.5% (and the 16%) « HipHop Virtual Machine" (http://www.hhvm.com/blog/2813/we-are-the-98-5-and-the-16). *hhvm.com*. December 2013. Retrieved 2014-02-23.
240. "The Definitive PHP 5.6, 7.0, 7.1, 7.2 & 7.3 Benchmarks (2019)" (https://kinsta.com/blog/php-benchmarks/). 2019-01-14. Retrieved 2019-04-19.
241. Krill, Paul (2017-09-20). "Forget PHP! Facebook's HHVM engine switches to Hack instead" (https://www.infoworld.com/article/3226489/web-development/forget-php-facebooks-hhvm-engine-switches-to-hack-instead.html). *InfoWorld*. Retrieved 2019-02-06.
242. "Announcement on GitHub removing HPHPc support" (https://github.com/facebook/hiphop-php/commit/fc5b95110ff75110ad55bb97f7c93a8c4eb68e3b). *GitHub*. Retrieved 2013-05-24.
243. "The PHP License, version 3.01" (https://www.php.net/license/3_01.txt). Retrieved 2010-05-20.
244. "GPL-Incompatible, Free Software Licenses" (https://www.gnu.org/licenses/license-list.html#GPLIncompatibleLicenses). *Various Licenses and Comments about Them*. Free Software Foundation. Retrieved 2011-01-03.
245. "PHP: Function and Method listing – Manual" (https://php.net/manual/en/indexes.functions.php). The PHP Group. Retrieved 2015-01-14.
246. "Introduction – Manual" (http://www.php.net/manual/en/intro.pdo.php). *php.net*. 2013-06-07. Retrieved 2013-06-13.
247. Darryl Patterson (5 August 2004). "Simplify Business Logic with PHP DataObjects — O'Reilly Media" (https://web.archive.org/web/20141216140653/http://www.onlamp.com/pub/a/php/2004/08/05/dataobjects.html). *ibm.com*. Archived from the original (http://www.onlamp.com/pub/a/php/2004/08/05/dataobjects.html) on 16 December 2014. Retrieved 16 December 2014.
248. "IBM — United States" (http://www-128.ibm.com/developerworks/db2/library/techarticle/dm-0612xia/). Retrieved 16 December 2014.
249. "Five common PHP database problems" (http://www-128.ibm.com/developerworks/library/os-php-dbmistake/index.html). *ibm.com*. 2006-08-01. Retrieved 2013-06-13.
250. "IBM Redbooks — Developing PHP Applications for IBM Data Servers" (http://www.redbooks.ibm.com/abstracts/sg247218.html). *redbooks.ibm.com*. Retrieved 16 December 2014.
251. "php[architect] Magazine – The Journal for PHP Programmers" (http://www.phparch.com/issue.php?mid=65). *www.phparch.com*.
252. Krill, Paul (19 October 2005). "PHP catching on at enterprises, vying with Java" (https://web.archive.org/web/20140713004345/http://www.infoworld.com/d/developer-world/php-catching-enterprises-vying-java-708). *InfoWorld*. Archived from the original (http://www.infoworld.com/article/05/10/19/HNphpshow_1.html) on 13 July 2014.
253. "Cross Reference: /PHP_5_4/ext/standard/" (https://web.archive.org/web/20120316010914/http://lxr.php.net/xref/PHP_5_4/ext/standard/). *php.net*. Archived from the original (http://lxr.php.net/xref/PHP_5_4/ext/standard/) on 16 March 2012. Retrieved 16 December 2014.
254. "Developing Custom PHP Extensions" (https://web.archive.org/web/20080218045752/http://www.devnewz.com/090902b.html). *devnewz.com*. 2002-09-09. Archived from the original (http://www.devnewz.com/090902b.html) on 2008-02-18. Retrieved 2008-02-25.
255. "Why Zephir?" (https://docs.zephir-lang.com/en/latest/motivation.html). *zephir-lang.com*. 2015-10-20. Retrieved 2015-12-14.
256. "PHP Credits" (https://php.net/credits). Retrieved 2018-12-16.
257. "Learn PHP Via PHP Training and PHP Certification" (https://www.zend.com/training/php). *www.zend.com*. Retrieved 2020-11-16.
258. Walker, James (2021-12-13). "What the New PHP Foundation Means for PHP's Future" (https://www.howtogeek.com/devops/what-the-new-php-foundation-means-for-phps-future/). *How-To Geek*. Retrieved 2023-11-26.

259. "The New Life of PHP – The PHP Foundation | The PhpStorm Blog" (https://blog.jetbrains.com/phpstorm/2021/11/the-php-foundation/). *The JetBrains Blog*. 22 November 2021. Retrieved 2022-06-16.
260. "The PHP Foundation: Impact and Transparency Report 2022" (https://thephp.foundation/blog/2022/11/22/transparency-and-impact-report-2022/). *thephp.foundation*. Retrieved 2023-11-27.
261. Pronskiy, Roman (2024-02-26). "The PHP Foundation: Impact and Transparency Report 2023" (https://thephp.foundation/blog/2024/02/26/transparency-and-impact-report-2023/). *The PHP Foundation*. Retrieved 2024-04-01.
262. Anderson, Tim. "PHP Foundation formed to fund core developers" (https://www.theregister.com/2021/11/23/php_foundation_formed_to_fund/). *www.theregister.com*. Retrieved 2023-12-05.
263. "Programming languages: This old favourite is gaining popularity again" (https://www.zdnet.com/article/programming-languages-this-old-favourite-is-gaining-popularity-again/). *ZDNET*. Retrieved 2023-12-05.
264. "PHP 8.1 Released With Enums, Read-Only Properties and Fibers" (https://www.i-programmer.info/news/98-languages/15050-php-81-released.html). *www.i-programmer.info*. Retrieved 2023-12-05.
265. "It's time for the PHP Foundation to #StopBreakingPHP" (https://trongate.io/news/display/TWN4GR/its-time-for-the-php-foundation-to-stopbreakingphp). *trongate.io*. Retrieved 2023-11-27.
266. "WordPress 6.4 PHP Compatibility" (https://make.wordpress.org/hosting/2023/11/16/wordpress-6-4-php-compatibility/). *Make WordPress Hosting*. 2023-11-16. Retrieved 2023-11-27.
267. "PHP" (https://www.sovereigntechfund.de/tech/php). *Sovereign Tech Fund*. Retrieved 2024-05-26.
268. "General Installation Considerations" (http://www.php.net/manual/en/install.general.php). *php.net*. Retrieved 2013-09-22.
269. "News Archive: PHP 5.3.3 Released!" (http://www.php.net/archive/2010.php#id2010-07-22-2). *php.net*. 2010-07-22.
270. "FastCGI Process Manager (FPM)" (http://www.php.net/manual/en/install.fpm.php). *php.net*. Retrieved 2013-09-22.
271. "Command line usage: Introduction" (http://www.php.net/manual/en/features.commandline.introduction.php). *php.net*. Retrieved 2013-09-22.
272. "Command line usage: Differences to other SAPIs" (http://www.php.net/manual/en/features.commandline.differences.php). *php.net*. Retrieved 2013-09-22.
273. "General Installation Considerations" (https://php.net/manual/en/install.general.php). *php.net*. Retrieved 2013-09-22.
274. "PHP: Apache 2.x on Microsoft Windows" (https://web.archive.org/web/20130926122011/http://php.net/manual/en/install.windows.apache2.php). *php.net*. Archived from the original (https://php.net/manual/en/install.windows.apache2.php) on 2013-09-26. Retrieved 2013-09-22.
275. "Command line usage: Introduction" (http://www.php.net/manual/en/features.commandline.introduction.php). *php.net*. Retrieved 2013-09-22.
276. "Installing PHP-GTK 2" (https://web.archive.org/web/20131212093441/http://gtk.php.net/manual/en/tutorials.installation.php). *php.net*. Archived from the original (http://gtk.php.net/manual/en/tutorials.installation.php) on 2013-12-12. Retrieved 2013-09-22.
277. "AWS SDK for PHP" (http://aws.amazon.com/sdkforphp/). *aws.amazon.com*. Retrieved 2014-03-06.
278. "Windows Azure SDK for PHP — Interoperability Bridges and Labs Center" (https://web.archive.org/web/20140320152702/http://www.interoperabilitybridges.com/projects/php-sdk-for-windows-azure.aspx). *interoperabilitybridges.com*. Archived from the original (http://www.interoperabilitybridges.com/projects/php-sdk-for-windows-azure.aspx) on 2014-03-20. Retrieved 2014-03-06.
279. "Runtime configuration: Table of contents" (http://www.php.net/manual/en/configuration.php). *php.net*. Retrieved 2013-09-22.
280. "php.ini directives: List of php.ini directives" (http://www.php.net/manual/en/ini.list.php). *php.net*. Retrieved 2013-09-22.

281. "Runtime configuration: The configuration file" (http://www.php.net/manual/en/configuration.file.php). PHP.net. Retrieved 2013-09-22.
282. "php.ini directives: List of php.ini sections" (http://www.php.net/manual/en/ini.sections.php). PHP.net. Retrieved 2013-09-22.
283. "Runtime configuration: Where a configuration setting may be set" (http://www.php.net/manual/en/configuration.changes.modes.php). PHP.net. Retrieved 2013-09-22.
284. "PHP Manual Image Processing and GD;" (https://php.net/manual/en/book.image.php). php.net. Retrieved 2011-04-09.
285. "PHP Server-Side Scripting Language" (https://web.archive.org/web/20160121223739/http://webmaster.iu.edu/tools-and-guides/programming-languages/php.phtml). Indiana University. 2007-04-04. Archived from the original (http://webmaster.iu.edu/PHPlanguage/index.shtml) on 2016-01-21. Retrieved 2008-02-25.
286. "JavaServer Pages Technology — JavaServer Pages Comparing Methods for Server-Side Dynamic Content White Paper" (http://java.sun.com/products/jsp/jspservlet.html). Sun Microsystems. Retrieved 2008-02-25.
287. "Five simple ways to tune your LAMP application" (http://www.ibm.com/developerworks/library/os-5waystunelamp/index.html). *IBM*. 2011-01-25.
288. "PHP at the core: Extension structure" (https://web.archive.org/web/20130926082102/http://www.php.net/manual/en/internals2.structure.php). *PHP.net*. Archived from the original (http://www.php.net/manual/en/internals2.structure.php) on 2013-09-26. Retrieved 2013-09-22.
289. "PHP at the core: The "counter" Extension – A Continuing Example" (https://web.archive.org/web/20130926082106/http://www.php.net/manual/en/internals2.counter.php). *PHP.net*. Archived from the original (http://www.php.net/manual/en/internals2.counter.php) on 2013-09-26. Retrieved 2013-09-22.
290. "Extension Writing Part I: Introduction to PHP and Zend" (https://web.archive.org/web/20130924233638/http://devzone.zend.com/303/extension-writing-part-i-introduction-to-php-and-zend/). Zend Technologies. 2005-03-01. Archived from the original (http://devzone.zend.com/303/extension-writing-part-i-introduction-to-php-and-zend/) on 2013-09-24. Retrieved 2013-09-22.
291. "Extension Writing Part II: Parameters, Arrays, and ZVALs" (https://web.archive.org/web/20130926091658/http://devzone.zend.com/317/extension-writing-part-ii-parameters-arrays-and-zvals/). Zend Technologies. 2005-06-06. Archived from the original (http://devzone.zend.com/317/extension-writing-part-ii-parameters-arrays-and-zvals/) on 2013-09-26. Retrieved 2013-09-22.
292. "Extension Writing Part II: Parameters, Arrays, and ZVALs (continued)" (https://web.archive.org/web/20130926091655/http://devzone.zend.com/318/extension-writing-part-ii-parameters-arrays-and-zvals-continued/). Zend Technologies. 2005-06-06. Archived from the original (http://devzone.zend.com/318/extension-writing-part-ii-parameters-arrays-and-zvals-continued/) on 2013-09-26. Retrieved 2013-09-22.
293. "Extension Writing Part III: Resources" (https://web.archive.org/web/20130926091645/http://devzone.zend.com/446/extension-writing-part-iii-resources/). Zend Technologies. 2006-05-12. Archived from the original (http://devzone.zend.com/446/extension-writing-part-iii-resources/) on 2013-09-26. Retrieved 2013-09-22.
294. "Wrapping C++ Classes in a PHP Extension" (https://web.archive.org/web/20130920011549/http://devzone.zend.com/1435/wrapping-c-classes-in-a-php-extension/). Zend Technologies. 2009-04-22. Archived from the original (http://devzone.zend.com/1435/wrapping-c-classes-in-a-php-extension/) on 2013-09-20. Retrieved 2013-09-22.
295. "Extending PHP with C++?" (https://stackoverflow.com/q/1110682). Stack Overflow. Retrieved 2013-09-22.
296. "How can I use C++ code to interact with PHP?" (https://stackoverflow.com/q/1502244). Stack Overflow. Retrieved 2013-09-22.
297. Golemon, Sara (2006). *Extending and Embedding PHP*. Sams. ISBN 978-0-672-32704-9.
298. "Request #46919: Multithreading" (https://bugs.php.net/bug.php?id=46919). *PHP.net*. Retrieved 2013-09-22.
299. "pthreads: Introduction (PHP Manual)" (http://www.php.net/manual/en/intro.pthreads.php). *PHP.net*. Retrieved 2013-09-22.

300. "PECL :: Package :: pthreads" (https://pecl.php.net/package/pthreads). *pecl.php.net*. Retrieved 2014-02-09.
301. "Manual:Installation requirements#PHP" (https://www.mediawiki.org/w/index.php?title=Manual:Installation_requirements&oldid=299556#PHP). MediaWiki. 2010-01-25. Retrieved 2010-02-26. "PHP is the programming language in which MediaWiki is written [...]"
302. "About WordPress" (http://wordpress.org/about/). Retrieved 2010-02-26. "WordPress was [...] built on PHP"
303. Kempkens, Alex. "Joomla! — Content Management System to build websites & apps" (http://www.joomla.org/about-joomla.html).
304. "PHP and Drupal" (https://web.archive.org/web/20100208205523/http://drupal.org/node/176052). Drupal.org. 16 September 2007. Archived from the original (https://drupal.org/node/176052) on 2010-02-08. Retrieved 2010-06-13.
305. "About" (https://web.archive.org/web/20100111055644/http://docs.moodle.org/en/About_Moodle). Moodle.org. Archived from the original (http://docs.moodle.org/en/About_Moodle) on 2010-01-11. Retrieved 2009-12-20.
306. "Server requirements of SilverStripe" (https://web.archive.org/web/20141128063118/http://doc.silverstripe.org/framework/en/installation/server-requirements). Archived from the original (http://doc.silverstripe.org/framework/en/installation/server-requirements) on 28 November 2014. Retrieved 13 October 2014. "SilverStripe requires PHP 5.3.2+"
307. Ide, Andy (2013-01-31). "PHP just grows & grows" (http://news.netcraft.com/archives/2013/01/31/php-just-grows-grows.html). Retrieved 2013-04-01.
308. "Usage Statistics and Market Share of PHP Version 4 for Websites, January 2025" (https://w3techs.com/technologies/details/pl-php/4). *w3techs.com*.
309. "Usage Statistics and Market Share of PHP Version 5 for Websites, January 2025" (https://w3techs.com/technologies/details/pl-php/5). *w3techs.com*.
310. "Usage Statistics and Market Share of PHP Version 7 for Websites, January 2025" (https://w3techs.com/technologies/details/pl-php/7). *w3techs.com*.
311. "Usage Statistics and Market Share of PHP Version 8 for Websites, January 2025" (https://w3techs.com/technologies/details/pl-php/8). *w3techs.com*.
312. "National Vulnerability Database (NVD) Search Vulnerabilities Statistics" (https://nvd.nist.gov/vuln/search/statistics?form_type=Basic&results_type=statistics&query=PHP&queryType=phrase&search_type=all). Retrieved 2019-11-22.
313. "PHP-related vulnerabilities on the National Vulnerability Database" (https://web.archive.org/web/20090628173101/http://www.coelho.net/php_cve.html). 2012-07-05. Archived from the original (http://www.coelho.net/php_cve.html) on 2009-06-28. Retrieved 2013-04-01.
314. "Developer Meeting Notes, Nov. 2005" (http://derickrethans.nl/files/meeting-notes.html#sand-boxing-or-taint-mode).
315. "Taint mode decision, November 2007" (https://web.archive.org/web/20090226124957/http://devzone.zend.com/article/2798-Zend-Weekly-Summaries-Issue-368#Heading1). Archived from the original (http://devzone.zend.com/article/2798-Zend-Weekly-Summaries-Issue-368#Heading1) on 2009-02-26.
316. "PHP: rfc:taint" (https://wiki.php.net/rfc/taint). *wiki.php.net*.
317. "Hardened-PHP Project" (https://web.archive.org/web/20190224012812/http://www.hardened-php.net/). 2008-08-15. Archived from the original (http://www.hardened-php.net) on 2019-02-24. Retrieved 2019-08-22.
318. "Snuffleupagus Documentation" (https://snuffleupagus.readthedocs.io/).
319. "Security: Using Register Globals" (https://web.archive.org/web/20130927161000/http://php.net/manual/en/security.globals.php). *PHP Manual*. PHP.net. Archived from the original (https://php.net/manual/en/security.globals.php) on 2013-09-27. Retrieved 2013-09-22.
320. "Magic Quotes" (https://web.archive.org/web/20140208000607/http://www.php.net/manual/en/security.magicquotes.php). *PHP Manual*. PHP.net. Archived from the original (http://www.php.net/manual/en/security.magicquotes.php) on 2014-02-08. Retrieved 2014-01-17.

321. " 'engine' configuration directive" (http://www.php.net/manual/en/apache.configuration.php#ini.engine). *PHP: Runtime Configuration*. PHP.net. Retrieved 2014-02-13.
322. "PHP Security Exploit With GIF Images" (https://web.archive.org/web/20130927162421/http://devzone.zend.com/1008/php-security-exploit-with-gif-images/). 2007-06-22. Archived from the original (http://devzone.zend.com/1008/php-security-exploit-with-gif-images/) on 2013-09-27. Retrieved 2013-09-22.
323. "PHP security exploit with GIF images" (http://www.phpclasses.org/blog/post/67-PHP-security-exploit-with-GIF-images.html). PHP Classes blog. 2007-06-20. Retrieved 2013-09-22.
324. "Passing Malicious PHP Through getimagesize()" (https://web.archive.org/web/20130921222424/http://ha.ckers.org/blog/20070604/passing-malicious-php-through-getimagesize). 2007-06-04. Archived from the original (http://ha.ckers.org/blog/20070604/passing-malicious-php-through-getimagesize/) on 2013-09-21. Retrieved 2013-09-22.
325. " 'enable_dl' configuration directive" (http://www.php.net/manual/en/info.configuration.php#ini.enable-dl). *PHP: Runtime Configuration*. PHP.net. Retrieved 2014-02-13.
326. "PHP function reference: dl()" (https://php.net/manual/en/function.dl.php). PHP.net. Retrieved 2013-09-22.
327. "My host won't fix their Trojan" (http://www.webhostingtalk.com/showthread.php?t=514779). WebHosting Talk. Retrieved 2013-09-22.
328. Raz0r (25 January 2013). "Simple Machines Forum <= 2.0.3 Admin Password Reset" (http://raz0r.name/vulnerabilities/simple-machines-forum/). *Raz0R.name — Web Application Security*.
329. Nibble Security. "TYPO3-SA-2010-020, TYPO3-SA-2010-022 EXPLAINED" (http://blog.nibblesec.org/2010/12/typo3-sa-2010-020-typo3-sa-2010-022.html).
330. "Криптостойкость и небезопасное сравнение" (https://web.archive.org/web/20170704214011/https://ahack.ru/articles/cryptographic-security-and-php-applications.htm). *ahack.ru* (in Russian). Archived from the original on 4 July 2017.
331. Krawczyk, Pawel (2013). "Most common attacks on web applications" (https://web.archive.org/web/20150415150236/https://ipsec.pl/web-application-security/most-common-attacks-web-applications.html). IPSec.pl. Archived from the original (https://ipsec.pl/web-application-security/most-common-attacks-web-applications.html) on 2015-04-15. Retrieved 2015-04-15.
332. Krawczyk, Pawel (2013). "So what are the "most critical" application flaws? On new OWASP Top 10" (https://ipsec.pl/application-security/2013/so-what-are-most-critical-application-flaws-new-owasp-top-10.html). IPSec.pl. Retrieved 2015-04-15.
333. "PHP: Rand – Manual" (https://php.net/rand).
334. "PHP: Mt_rand - Manual" (https://php.net/mt_rand).
335. "PHP: Random_int – Manual" (https://php.net/random_int).
336. Argyros, George; Kiayias, Aggelos (10 August 2012). "I Forgot Your Password: Randomness Attacks Against PHP Applications" (https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/argyros). *usenix.org*. USENIX. Retrieved 19 April 2024.
337. "PHP: RFC:release_cycle_update" (https://wiki.php.net/rfc/release_cycle_update).
338. "PHP Support for PHP 7.2 – 8.0 | PHP LTS | Zend by Perforce" (https://www.zend.com/services/php-long-term-support). Retrieved 2024-05-23.
339. "Pagely PHP Long Term Support Page" (https://pagely.com/solutions/php-long-term-support/). *Pagely*. Retrieved 2024-09-14.

Further reading

- Ford, Paul (June 11, 2015). "What is Code?" (https://www.bloomberg.com/graphics/2015-paul-ford-what-is-code/). *Bloomberg Businessweek*. "What's the Absolute Minimum I Must Know

About PHP?"

External links

- Official website (https://www.php.net)