Premium Managed WordPress Hosting

10 Things Not to Do in PHP 7

By Daniel Pataki • Updated on May 20, 2019

I've already shared some of the upcoming features of PHP 7, in this article I thought I'd take a look at some of the bad patterns we should stop using as we switch to the lightning fast PHP 7. And don't forget to check out our new mega-benchmark of the final version of PHP 7.2.

1. Do Not Use mysql_ Functions

The time has finally come when you won't just be advised to stop using `mysql_` functions. PHP 7 will remove them altogether from core which means you'll need to move to the far better `mysqli_` functions, or the even more flexible PDO implementation.

2. Do Not Write Wasteful Code

This one may be a no-brainer but it will become increasingly important because the speed increases in PHP 7 may hide some of your issues. Don't be content with your site speed simply because the switch to PHP 7 made it faster.

better, take a look at our beginners' guide to speed optimization article.

As developers you should always make sure to load scripts only when they are needed, concatenate them when possible, write efficient database queries, use caching when possible and so on.

3. Do Not Use PHP Close Tags at the End of a File

If you take a look, most core WordPress files omit the ending PHP tag when a file ends with PHP code. In fact, the Zend Framework specifically forbids it. It is not required by PHP and by omitting it at the end of a file you are making sure that no trailing whitespace can be added.

4. Do Not Pass by Reference if Not Needed

I personally don't like passing by reference. I understand that in some cases it is useful, but in many others it makes code harder to understand and follow and especially difficult to predict the result.

Apparently, people think it makes their code faster though which according to respectable PHP programmers is just not true.

One example of why references are bad is PHP built in `shuffle()` or `sort()`. Instead of returning a shuffled or sorted array, they modify the original which is completely illogical to my mind.

Performing database queries in a loop is just wasteful. It puts unnecessary strain on your systems and it is likely you can achieve the same result faster outside the loop. When I bump into a situation where this would be needed I can usually solve the issue with two separate queries I use to build an array of data. I then loop over the array, no need to perform queries in the process.

Due to the way WordPress works there may be some exceptions to this. While `get_post_meta()` will grab a meta value from the database, you can use it in a loop if you're looping through one specific post's metadata. This is because when you use it for the first time WordPress actually retrieves all metadata and caches it. Subsequent calls use the cached data, not database calls.

The best way to work these things out is to read function documentation and to use something like the Query Monitor.

6. Do Not Use * in SQL Queries

All right, this one is more of a MySQL issue, but we tend to write our SQL code in PHP so I say it's fair game. In any case, don't use wildcards in SQL queries if you can avoid them, especially if you have a database with a lot of columns.

Specify the exact columns you need and only retrieve those. This helps minimize your resource usage, protect your data and make things as clear as possible.

While on the subject of SQL, know your available functions and test for speed as much as possible. When calculating averages, sums or similar numbers use SQL functions instead of PHP functions. If you are unsure of the speed of a query test it and try some other variations – use the best one.

It is not wise to trust user input. Always filter, sanitize, escape, check and use fallbacks. There are three issues with user data: we developers don't take every possibility into account, it is frequently incorrect and it may be intentionally malicious.

A well thought out system can protect against all of these. Make sure to use built in functions like `filter_var()` to check for proper values and escaping and other functions when working with databases.

Struggling with downtime and WordPress problems? Kinsta is the hosting solution designed to save you time! Check out our features

WordPress has a bunch of functions to help you out. Take a look at the Validating, escaping and sanitising user data article for more information.

8. Do Not Try to Be Clever

Your goal should be to write elegant code that expresses your intentions the most clearly. You may be able to shave off an extra 0.01 second off each page load by shortening everything to one letter variables, using multi-level ternary logic and other cleverness but this really is nothing compared to the headaches you will be causing yourself and everyone else around you.

Name your variables appropriately, document your code, choose clarity over brevity. Even better, use standardized object oriented code which more or less documents itself without the need for a lot of inline comments.

PHP has been around for a long while now, websites have been made for even longer. Chances are that whatever you need to make, someone has made before. Don't be afraid to lean on others for support, Github is your friend, Composer is your friend, Packagist is your friend.

From loggers to color manipulation tools, from profilers to unit testing frameworks, from Mailchimp APIs to Twitter Bootstrap, everything is available at the push of a button (or typing of a command), use them!

10. Do Not Neglect Other Languages

If you're a PHP person it is now standard practice to know about HTML, CSS, Javascript and MySQL at least. When you have a pretty good handle on these languages it's time to learn Javascript again. **Javascript is not jQuery**. You should learn Javascript properly to be able to utilize it efficiently.

I would also recommend learning all about object oriented PHP, it is a life-saver and will make your code better by orders of magnitude. It will also open doors to languages like C# and Java, they will be a lot easier to understand with OOP under your belt.

Broaden your knowledge by learning about package managers, build scripts, Coffeescript, LESS, SASS, YAML, templating engines and other awesome tools. I would heartily recommend looking at other frameworks, Laravel in particular.

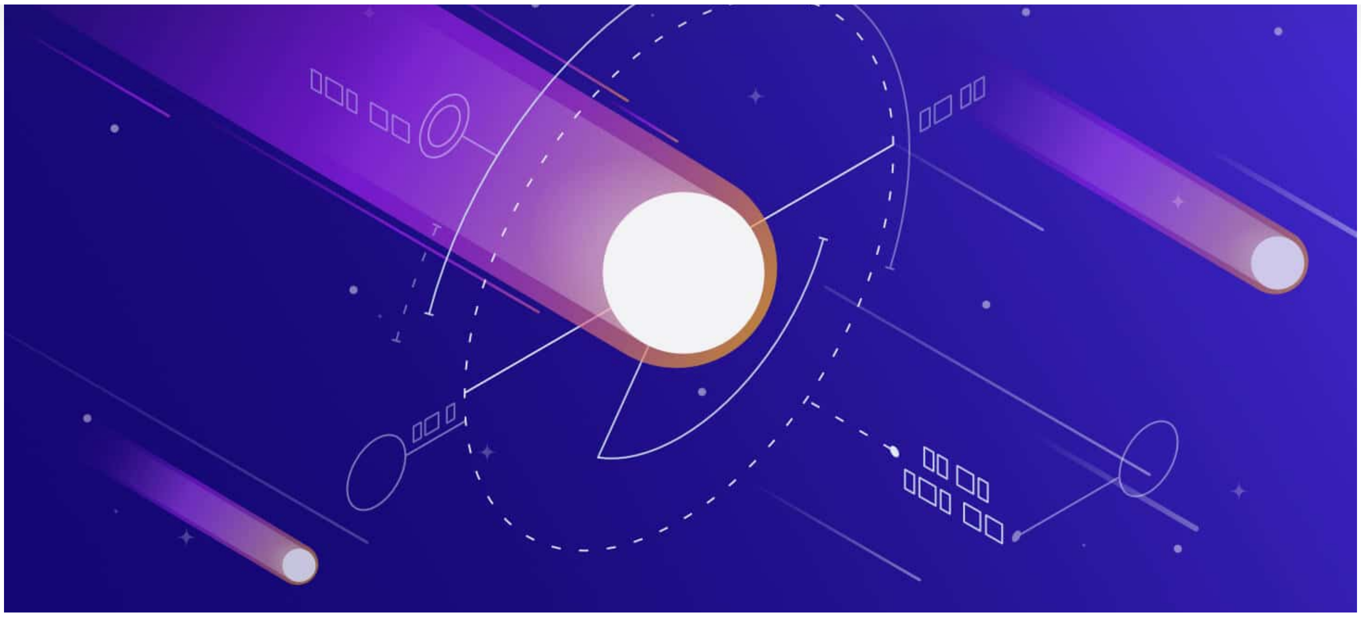
When you're doing pretty well with these, what about Ruby, Ruby on Rails, app development for Android, iPhone, Windows Phone? You'd think that there is no point because these fall outside your comfort zone and work needs, but that's just the point. Each language has something useful to teach and a little extra knowledge

If you enjoyed this article, then you'll love Kinsta's WordPress hosting platform. Turbocharge your website and get 24x7 support from our veteran WordPress team. Our Google Cloud powered infrastructure focuses on auto-scaling, performance, and security. Let us show you the Kinsta difference! Check out our plans

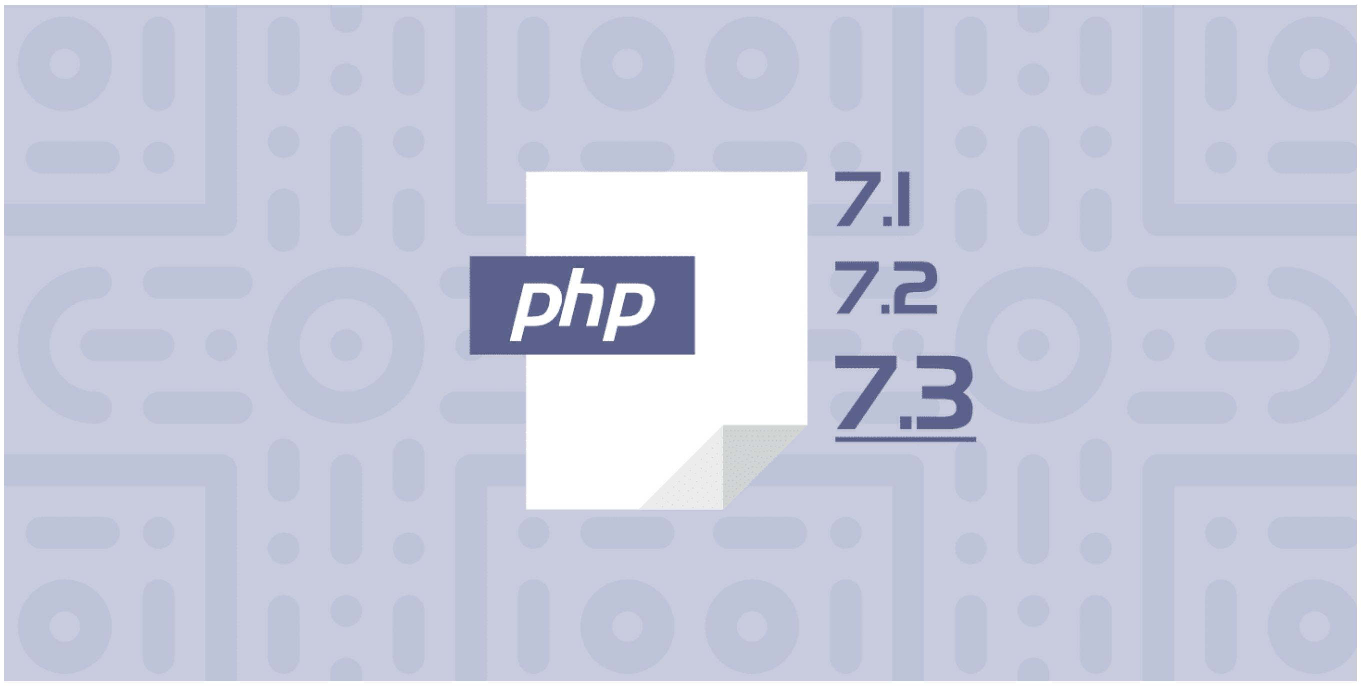
Hand-picked related articles

What's New in PHP 7.3 (Now Available at Kinsta)

Kinsta Premium Managed WordPress Hosting



The Definitive PHP 5.6, 7.0, 7.1, 7.2 & HHVM Benchmarks (2018)



Why You Should Be Using Supported PHP Versions

Alec Kinnear • November 10, 2015 at 8:40 pm

Thanks for the tips. I especially like the “Do not try to be clever one.” Lots of trouble to be saved there.

On the other hand, I’d actually say it’s better to stay in one framework or stack and go really deep. The generalist coders I know are a lot slower than the specialists. Of course having at least one other programming language to keep an eye on can widen intellectual horizons.

Our company flourishes thanks to staying focused on the WordPress stack. I’d hate to admit Joomla or Drupal back in. There’s not enough man hours to be expert in all of them.

Juan • November 11, 2015 at 8:26 am

Hi, i think that steps 4, 5 and 6 (mainly the 4th step) are only based in personal opinion and should be removed from this article to not confuse php beginners.

Daniel Pataki • November 12, 2015 at 8:55 am

Hi Juan,

Not really. Passing by reference is usually a lot slower execution-wise and also a lot more confusing. Queries in a loop can be ok sometimes, but open your code up to all sorts of issues if anything goes wrong. Also, using * in queries is not recommended for the same reason. If you don’t need it, don’t query or it.

That said, there ARE legitimate uses for each, these are general guidelines.

If passing by reference is really “a lot slower”, then there is something implemented very badly in PHP. Passing by reference means passing a pointer. If you pass e.g. an array by value it has to be copied. And PHP arrays are surely quite heavy. I also find it OK that some array functions like sort work by reference. But I agree, you should not pass by reference if you are not sure what you are doing.

But usually the plain PHP code is not the problem. Usually the database queries are the problem. (And too much client-server-communication for loading images, scripts, ads, …). So keeping the number of queries low is a good idea. I saw code where someone had a function which loaded some basic data from the database. And he called the function all over the scripts which created one page. Even within loops. So not only for beginners it’s a good idea to log database queries while developing to see what’s going on in the database when a page is created.

I never use SELECT * FROM … (just for ad hoc queries). But I’m pretty sure that this would not be a performance problem. This could become a problem if you load 100,000 or millions of rows – but then you have another problem. So for some hundreds of rows it’s probably not noticeable if you load 10 columns or 50 columns per row. Though: you should not do it! And if you would do it all the time memory usage could become an issue.

Denis Komnenovic • March 23, 2016 at 3:38 am

>>>But usually the plain PHP code is not the problem. Usually the database queries are the problem<<<
Sorry, but this is nonsense, to say that the Php-Code are not the problem. If you pass your sql queries through Php, you have so many possibilities to do this, if you don't use a smart and good Orm-Framework which defends your from writing some bad code. I was probably spend the half of my career just to repair some php-nonsense and wastefull php-programming-code and repair security

programmers, cause the most of them, was web "designers" and not programmers or software developers, which doesn't have any ideas about software-design-patterns, performance algor … and the result of real bad coded programmes, modules, extensions, how ever – specially in the typo3-world, where the web-workers spend to much time to learn some typoscript-hacks (some unimportant and unnecessary scripting-language with out any logic) and they was thinking just cause they use a object-reference to some method there programming in the oo-paradigma. Real software-developers and engineers are people who can programming in higher programming languages like Java, C++ or Perl cause they understand much more about hows programming languages and interpreters working. But i have to say, with Php7, which i was testing the sins last year, this certain stances will be changed, it is a big evolution and serious programming language and i believe the Php7-world will be much more professional as the world around php4 and the first releases of php5.

Hannes • March 24, 2016 at 5:53 am

My statement was meant for programmers who know what they are doing. I have to admit that I do not have that much experience with PHP optimizations but it was an issue from time to time in the last 15 years. Usually I had to deal with bigger (today I would say medium sized) databases. ORM were usually not used. When we subtracted the total query time from the total page creation time the time spent with PHP code execution was mostly not an issue. And my statement was also an answer about the post “that passing by reference is slower”!? I’m pretty sure that it’s the other way around and that it should not be an issue in most cases.

But I also had code from people who did not know what they are doing. E.g. calling a Get-function all over the page creation

Today with frameworks and OR-mappers many parts of the actually executed code is hidden from the developer. It's convenient for developer but often much more data is loaded from the database than actually needed and OR-mapper need quite some logic to do their stuff. But as computers are much faster today, it's a common approach to use these frameworks to write code faster and more easily. Even if the code is a bit slower than it could be. Further optimization should be done only if it's necessary.

But even with this frameworks people should know what they are doing. You should monitor the database queries to get a feeling what the ORM mapper does and how you could possibly reduce the number of slow queries.

Denis Komnenovic • March 24, 2016 at 8:01 am

Ok, from this aspect you are totally right and i agree with your other statements and it is true that sometimes it's not easy to write a complex and optimal sql query in a same time and of course the OR-mapper-frameworks cannot replace the knowledge about some more complex and deeper sql queries, which can waist much of performance. But when software-engineers talking about that, that optimizations should be done at least, they actually talking about higher programming languages like Java, Perl or C++, not Php, cause you have stuff like faster static arrays, dynamic, but more slow arraylists, hashmaps, hashtables, linkedlist and so on, some times you it is easier to not the fastest data-construct or many castings and converting stories and so on, all of stuff which are not present in "Php4 or Php5", even not in Php7, so if you just have to think about dynamic arrays and you has no brain pain with castings and reverse castings of complex types (not just primitive data types), it should not be a problem to develop be slower – this is really a joke, isn't.

Jeffery Ahern • September 23, 2016 at 4:26 pm

For PHP5+ beginners – 4th is the most important step in this article. Pass by ref in PHP is not the same as in C (http://php.net/manual/en/language.references.arent.php). And most beginners don't understand the difference between PHP and C reference, which tends to lead to Premature Optimization (http://c2.com/cgi/wiki?PrematureOptimization).

The Zend engine has optimizations that handle that for you, and when you pass by ref, you deoptimize the zend engine and have a performance hit because of the deoptimize. What the author should have done was cite the reason why.

And before the comment "That does not really make much sense. That is some really really really inefficient reference handling", please have some knowledge about pass by ref in PHP.

I'm not making a claim that PHP is better than any language, I just have gotten tired of pulling that deoptimization out of code bases.

Juan • September 26, 2016 at 4:46 am

When i said that step 4 was based in personal opinion i was refering to "One example of why references are bad is PHP built in shuffle() or sort(). Instead of returning a shuffled or sorted array, they modify the original which is completely illogical to my mind." specifically to "is completely illogical to my mind".

Pass by reference is completely logical when you need it, eg: when you need to modify the value inside the function.

You are right, for simple values or small arrays that you dont need to modify inside the function its faster to pass by value (in PHP). But when pass it by reference and modify it inside the function.

Francis Kim • November 11, 2015 at 10:48 am

Really great tips here, thanks.

Face Down • November 11, 2015 at 4:03 pm

All these tips are well-known and everyone should follow them even with very old php versions, not just follow them now, when php 7 will release.

Kim • November 12, 2015 at 5:36 am

Thanks:)

atif • November 12, 2015 at 7:39 am

A great Article. But i would like to add that few things were general for example u said naming variables with such names which gives sense to other programmers too. I agree but it is a general in PHP programming not specific to PHP7.

Siempre Makinando • November 12, 2015 at 9:09 am

I don’t think sort(), which changes the original variable has a wrong behavior. I think you waste memory if you have both variables (sorted and unsorted) at the same time

napolux • November 12, 2015 at 11:09 am

Always the same stuff since PHP5…

Regis FLORET • August 8, 2017 at 12:42 am

You mean : Always the same stuff from far before PHP first release ?

João Paulo Cercal • November 16, 2015 at 6:09 pm

I really think that this article is valid to all versions of PHP language. Thanks Mr Pataki.

tbelknap • November 16, 2015 at 8:44 pm

So… you say a lot about WordPress. But does WordPress plan on supporting this new version officially?

GnRDenver • November 23, 2015 at 10:00 am

Probably when php5.6 is 2 years past EOL :p

rvm • November 19, 2015 at 9:00 am

Hitesh Parekh • November 30, 2015 at 5:58 am

This may not be just for PHP7 only, but it can be applied to other versions greater than PHP4. And to add PHP is not HTML.

Jason Knight • December 21, 2015 at 3:12 am

The pass by reference thing is a bunch of nonsense if you know the first blasted thing about how programming languages work… As someone who still writes assembly, I'm aware of what's REALLY going on with that… Honestly, functions that return copies of things piss me off as they are wasting time AND memory!

THOUGH THAT DEPENDS ON WHAT YOU ARE PASSING!

Simple fact is, what in PHP is called "passing by reference' in a real language is called using pointers. On a 64 bit platform that's probably under the hood in PHP… 64 bits (duh) — 8 bytes. Since ALL PHP objects are pointer based abstractions thanks to the lack of strict typecasting (booh, hiss) that's the smallest size you'll EVER see used for passing or referring to a variable.

When you pass a parameter normally in 99% of programming languages where you end up with a copy you can modify freely without changing the original, a copy of the variable is placed on what's called the "stack" — the value in it's typecast size is placed there… in PHP's case I would assume it's passing pointers, so when you pass a value normally it not only puts the pointer to the copy on the stack, but the copy too.

What's going to be faster, passing 8 bytes to point at the original, or 8 bytes plus however big the variable is in memory. Between memory allocation, de-allocation, Garbage collection and stack fragmentation (since I doubt they maintain both a stack and a heap old-school) that pass by reference will be faster and use less memory. Anyone saying otherwise doesn't understand how variables work, particularly arguments and locals!

nilbud • February 16, 2016 at 10:57 am

You're bang on the money these rank amateurs who think confidence is a substitute for knowledge are a pain in the hole.

Thomas Sherwin • June 30, 2016 at 8:11 am

You can't start off as a pro but in our current environment it seems like many people are expected to know more than is realistic for someone to know. People making hiring decisions have no idea how to evaluate technical or programming skills. This leads to a lot of people bluffing about what they know and trying to be confident about it. In reality they're just struggling because they don't have solid fundamentals.

frickenate • February 29, 2016 at 3:02 pm

If you even use PHP, you probably started using it back in version 4.x where references were more commonly required. In PHP 5+, unless you are writing a function or method that is modifying the passed argument, you do not want to use a reference. It's actually *less* performant to pass read-only arguments by reference. PHP is not C, and is not part of the "99% of programming languages" you mention. PHP's references are not strictly equatable to pointers from other languages. PHP uses copy-on-write, which in fact requires less tracking overhead by the Zend engine than references do.

"if you know the first blasted thing about how programming languages work" – pretty harsh criticism considering you're the one misinformed about how variables in PHP work.

Jason Knight • March 1, 2016 at 1:11 am

Since pass by reference by definition is a pointer — unless they've gone and made it grossly inefficient and jacktarded — ANY time the data is larger than the size of a pointer passing that data would be less efficient as you'd have to make a complete copy of that data.

Again, what's going to be faster, passing a 8 byte pointer to a multi-kilobyte string, array, or object — or making a complete separate copy of that large data set? It's just common sense.

The only way your argument would make ANY sense is if they are making a complete copy either way, AND writing to both that copy AND the remote one on change — which would be such an idiotic mouth-breathing dumbass halfwit a way of doing things, any programmer that would do that needs a good swift kick in the groin!

If it doesn't work that way in PHP, then its even DUMBER than I thought. Honestly given some of the "insecure by design" practices that would be quite an accomplishment on their part.

Of course with ALL PHP variables in fact being variable type objects, I very much doubt that a 32 bit integer only consumes 4 bytes… probably more like 32. I should dig into the code base and have a look at that whilst I'm in there trying to make include/require refuse to read anything that doesn't end in .php and to make all other file operations incapable of reading/writing same. You know, plugging one of the most obvious security holes in the language?

Sorry, I've been programming in everything from hand assembling machine language and entering it on toggle switches, to writing my own P-Code interpreter. What you are saying makes little to no sense unless PHP is even MORE jacktarded than I already thought under the hood.

But then, the only reason I use PHP at all is it has whitespace neutral strings making it easier to work with markup, you can guarantee it will be the late 1980's. It's JUST enough to overcome all the ridiculous (and often absurd) aspects of the language.

---

frickenate • March 1, 2016 at 1:18 pm

> Since pass by reference by definition is a pointer

This is the notion you need to forget. PHP references are not pointers. References in PHP are aliases, like file system hard links. With $a = 1; $b = &$a; the variable $b is not a pointer to $a – instead, $a and $b are identical aliases each pointing to the same zval internally.

Similar with $a = 1; $b = $a; – no copy is created. Both $a and $b point to the same zval struct, and is refcounted. A new zval is only created if you modify $a or $b. This is copy-on-write.

I would go into more detailed description of the internals, but I don't fully understand the changes in PHP 7's zval struct to explain the differences compared to PHP 5. Suffice it to say, if you use references in cases where you are not modifying the caller's copy, you are trying to optimize something that PHP already handles with copy-on-write. And worst case scenario, PHP actually has to make copies to break references. In PHP 5 if you pass a variable by reference to a method and then call something like strlen() on it, a copy is made to break the reference. This is apparently not the case in PHP 7, but it is said that references still incur more tracking overhead than copy-on-write.

Source: https://nikic.github.io/2015/05/05/Internal-value-representation-in-PHP-7-part-1.html

---

Jason Knight • March 1, 2016 at 2:17 pm "reference" or "alias" not simply be a pointer to the original, since all you're doing is OPERATING ON THE ORIGINAL?!? What legitimate reason is there for that apart from developer ineptitude and/or ignorance?

alias/reference/whatever trendy bullshit new name for the same thing is popular today — You pass a pointer to the original, as PHP tracks refreences you increment it's references as part of the parent object… on exit you decrease the references, DONE. When references hits zero, you free up the memory block.

It should not be "making a copy" just because something else is referencing it. What possible bullshit reason could they have for doubling down on memory use like that?!?

The past .. decade or so, I've heard of some STUPID shit in programming languages; just look at the mouth-breathing idiocy known as HTML/CSS/JS "frameworks" — but if what you are describing is what's going on… Wow, that's some halfwit inept bullshit right there!

It would LITERALLY mean these folks aren't qualified to be creating programming languages.

frickenate • March 1, 2016 at 2:29 pm

> PHP tracks refreences you increment it's references as part of the parent object… on exit you decrease the references, DONE. When references hits zero, you free up the memory block.

That's exactly what it does… it tracks it as a single zval struct with flags: a counter for refcounting of copy-on-write to prevent copies, and a flag to track whether it's a reference. It's just that rather than $a explicitly referencing $b, $a and $b are basically

Jason Knight • March 1, 2016 at 3:38 pm

Actually I scanned it on the first reply, but at your prodding I re-read it and… uhm… they're actually saying what I'm saying and it doesn't support what you're saying at all.

Their use of the terminology is rubbish and gibberish, since they seem to be saying alias and reference when they MEAN pointer, but…

Though your comparison to filesystem aliases is much the same nonsense; it's not like an alias on a filesystem is making a complete copy, it's a pointer to the original — well, to be more specific a directory entry containing a name and a pointer to the original. How do you think they work? It's the same as a http redirect, it's not copying the entire site, it's just pointing at it! That's what poitners DO!

zval (php5) and _zend_value (php7) are simply the structure stored on the heap, with references being separate from them:

… and I quote:

>First of all, note that the value union is now 8 bytes instead of 16.
>It will only store integers (lval) and doubles (dval) directly,
>everything else is a pointer. All the pointer types (apart from
> those marked as special above) use refcounting and have
>a common header defined by zend_refcounted

zend_refcounted being a pointer control structure.

Of course the article you linked to doesn't even MENTION passing by reference, only reference control structures… So I'm really not sure where you're coming up with your claims. In fact that article in a ridiculously convoluted way just says PHP 5 did (which seems a bit silly, but whatever) to it's own control structure — which seems kind-of pointless unless the implementation is as needlessly and pointlessly overthought as the explanation on that site.

At which point I'm wondering if you even know what passing by reference means.

Mitch J.V. • September 9, 2016 at 1:46 am

Jason, please don't procreate. I will fund required contraception. The amount of bullshit you spewed is enough for manure application of entire Brasil.

I sincerely doubt you have any ability as a programmer. If you did, you'd read the fucking manual before acting all high and mighty.

Do you even have a job?

Jason Knight • September 10, 2016 at 2:42 am

Hey look, a six month late bounce just to go 100% ad-hominem. Shall we call your parents and see if they think you're ready to have an adult discussion?

You SOUND like one of those Sitepoint suck-up sycophants…

Mitch J.V. • September 10, 2016 at 3:45 pm angry asshole would come up with anything better. I don't know what Sitepoint suck-ups are, I just know that Sitepoint is a breeding place of retards who think they are programmers, offense intended.

Now that you've been offended, please go play with your "assembly" somewhere else and don't contribute to the world, we have enough scum already, no need for you to add up to the pile. Piece of shit.

Jeffery Ahern • September 23, 2016 at 12:35 pm

For someone who has such a breath of languages and the years of experience that you claim to have, I would have thought that by now, you would understand that not all languages are the same. Nor should they ever be used as the same. PHP is NOT C. PHP is written in C(well, mostly).

I run into this alot working in the PHP community. Someone who comes from a non PHP background and does not bother to look at the PHP manual because 99% of programming languages are "the same". Next time you make assumptions about a language, at least do the common curtisy of looking to make sure that is true in that language. PHP is a "higher" (used very loosely) language that does not handle everything the way you may thing it does. http://php.net/manual/en/language.references.pass.php is PHP documentation. and you can look at the source code for how it handles that if you so choose.

@frickenate:disqus already went into the reasons why, and I understand that you are arguing based on your experience with other languages, but please remember, its PHP, and people spreading ignorance like your comment about how bad PHP is, is well, just that. And some Junior level developer is going to read your comment, place some code in because of it, and I will have to remove it, because what you suggest is a de-optimization.

Item #9 and #10 contradict #4. Overall, removing passing by reference "programming capabilities' broke code and makes is very hard and costly to change code, in particular our PHP extensions. It doesn't make sense to put costly migration and "porting" barriers up to support PHP7.

Ilyas Mimo • May 31, 2016 at 4:02 am

"Do Not Pass By Reference …" !! Should be reviewed in this article

Bangali Beta • June 27, 2016 at 1:22 pm

Hello,

I'm a complete beginner in Php. So that means, I don't know Php 7 or any of the previous versions.
Now, I want to start learning Php but from the latest version 7 only.
So, where can I learn Php 7 as a beginner ? I mean, which online text/video tutorial would teach me Php 7 from the beginning ?
Most tutorials I find assume I know Php 5 and only mention what's new in Php 7 and just teach what's new in Php 7. Those tutorials are no good to me since I don't know Php 5 or any previous versions. I need a tutrial that would teach me orthodox Php 7 (and not frameworks, etc.) and object oriented php (OOP) which would teach me right from the beginning starting from the "Hello World!". I think you know what I mean.
(NOTE: Not interested in learning Php 5 or any previous versions. Why learn old timers which are about to go extinct. Plus, in Php 7, you write less code and the page loading are twice as faster. Why should I learn old timers where the code would have to be double long resulting in pages loading double slow compared to Php 7 ?).

I do have some programming background but not in Php.

Thank You

Ali

Inferno2ss • July 24, 2016 at 9:45 am

I have to disagree with the * in MYSQL queries. I have some large column tables than I join together and as stupid as it sounds and is, they actually run 25% faster when I do a SELECT a.*, b.* FROM as opposed to SELECT a.id, a.first_nm, a.last_nm, b.city, b.address or whatever. I know it uses more IO and network bandwidth and seems stupid, but that is MYSQL for you.

Kelvin • January 7, 2019 at 12:00 am

I have been writing MYSQL queries for a long time and i think i agree with this statement.
SELECT * FROM table

Performs better than the
SELECT name, age, ** FROM table

Someone can confirm the time to execute this two statements

Arthur Kushmantsev • August 7, 2016 at 4:48 am

Ah. never mind – 2 items just about SQL and the final – 8. Do Not Try To Be Clever, made the switch looooool, ROFL

mindpower • December 8, 2016 at 2:03 pm

1) learn a better language :-D

There's always a better language than the language you do

mindpower • August 26, 2017 at 12:50 pm

True, but there's a hundred languages better than PHP.

E pluribus unum • August 26, 2017 at 12:59 pm

Meh, better I doubt it, unless you don't know what you're talking about. Languages are made for an application or a purpose. If you compare C and PHP then you don't understand what PHP is for… if you compare Javascript and PHP… same, and so on. Are there better languages? meh… maybe, but they don't do specifically what PHP does with the same ease.
So the choice of a language is usually dictated by what you have to do… so there's an application for Perl, one for PHP, one for Javascript, one for Java, one for Ruby, one for C, one for C++, one for C#, one for Bash, one for Sh… and so on… if you still think that "there's one language better than them all" then you still need to grow

mindpower • August 26, 2017 at 1:15 pm

I know what PHP is for. It's for people who don't know other languages so they do not realise what a clusterfuck of a language PHP is.

There's just no valid excuse for using PHP in today's world. It's quite honestly the worst option available and you will do well to move on and learn a grown up language.

E pluribus unum • August 27, 2017 at 3:30 am

I guess you're just a bad, close minded, programmer.

mindpower • August 28, 2017 at 2:13 am

Hurling insults when you can't defend your language? Says a lot.

A bad, closed-minded programmer makes excuses for a bad language and doesn't explore alternatives.

Jason Knight • August 28, 2017 at 11:35 pm

Until you need whitespace neutral strings for easier to maintain output, easier access to database fields where the field type on return could be any type, or that with PHP 7 it's now faster for raw computing tasks than Java or .NET.

Sure, it has it's problems… but what for web development would you suggest as an alternative? Have you seen the idiotic halfwit rubbish .NET ends up being once Visual Studio gets its grubby mitts on it? Have you ever TRIED to make easy to debug through properly formatted output using C? Have you ever used Ruby or Python for HTML? They're a joke!

I mean hell, I'd love to use ADA… on most counts it's the best language out there from a security and logic standpoint; it also inhales upon the proverbial equine of short stature when it comes to doing something like outputting HTML.

I'll be first to admit PHP has a LOT of problems, but for web development PARTICULARLY as of PHP 7 it has a lot LESS of them than the alternatives!

content...

Ivan Zarechnii • December 30, 2016 at 2:07 pm

This is noobest article ever just hype of php 7. Not a word about new features in php 7 or what it makes obsolete. 80% of this advice's been there for 10+ years

8,9,10 should be flushed down the toilet especially 9 YES GO REINVENT THE WHEEL AND A BICYCLE ALSO BECAUSE UNLESS YOU DO YOU ARE GOING TO BE LAB ASSISTANT AND NOT ACTUAL SCIENTIST HE DISCOVERS AND DOES THINGS. IF GUY WHO INVENTED MVC PATTERN THOUGHT LIKE THIS HE WOULD NEVER INVENT IT. SHAME ON Daniel Pataki

Mike • April 1, 2017 at 2:51 pm

"Do Not Write Wasteful Code" – Everything depends on a lot of things. For personal website you can make a lot of files and enjoy the crazy schema in hundreds of folders and files. For high-loaded projects this may kill your disk IO rate, and finally damage your file system pretty soon.
"Do Not Pass By Reference If Not Needed" – It is proved and easy to check that passing data by reference increases performance significantly, unless it is not a print(hello) script.
"Do Not Try To Be Clever" – Made me laughing all the day)) No comment. It is the worse motto for beginners..

E pluribus unum • June 13, 2017 at 6:19 pm

the mysql -> mysqli thing makes me laugh. as if you could not have already implemented mysql in your own set of classes. Sometimes deprecation is really stupid... the "because you could use it wrong" is really really demented.

Regis FLORET • August 8, 2017 at 12:47 am

On point 6
Since when you didn't benchmark `select count(*) from WHATEVER` and `select count(id) from WHATEVER` ?

The storage engines are now far more optimized since PHP 5.
The point 6 should be "Know your storage engine and learn SQL language as well as your main stream language (here: PHP)"

Luiz Filipe Machado Barni • August 17, 2017 at 6:44 am

Try not to do these things in PHP >= 0.1, not just PHP 7.0, what about 7.1?

BM • January 6, 2018 at 8:28 pm

YAML is for people who want to bloat their files, press more buttons, and have no idea how to use a text editor.

IMHO it isn't worth the headaches and should never have been accepted as a subset of JSON.

Shyam • January 16, 2018 at 11:20 pm

This are common practices NOT for PHP7 ONLY. I was misguided to this article from search engine. I was hoping something will be relevant to PHP7 only.
Good article though.

---

do not use frameworks :)

Albert Einstein • March 3, 2018 at 2:29 am

But dont act you are clever than other if you just use someones code. Dont be an arrogant if you're simply not reinvented the wheel, I rather be an inventor than building wheels and pretend its my own invention.

Bruno Della Motta • March 15, 2018 at 1:19 pm

OOP sucks… Most frameworks don't pass benchmark tests…

Robert Brown • November 12, 2018 at 1:29 am

This is a very good informative blog. PHP frameworks are is the most powerful open source platform. Easy to configure, responsive web page development.

Mohan Lal • January 27, 2019 at 9:56 pm

Hey, is this is a new version of PHP. I never use this but i think the things you mentioned above are also applicable for all PHP.

webdesigner • May 27, 2019 at 4:00 pm

Code doesn't have to be "elegant" it has to bloody work.

christoper stalin • May 31, 2019 at 6:23 am

Thanks for sharing this information and more informative than another blog.

Leave a Reply

Comment policy: We love comments and appreciate the time that readers spend to share ideas and give feedback. However, all comments are manually moderated and those deemed to be spam or solely promotional will be deleted.

Comment

Name

Email

☐ I agree to the Terms and Conditions and Privacy Policy

POST COMMENT

Features

Managed hosting

Advanced features

WooCommerce hosting

Enterprise hosting

Secure hosting

Free migrations

Company

About us

Clients & case studies

Careers

Press

Partners

Affiliate program

Why us

Resource Center

Knowledge Base

Learn WordPress

Blog

Affiliate Academy

Kinsta vs WP Engine

Kinsta vs SiteGround

Kinsta vs Flywheel

Speed up WordPress

HAVE A QUESTION?

Follow us

MyKinsta

Login

What is MyKinsta?

System status

Feature updates

English